Marching Squares-based Approach to Finding the Convex Hull of a Planar Set of Points

Yong Yue and Carsten Maple

Department of Computing and Information Systems, University of Luton
Park Square, Luton LU1 3JU, United Kingdom

Abstract

Many applications required the use of the convex hull of a finite set of geometric entities in the plane. Finding the convex hull has been a key problem in computational geometry. Although a considerable number of algorithms have been proposed since the 1960s, there are still issues associated with the effectiveness and implementation. This paper presents a practical approach applying the marching squares algorithms to the Jarvis' march for a planar set of points. The points can be obtained from practical situations, for example digitised scanned images.

Keywords: Algorithm, Convex hull, Marching squares, Planar set, Shape analysis.

1. Introduction

The convex hull of a set in space E^d is defined to be the smallest convex-set containing the set. Many areas require the use of convex hulls such as collision avoidance, pattern/feature recognition, image processing, stock cutting and allocation, geometric matching and scheduling, NC machining, and tolerance analysis. It is important to observe that among the applications, the original problem arises often in curved geometry and is approximated by faceting to adapt existing algorithms for the convex hull of a point set.

Considerable effort has been seen on fundamental algorithms for computing the convex hull of point sets, some of the earliest work being [1-3], and on sophisticated techniques to deal with efficiency or special situations such as [4-5]. Although some attempts have been made to tackle the convex hull problems for curvilinear segments [6-9], little published work detailing actual implementations can be found.

The algorithms described in [6] proposed a general approach (edge-based) to the calculation of the convex hulls on polygons with curved segments, but the difficulties of their implementations are unclear. Furthermore, because of their nature, they are unable to exploit analytical solutions.

The work in [7] was based on the Voronoi diagram of planar polygons composed of straight and circular lines. The results of the convex hull and Voronoi diagram for a polygon were linked via the Delaunay triangulation because it was both a partitioning of the convex hull and the dual of the Voronoi diagram. The calculations of the convex hull and Voronoi diagram have the same complexity for a point set. The Jarvis' march-based analytical approach [8] calculated tangents between boundary segments of straight and circular line segments. The implementation is straightforward but it is difficult to extend the method to higher dimensions. Although elegant relationships exist between the two constructions, the algorithms for generating them are different in terms of the difficulty of implementation.

The current work is motivated from a reverse engineering point of view. Scanned images are digitised into scattered points in the plane and the convex hull is often required. The point set can be represented by a matrix of marching squares, two-dimensional form of the marching cubes [10]. This matrix is then used to find the convex hull. It is obvious that the method can be used to approximate the convex hull of a planar set of curvilinear segments.

The rest of the paper will provide a brief description of the marching squares, and then discuss the convex hull calculation. Finally, conclusions and further effort for the work will be made.

2. The Marching Squares

Based on Marching cubes [10], the Marching square can be defined to provide a piecewise-linear approximation of a 2D shape. In defining the Marching squares, each vertex of a square is assigned a value. The value of 0 corresponds to a vertex outside the shape while a non-0 value means the vertex on the boundary or in the interior of the shape. To uniquely define the square for approximation, the lower left vertex is assigned the weighting 0 or 1, the lower right

vertex 0 and 2, the top right vertex 0 and 4 and the top left 0 and 8 (Figure 1).

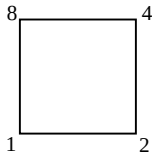

Fig 1: Vertex weighting of a Marching square

There are 16 possible intersections between the 4 vertices and the shape, which can be uniquely identified using binary numbers 0 to 1111, as illustrated in Figure 2 where decimal values are used to represent the intersections.

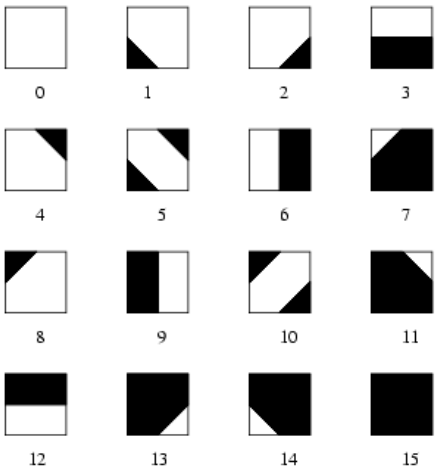

Fig 2: Approximation index of the Marching squares

Uniform grids (squares) are placed over a shape and the Marching square approximation is generated. Figure 3 provides a marching square representation of a shape where the grids are very coarse to depict the effect.

