Computer-Aided Design 70 (2016) 153–160

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

A Total Order Heuristic-Based Convex Hull Algorithm for Points in the Plane☆

Abel J.P. Gomes

Instituto de Telecomunicações, Universidade da Beira Interior, Portugal

HIGHLIGHTS

- We propose a 2D convex hull algorithm based on comparison operators.
- We propose a 2D convex hull algorithm that outperforms Quickhull.
- We propose a 2D non-convex hull algorithm.

ARTICLE INFO

Keywords:
Convex hull
Geometric algorithms
Computational geometry

ABSTRACT

Computing the convex hull of a set of points is a fundamental operation in many research fields, including geometric computing, computer graphics, computer vision, robotics, and so forth. This problem is particularly challenging when the number of points goes beyond some millions. In this article, we describe a very fast algorithm that copes with millions of points in a short period of time without using any kind of parallel computing. This has been made possible because the algorithm reduces to a sorting problem of the input point set, what dramatically minimizes the geometric computations (e.g., angles, distances, and so forth) that are typical in other algorithms. When compared with popular convex hull algorithms (namely, Graham's scan, Andrew's monotone chain, Jarvis' gift wrapping, Chan's, and Quickhull), our algorithm is capable of generating the convex hull of a point set in the plane much faster than those five algorithms without penalties in memory space.

1. Introduction

The *convex hull* $\mathbb{H}(\mathcal{P})$ of a planar point set $\mathcal{P}$ can be defined as the smallest convex polygon that encloses $\mathcal{P}$. Every point of $\mathcal{P}$ belonging to the boundary of $\mathbb{H}(\mathcal{P})$ is called an *extreme vertex*. The notion of convex hull is considered by many as one of the most fundamental geometric structures we find in computational geometry, computer graphics, robotics, etc. [1]. In fact, important problems in computational geometry like Delaunay triangulation, Voronoi diagrams, halfsapce intersection, etc. can be reduced to the problem of computing the convex hull of a set of points [2].

Besides, the problem of finding the convex hull crosses many research domains and applies to an endless number of problems and situations. For example, convex hulls play an important role in computer vision [3], pattern recognition [4,5], visual pattern matching [6], operations research [7], path planning and obstacle avoidance in robotics [8,9], astronomy [10,11], and biology and genetics [12], just to mention a few of them.

The remainder of the paper is organized as follows. Section 2 overviews the prior work on convex hull algorithms. Section 3 describes the TORCH (Total Order-Based Convex Hull) algorithm step by step. Section 4 carries out the complexity analysis of the TORCH algorithm. Section 5 compares the TORCH algorithm to other well-known convex hull algorithm with reference to both arbitrary and definite sets of points in the plane. Section 6 concludes the paper, with some hints for future work.

2. Related work

According to Avis et al. [13], the convex hull algorithms fall into two categories: *graph traversal* and *incremental*. Graham scan [14], Jarvis march [15] and monotone chain [16] are representatives of graph traversal algorithms. In the graph traversal algorithms, the input points work as vertices of a graph whose edges are formed temporarily to check whether two connected edges are convex or not. For example, Graham scan uses the angle between two connected edges to decide about the convexity on the shared vertex.

☆ This paper has been recommended for acceptance by Scott Schaefer and Charlie C.L. Wang.

E-mail address: agomes@di.ubi.pt.

