A Survey of Two and Three Dimensional Convex Hull Algorithms

Deepak Maniyani (UFID: 9399-4924)

University of Florida, Gainesville, Florida, united states, deepak.maniyani@ufl.edu

ABSTRACT

This paper will discuss about the algorithms used in computation of convex hull for a given set of points in two and three dimensional Euclidean space. We will be going through the basic concepts used in three dimensional computational geometry. We will also be highlighting how important output sensitiveness is for any practical implementation of convex hull algorithm. We will be mainly focusing on two of the most practical algorithms for the convex hull problem.

1.1 INTRODUCTION

From the given a set of points on a cartesian plane, we have to form a smallest convex polygon using these points in such a way that all the points in the given set either will lie in the interior of the convex polygon or the edge of the polygon. Convex hull problem is one of the most intensively studied problems in computational geometry. It has a wide range of applications like obstacle detection and collision avoidance, mesh generation, clustering, file searching, image processing, urban planning etc. First we will discuss the importance of output sensitiveness for convex hull problem and we introduce interior elimination method by Akl-Toussaint[3]. Then we will review Chan's algorithm[2] and Quickhull algorithm[1] for two and three dimension. And finally we will conclude by listing the determining facts while choosing an convex hull algorithm for implementation.

1.1.1 Output sensitive algorithms:

If the running time complexity of an algorithm can be expressed in terms of its output size then it is an output sensitive algorithm. It is one of the important features to have in a convex hull algorithm, as in most of the practical applications the number of vertices in the boundary of polygon is very less compared to the total number of vertices inside the interior. Also using reduction we can prove that the lower bound for comparative sorting applies to convex hull computation which is $\Omega(n \log n)$. So only way to get around that will be to make our algorithm output dependent. Jarvis march algorithm, "The ultimate planar convex hull algorithm" by Kirkpatrick and Raimund Seidel are one of the earliest output sensitive algorithm designed for convex hull problem. Another approach to improve the output sensitiveness of the algorithm will be to eliminate all the points which can't be part of the convex hull [3]. This idea was proposed in the heuristic convex hull algorithm by Selim Akl and G. T. Toussaint published in 1978 [3]. Their algorithm is based on the following simple steps:

1. Calculate the extreme endpoints in both x and y coordinates.
2. Eliminate all the points that lies in the interior of the polygon or polytope formed by joining these points

A similar approach can be used higher dimensions. Akl-Toussaint is used even today as a pre-processing steps in the convex hull algorithms.

1.2 CHAN'S ALGORITHM:

Chan's algorithm is a very famous optimal output sensitive algorithm for the convex hull problem. Like Johnson's algorithm for shortest paths it combines two other existing algorithms to produce a smart solution. Chan's algorithm can be seen as a special version of the Jarvis's march gift wrapping algorithm. It uses Graham's scan algorithm internally. Interestingly both are slower algorithm compared to Chan's. Graham scan requires sorting of the points as a pre-processing step which bounds the algorithm to have a minimum complexity of $O(n \log n)$. Jarvis march is again an output sensitive algorithm. It does an angle search for each vertex belongs to the convex hull which costs us $O(n)$ time a total of *O(hn*) time complexity; where h is the number of vertices in the convex hull. Chan's algorithm computes the convex hull in *O(nlogh)* time bound.

1.2.1 Algorithm:

1. Partition the point into in to set of size 'm' where m lies between 1 and n
2. For each of the ⌈n/m⌉ group of point we calculate convex hull separately using Gram's Scan
3. Pick a right most (any extreme end will do) point P_0 in the x-coordinate. From P_0 calculate tangents to all the other groups. Pick the steepest tangent as an edge to the convex hull
4. Repeat the step 3 for the newly added edge and repeat until all the points are added to the convex hull or 'm' times whichever is earlier.

1.2.2 Observation:

In step 2 we can use Gram's Scan or any other algorithm which will guaranty us with a better worst case performance. So each iteration will take *O(mlogm)* time which leads us to having a total pre-processing time of *O((n/m) (m log m)) = O(n log m).* Now we are left with the wrapping step where we will be using Jarvis march algorithm. Finding the tangent from one point to another point which lies on another group's convex hull which is a convex m-gon takes *O(logm)* time[2]. And we will be computing for all groups (n/m) and for each point on the convex hull (h). So total time taken by this algorithm is

$$O(nlogm+ h((n/m)logm)) = O(n(1+h/m)logm)$$

If we substitute m = h in the above equation the complexity reduces to *O(n(1+ h/m)log m).* But we don't know the value of h in prior. So the Chan uses a technique similar one used in the Chazelle and Matousek's algorithm [4]. From the steps mentioned above it is clear that if we choose the value of m anywhere below h the algorithm will returns an error. So this observation leads us to a guessing technique by increasing the value of m. But we have to increase the number rapidly with every iteration otherwise it will bring down the performance. So we increase m's value exponentially. We are able to do so because even if m is greater than h such that $m = hk$ for some constant *k*, still we will be within the running time of *O(nlogh)*. So sequences of guesses will be $m = 2^{2^t}$ for t=1,2,3.. while m<n

So guess cost is = n.$log\ 2^{2^t}$ = n.2^t

$$\sum_{t=1}^{\lceil loglogh \rceil} \mathrm{n.2t} = \mathrm{O}(\mathrm{n}.2^{\lceil loglogh \rceil+1}) = \mathrm{O(nlogh)}$$

1.2.3 Chan's algorithm for three-dimension:

Now we will analyse the application of the Chan's algorithm for three dimension. For that we need to know how to apply the underlying Jarvis march and Graham's scan algorithm to three dimension. The higher-dimensional analogue of Jarvis's march is Chand and Kapur's [5] gift-wrapping method [2].In fact Jarvis march algorithm is a two dimensional specialization of the former.