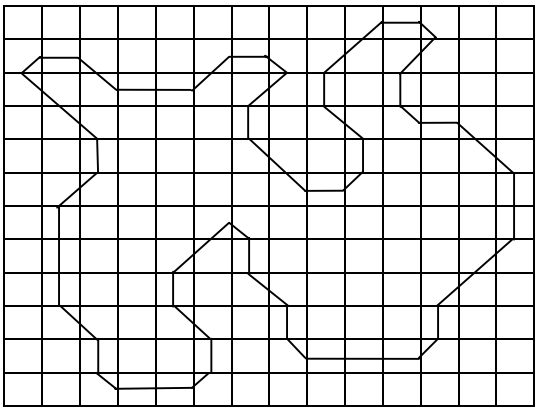

Fig 3: A shape and its approximation with marching squares

A 2D array can then be constructed with the index values of the grids:

0	0	0	0	0	0	0	0	0	2	3	1	0	0
2	3	1	0	0	2	3	1	2	7	13	8	0	0
4	14	11	3	3	7	13	8	6	15	9	0	0	0
0	4	14	15	15	15	9	0	4	14	11	3	1	0
0	0	6	15	15	15	11	3	0	9	15	15	11	1
0	2	7	15	15	15	15	11	3	7	15	15	15	9
0	6	15	15	15	13	14	15	15	15	15	15	15	9
0	6	15	15	13	8	9	15	15	15	15	15	13	8
0	6	15	15	9	0	4	14	15	15	15	13	8	0
0	4	14	15	11	1	0	6	15	15	15	9	0	0
0	0	6	15	15	9	0	4	12	12	12	8	0	0
0	0	4	12	12	8	0	0	0	0	0	0	0	0

Only those squares with a significant index value (i.e. between 1 and 14) are of concern. These squares are termed the boundary squares. Given the marching direction, the index value of a square and the entrant direction of the boundary to the square, the exit direction of the boundary to the square can be uniquely determined. Moreover, the index value of the adjacent square has only a few possibilities. For example, when travelling around the boundary counter-clockwise, a square with an index value of 13 indicates that the boundary edge of the shape continues at the adjacent square on the right; and the index value of the adjacent square must be 8, 10, 12 or 14.

Therefore, a shape can be represented with an array of boundary squares. The index values of the boundary squares for the shape represented in Figure 3 are as follows (starting with the leftmost one among the lowest boundary squares):

4	12	12	8	9	1	11	9	13	8	13	14	9	4
14	6	4	12	12	12	8	9	13	8	13	8	9	9
1	11	1	3	11	9	13	8	1	3	2	7	2	6
4	14	9	7	3	11	3	11	9	13	8	1	3	2
7	3	3	11	1	3	2	4	14	4	14	6	7	2
6	6	6	4	14	6								

3. The Proposed Approach

The proposed method is based on the Jarvis' march [3] which is the simplest instance of Chand and Kapur's "gift wrapping" approach [1], where the wrap would be a rope instead of a sheet. In the Jarvis' march (Figure 4), an origin point outside the point set is found and a radius arm is swung about this origin in an arbitrary direction until a point of the set is met (i.e. the point has the smallest polar angle). This point becomes the first point on the convex hull and is taken as the new origin and a radius arm is swung in the same direction as before until the next convex hull

point is found. This is repeated until all the points are enclosed in the convex hull.