http://dx.doi.org/10.1016/j.cad.2015.07.013
0010-4485/

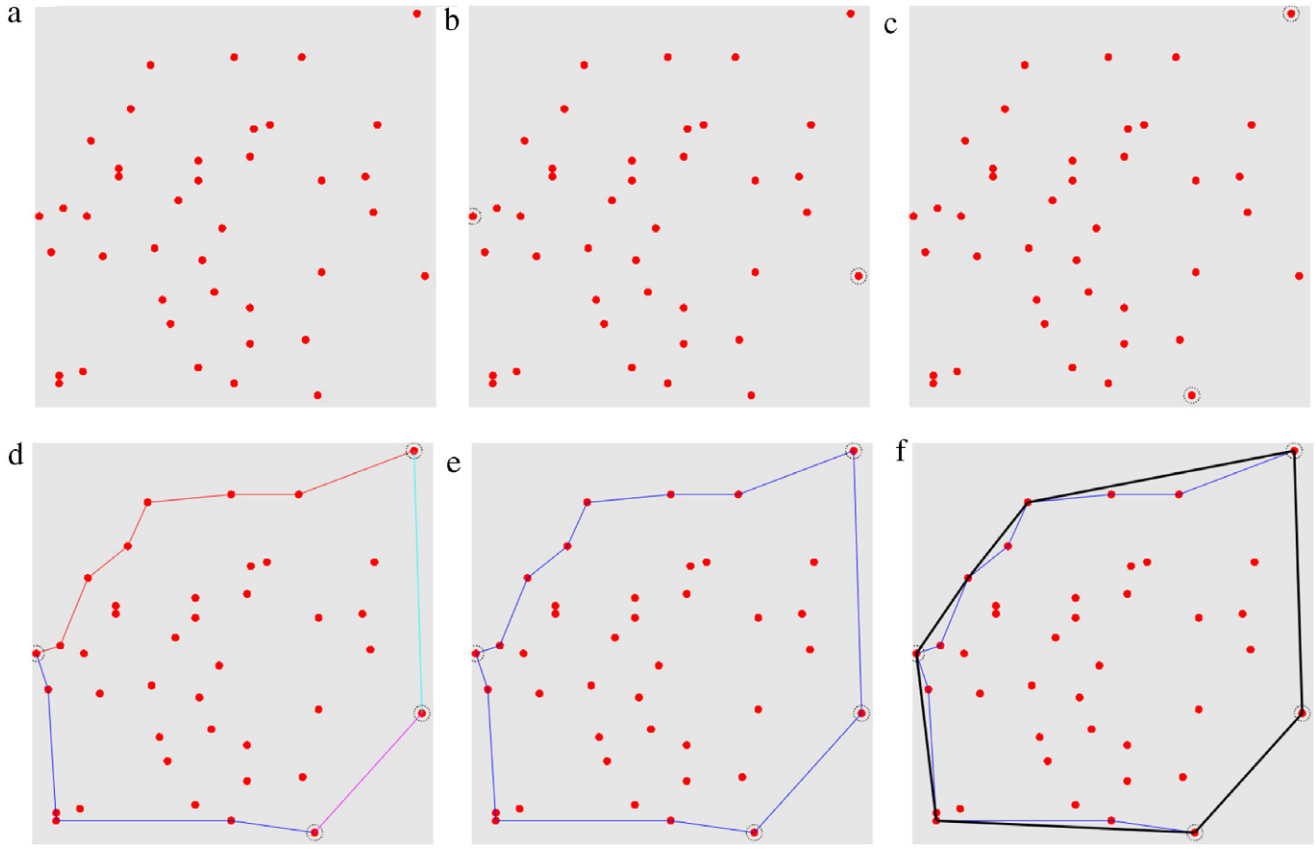

Fig. 1. The algorithm steps: (a) set of 40 points $\mathbb{P}$ sorted along x in the domain $D = [0, 100] \times [0, 100]$; (b) west and east poles found (dotted circles); (c) south and north poles found (dotted circles); (d) lateral hulls: $\mathbb{H}_{SW}$ in blue, $\mathbb{H}_{SE}$ in magenta, $\mathbb{H}_{NE}$ in cyan, and $\mathbb{H}_{NW}$ in red; (e) approximate convex hull $\mathbb{A}$ in blue after connecting lateral hulls counterclockwise; (f) convex hull $\mathbb{H}$ after discarding concave vertices of $\mathbb{A}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Incremental algorithms start from an initial convex hull (e.g., a triangle), checking then whether each of the remaining points belongs to the current convex hull or not. If any of these remaining points is outside the current convex hull, then the convex hull is updated accordingly. Quickhull [17], divide-and-conquer [1] and incremental [18] algorithms are some representatives of this category.

The convex algorithm proposed in this article, called total order heuristic-based convex hull (TORCH) algorithm, is in its essence a sorting algorithm, being its geometric computations reduced to a minimum. As a result of such a sorting procedure we immediately obtain an approximate convex hull (i.e., a non-convex hull) that contains all the extreme vertices of convex hull and a few concave vertices. The last stage of the algorithm consists in discarding those concave vertices from the approximate convex hull using the geometric operation CCW that is employed in Andrew's monotone chain. Our algorithm was designed for serial computing, so that we do not use any GPU resources or other parallel resources in any way. The main contribution of the TORCH algorithm lies in the use of comparison operators to build up the convex hull from the four extremal points. Besides, TORCH outperforms Quickhull, which likely is the fastest amongst the currently known algorithms.

3. The algorithm

The general idea of the algorithm is that a convex hull is like a 2-dimensional ball (or circle), and as every single 2-dimensional ball it has four turning points or poles, that is, west, east, south and north. The west pole is the leftmost point, the east pole is the rightmost point, the south pole is the bottommost point, and the north pole is the topmost point. This is similar to the computation of the elimination quadrangle of the Quickhull algorithm (cf. [17]), whose vertices are precisely those poles.

The algorithm proposed in the present article distinguishes from others, including Andrew's and Quickhull, in the manner how the four hulls between the turning points (or poles) are calculated. In general terms, and after allocating memory for input points, our algorithm consists of the following steps:

(1) Sort the point set in the x-direction of the domain (Fig. 1(a)).
(2) Find the leftmost and rightmost points (inside dotted circles) shown in Fig. 1(b).
(3) Find the bottommost and topmost points (inside dotted circles) shown in Fig. 1(c).
(4) Find the four lateral hulls between turning points (Fig. 1(d)).
(5) Construct approximate convex hull (in blue) by merging the four lateral hulls (Fig. 1(e)).
(6) Inflate the approximate convex hull towards the convex hull in black (Fig. 1(f)). This inflating operation is also called here *convexification*.

Let us then describe these steps of the algorithm in more detail in the following subsections.

3.1. Sorting points

After allocating the entire set $\mathcal{P} = \{p_i\}_{i=0,\ldots,k-1}$ of input points in a 1-dimensional array, we proceed to their lexicographical sorting in the x-direction, as in Andrew's algorithm [16]. This results in a sorted array $\mathbb{P}$. For this purpose, we have adopted the introspective sort (or introsort) algorithm due to Musser [19],