1.2.2. Observations from Chand Kapur's gift wrapping method:

1. Given an initial facet f_0 we can compute three adjacent facet by determining each edge e by choosing a point p that maximizes the angle between f_0 and *conv(e U {p})* (obviously p should not be part of the edge e).
2. Using Breadth first or Depth first pattern we can rest of the facets by repeating the above steps
3. We will use memoization to store the computed results to avoid redundant calculations.
4. A total of 3(2h - 4) wrapping steps are required. So the running time of the algorithm is *O(nh).*

Similarly for computing the initial convex hull computation of the each of the group members of size 'm' instead of Graham's Scan we use Preparata and Hong's three dimensional convex hull algorithm [6]. So now the only remaining steps is to compute the tangent. Here we have consider the difference that the computation of tangent is not to a polygon but to a polyhedron. It require *O(n)* time complexity. Here a special data structure called Dobkin and Kirkpatrick hierarchical representation is used in order to achieve a time complexity of *O(logm)*

1.2.5 Chan's three dimensional algorithm steps:

1. Partition the points into groups of size m

2. For each group Compute conv(Pi) by Preparata and Hong's algorithm and store it in a Dobkin-Kirkpatrick hierarchy
3. Calculate the initial facet f_0
4. Use the Chand Kapur’s method to compute the rest of the facets

This steps are repeated for the different values of ‘m’ until *$m \geq h$.* by Preparata and Hong's algorithm has same complexity as Graham’s which is *O(mlogm).* So similar to the 2-d method with sequence of guesses of the value of ‘m’ we attain the same time complexity of *O(nlogh)*

1.3 QUICKHULL ALGORITHM:

The name comes from the way the algorithm partitions the input in the similar fashion of the quick sort. Similar to Quicksort although the algorithm don’t provide a good worst case performance compared to the others, it is one of the highly used convex hull algorithm for implementation purpose.

1.3.1 Algorithm:

The algorithm contains following steps for the 2- dimensional problem:

1. Find the left most and right most points by finding minimum and maximum x coordinates. Since we know that these will always be part of the convex hull we add them to the convex hull set.
2. The line connecting the left most and right most point will divide the rest of the points into two sets and we process them recursively.
3. Determine the farthest point from the line and add it to the convex hull set.
4. All the points inside triangle formed by the three point are eliminated because they can’t be part of the convex hull
5. Repeat step 3 and 4 until all points are either lie inside the convex hull or are the vertices of the convex hull

. We can observe that the steps in the Quckhull is very much similar to the Akl-Toussaint heuristic algorithm. So the performance of the algorithm will be highly dependent upon the input.

1.3.2 Quickhull algorithm for three-dimension:

Quickhull has two basic operations: i) Oriented hyper plane ii) signed distance to hyperplane [3]. In 2-d they are analogues to a line and the Cartesian distance of a point from the line. The first step in the Quickhull is to compute a maximal simplex which is a tetrahedron for 3-d.

1. Select the first three point by using the 1-3 steps in the above algorithm for 2-d. Triangle formed by these points gives us the oriented hyper plane. Now using the signed distance concept we will select the farthest point from this hyperplane. Connecting these four points gives us the initial hull with a tetrahedron shape.
2. Now for each facet we will repeat the following steps. First we define ‘output set’ and add all the points that are above the facet to it.
3. Calculate the farthest point p in the output set with respect to the facet
4. Now we define ‘visible set’ and add all visible facets to it.
5. For each neighbours in the unvisited facet in the visible set if it is above the point v then we add it to the visible set.
6. Now we will recursively add points to the hull by calculating the horizon. Horizon are basically the edges connecting a visible and a non-visible point from a current view point. For each horizon edge we can form a triangular face and add this facet to the convex hull.
7. Each time we add a new node to the convex hull all points which comes inside of the hull will be eliminated

The worst case performance of the algorithm is *$O(n^2)$.* It has a space complexity of *O(n).* But because of the recursive interior elimination involved in the algorithm it gives a better average running time. Especially when most of the points lies in the interior of the hull, Quickhull algorithm will be a good implementation choice.

1.4 CONCLUSIONS:

Based on Graham scan, wrapping method and elimination procedure mentioned by Akl-Toussaint, there many convex hull algorithms and enhancements that are being developed today. But choosing one of them for implementation will depend upon the characteristics of the problem in hand. Output sensitiveness, average running time and worst case time complexity will be determining factors in choosing such algorithms.

1.5 REFERENCES

[1] C. Bradford Barber, David P. Dobkin and Hannu Huhdanpaa (1996), "The quickhull algorithm for convex hulls", ACM Transactions on Mathematical Software (TOMS), v.22 n.4, p.469-483 DOI: https://doi.org/10.1145/235815.235821
[2] Timothy M. Chan (1996). "Optimal output-sensitive convex hull algorithms in two and three dimensions". Discrete and Computational Geometry, Vol. 16, pp.361–368.
[3] Akl, S.G., Toussaint, G.T. (1978), "A Fast Convex Hull Algorithm", Information Processing Letters 7(5), pages 219-22,2
[4] B. Chazelle and J. Matousek (1995), "Derandomizing an output-sensitive convex hull algorithm in three dimensions", Comput. Geom. Theory Appl., 5:27-32
[5] D. R. Chand and S. S. Kapur (1970) , "An algorithm for convex polytopes", J. Assoc. Comput. Mach., 17:78-86.
[6] E P. Preparata and S. J. Hong (1977), "Convex hulls of finite sets of points in two and three dimensions", Commun. ACM, 20:87-93.