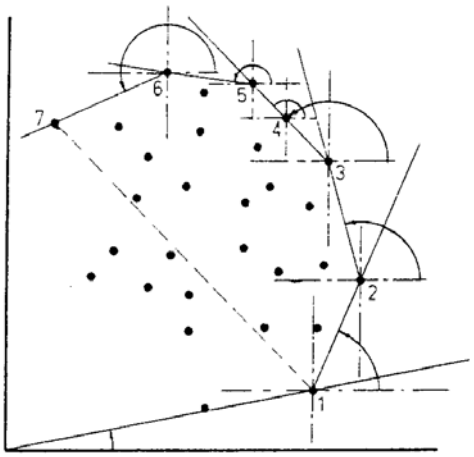

Fig. 4: Jarvis' march.

While Jarvis' march requires a considerable amount of floating point calculations for the angles, our approach takes the advantage of the index values of marching squares for the convex hull computation. Integer comparisons are made and float calculations do not take place until really necessary.

The construction of the convex hull is not approximated although the shape itself is approximated with the marching squares. The two intersection points on the square are used to construct the convex hull. Since the two adjacent boundary squares always share an intersection point, only the second intersection point on the square is used in constructing the convex hull. Therefore, when a boundary square is referred to, it means that its second intersection point is considered.

The proposed method works in two stages. The first stage calculates the boundary squares and converts effective ones among them into candidate points for the convex hull. The second stage evaluates the candidate points and constructs the convex hull using the Jarvis' march.

First of all, it can be noted that the four extreme squares can be found straightaway, and the convex hull construction can be performed in the four quadrants (in terms of the evaluation angles in the Jarvis' march) using the same procedure.

With an analysis of the connective relationships between the boundary squares, a set of rules for the data reduction can be established from the index values of the boundary squares. For Quadrant I (i.e. the evaluation angle is between 0° and 90° for the Jarvis' march), the data reduction rules are discussed below.

Because no angles of 90° or larger are allowed in the first quadrant, the boundary squares noted below have no effect and are not considered for the convex hull construction.

- Any boundary squares, which progress in the -*x* direction, i.e. those boundary squares with an index value of 1, 2, 3, 7 or 11.
- Any boundary squares, which progress in the *y* direction, i.e. those boundary squares with an index value of 6 or 9.
- Non-ending boundary squares (the ending boundary squares being those with an index value of 1, 2, 4 or 8), which cause either concavity or are co-linear with the next boundary square to them, i.e. those boundary squares with an index value of 7, 11, 13 or 14 (note that the index values of 7 and 11 are considered already).
- Any boundary squares, which are co-linear in the middle with other boundary squares. This means that those boundary squares with an index value of 4, 8 or 12 are evaluated.

It should be noted that boundary squares with the index value of 5 or 10 are not considered here since they represent tiny details (as the squares are of a very fine size) in most situations. The algorithm is described below.

Stage 1: reduction of boundary square data

Let v_i be the index value of boundary square b_i for the array of boundary squares representing the shape, where $0 \le i < n$ and n is the number of the boundary squares.

1) Sort the array of boundary squares in the counter-clockwise order so that the lowest-left square becomes the first element in the array.

2) Take the second intersection point of the first boundary square (the lowest-left) b_i ($i = 0$), as the first candidate point for the convex hull to be constructed in the second stage. Interestingly, it can be noted that the index value for this square, v_i is 4.

3) While the next boundary square b_i ($i = i+1$), is not a rightmost in the boundary square matrix: if its index value v_i, is 1, 2, 3, 6, 7, 9, 11, 13, or 14, ignore b_i and repeat this step with the next boundary square b_i ($i = i+2$); otherwise for the index value of 4, 8, and 12, go to Step 4, Step 5 and Step 6 respectively.

4) If the index value for the next boundary square b_i ($i = i+2$), is not 12 or 14, take the second intersection point of the square as the next candidate point for the convex hull; otherwise, ignore the square. Go to Step 3.

5) If the index value for the next boundary square b_i ($i = i+2$), is not 13, take the second intersection point of the square as the next candidate point for the convex hull; otherwise, ignore the square. Go to Step 3.

6) If the index value for the next boundary square b_i ($i = i+2$), is not 12 or 14, take the second intersection point of the square as the next candidate point for the convex hull; otherwise, ignore the square. Go to Step 3.

7) If the boundary is the rightmost, all the candidates points for the convex hull in Quadrant I are selected and the first stage for Quadrant I is completed. Proceed with the second stage.

Stage 2: construction of convex hull

This stage simply performs the Jarvis' march within Quadrant I (i.e. the evaluation angle is from 0° inclusive to 90° exclusive).

These two stages are performed for the other three quadrants one by one. Figure 5 shows the convex hull for the data presented in Figure 3.

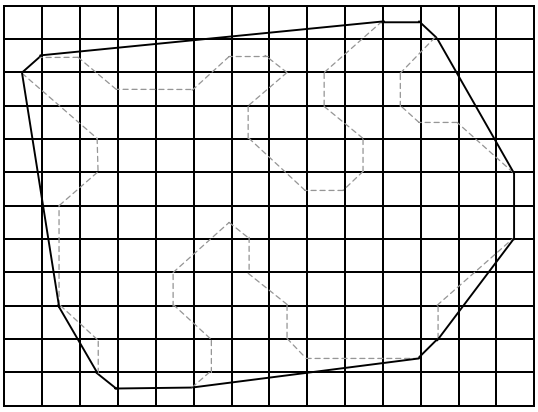

Fig 5: Convex hull obtained with the proposed method

An initial implementation of the algorithm has been carried out in Visual C++. OpenGL has been used as the graphic user interface to display the resulting convex hull. The input to the program is an array of boundary squares representing a shape which could be obtained from digital scanning or simply from a finite planar set of curvilinear line segments.

4. Conclusions

This paper has presented a new approach for finding the convex hull of a finite planar set. The application of the marching squares to the Jarvis march provides an effective means to compute the convex hull of a finite set of points obtained from any geometric entity in a plane through digitisation. From the description of the algorithm, it can be seen that the computations are mainly integer comparisons and float calculations are limited only when they are really necessary. Initial implementation has yielded encouraging results.

Further effort will be made in several respects: a) a comprehensive test of the methods with a wide range of test instances; b) enhancement of the method by employing the rotating squares [11]; and c) extension of the algorithm to 3-dimensional situations.

References

[1] D.R. Chand, and S.S. Kapur, "An algorithm for convex polytopes", *Computing Machinery*, 17(1), pp. 78-86, 1970.

[2] R.M. Graham, "An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set", *Information Processing Letters*, 1(4), pp. 132-133, 1972.

[3] R.A. Jarvis, "On the Identification of the Convex Hull of a Finite Set in the Plane", *Information Processing Letters*, 2(1), pp. 18-21, 1973.

[4] H.E. Bez, and J. Edwards, "Distributed Algorithm for the Planar Convex Hull Problem", *Computer Aided Design*, 22(2), pp. 81-86, 1990.

[5] T.M. Chan, "Dynamic Planar Convex Hull Operations in Near-Logarithmic Amortized Time", *Journal of the ACM*, 48(1), pp. 1-12, 2001.

[6] A.A. Schaffer, and C.J. Van Wyk, "Convex Hulls of Piecewise Smooth Jordan Curves", *Journal of Algorithms*, 8(1), pp. 66-94, 1987.

[7] C.K. Yap, "An O(n log n) Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments", *Discrete and Computational Geometry*, 2(4), pp. 365-93, 1987.

[8] Y. Yue, J.M. Murray, J. Corney, and D.E.R. Clark, "Finding the Convex Hull of a Planar Set of Straight and Circular Line Segments", *Engineering Computations*, 16(8), pp. 858-875, 1999.

[9] J.K. Johnstone, "Giftwrapping a Curve with the Convex Hull", *Proceedings of the 42nd Annual Southeast Regional Conference*, pp. 224-227, Huntsville, Alabama, U.S.A., 2-3 April 2004.

[10] W.E. Lorensen, and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm", *Computer Graphics*, 21(4), pp. 163-169, 1987.

[11] C. Maple, and A. Donafee, "A Boundary Representation and Comparison Technique for 2D Objects", *Proceedings of the 2002 IEEE Computer Society International Conference on Information Visualisation (IV2002)*, pp. 414-419, London, U.K., July 2002.