Proceedings of the IASTED International Conference
ADVANCES IN COMPUTER SCIENCE AND TECHNOLOGY
January 23-25, 2006, Puerto Vallarta, Mexico

FAST APPROXIMATION OF CONVEX HULL

Ladislav Kavan
FEE CTU in Prague
Karlovo nam. 13
Prague 2, Czech Republic
email: kavanl1@fel.cvut.cz

Ivana Kolingerova
University of West Bohemia
Univerzitni 8, Box 314
Plzen, Czech Republic
email: kolinger@kiv.zcu.cz

Jiri Zara
FEE CTU in Prague
Karlovo nam. 13
Prague 2, Czech Republic
email: zara@fel.cvut.cz

ABSTRACT
The construction of a planar convex hull is an essential operation in computational geometry. It has been proven that the time complexity of an exact solution is $\Omega(NlogN)$. In this paper, we describe an algorithm with time complexity $O(N + k^2)$, where k is parameter controlling the approximation quality. This is beneficial for applications processing a large number of points without necessity of an exact solution. A formula for upper bound of the approximation error is presented.

KEY WORDS
convex hull, approximation, linear time

1 Introduction

We consider only the planar case, although the generalization to higher dimension would be possible. It has been shown by [6], that an algorithm performing only quadratic tests needs at least $cN \log N$ operations, where c is a constant and N is the number of input points. There exists a variety of algorithms achieving this best possible time complexity, showing that the problem itself has time complexity $\Theta(N \log N)$, see [5, 4]. Very popular algorithms are described in [2, 1].

However, in certain applications we do not need an exact convex hull, i.e. the smallest convex set enclosing given set of points. Instead, *small enough* convex set enclosing input points can be sufficient. We show that in this case we can compute the approximate convex hull with linear time complexity. The inaccuracy of the algorithm can be controlled by user specified parameter k. The total complexity of the resulting algorithm is $O(N + k^2)$. We show that the error of approximation approaches zero as k approaches infinity.

2 Simple Approximation Algorithm

Assume we are given a finite set A of 2D points. Let us denote $CH(A)$ as the (accurate) convex hull of A and suppose that $CH(A)$ contains the origin[1]. It is well known that $CH(A)$ is the intersection of all half-spaces containing A. The first idea to approximate the convex hull, inspired by k-DOP bounding volumes [3], is to restrict the set of intersected half-spaces.

In particular, we confine ourselves to k half-spaces defined as follows. Let angle $\alpha = \frac{2\pi}{k}$. We can partition the plane to sectors centered in the origin and given as

$$S_i = \{x \in R^2 : atan2(x) \in \langle \alpha i, \alpha(i+1))\}$$

$$s_i = S_i \cap A$$

where $i = 0, \ldots, k-1$. The half-spaces will be defined by normal vectors n_i in the centers of the sectors, i.e.

$$n_i = (\cos(\alpha i + \alpha/2), \sin(\alpha i + \alpha/2))$$

More precisely, half-space

$$H_i = \{x \in R^2 : \langle n_i, x\rangle \leq o_i\}$$

where $o_i, i = 0, \ldots, k-1$ are the offsets of the half-spaces.

The first, simple approximation algorithm is straightforward. We set the offsets o_i so that each half-space tightly bounds A in its direction n_i. Formally,

$$o_i = \max_{x \in A} \langle n_i, x\rangle$$

We denote this approximate convex hull given by k half-spaces as

$$ACH_k(A) = \bigcap_{i=0,\ldots,k-1} H_i$$

Notice that $ACH_k(A)$ is really convex, since it is an intersection of closed half-spaces, and moreover it is an outer convex hull, i.e.

$$CH(A) \subseteq ACH_k(A)$$

If $|A| = N$, then the algorithm $ACH_k(A)$ runs in $O(Nk)$ time[2]. This may or may not be better than standard $O(N \log N)$ accurate algorithm, depending on parameter k. As will be discussed later, the variable k controls the error. It is questionable whether this algorithm is better than the accurate ones. However, we will use it for the error analysis of the improved algorithm.

[1]This presumption can be easily satisfied by choosing point from $CH(A)$, for example one point from A, and shifting the other points appropriately. Later we see that it is more advantageous to choose the center of the minimal containing sphere, but this is not as easy to compute.

[2]One could treat k as a constant and thus remove it from the O-notation, but we consider k to be one of the input parameters, because it is tightly connected to the quality of approximation.

3 Improved Approximation Algorithm

The algorithms for convex hulls are inherently as complex as the algorithms for sorting. To obtain truly linear algorithm, we draw the inspiration from linear sorting algorithms, such as radix-sort (bucket-sort)[3].

According to this observation, we find all points $x \in A$ lying in sector s_i. This preprocessing takes time $O(N)$ (truly independent of k) and memory $O(N + k)$. Intuitively, the points in sector s_i will be those giving maximal o_i. However, if we set really

$$o_i^* = \max_{x \in s_i} \langle n_i, x \rangle$$

we could get points of A outside the constructed hull. This is not a problem in itself, because the approximation needs not be an outer approximation, but the drawback is that the distance of a point outside the hull is not bounded[4].

To put the error within control, we must append one more step. For each sector s_i, select the point $a_i \in s_i$ that maximizes the dot product in direction n_i, i.e.

$$\langle n_i, a_i \rangle = \max_{x \in s_i} \langle n_i, x \rangle$$

and account them to the maximum for all sectors

$$o_i' = \max(\max_{x \in s_i} \langle n_i, x \rangle, \max_{j=0,\dots,k-1} \langle n_i, a_j \rangle)$$

The half-spaces

$$H_i' = \{x \in R^2 : \langle n_i, x \rangle \leq o_i'\}$$

form the second approximation of the convex hull,

$$BCH_k(A) = \bigcap_{i=0,\dots,k-1} H_i'$$

Note that $BCH_k(A)$ needs not be a superset of $CH(A)$, but the distance of a point outside $BCH_k(A)$ is bounded, as will be shown in the next section. Notice also

$$BCH_k(A) \subseteq ACH_k(A)$$

because $o_i' \leq o_i$. Intuitively, the $ACH_k(A)$ contains $CH(A)$ and usually is larger. The $BCH_k(A)$ reduces the redundancy of $ACH_k(A)$, but it may be smaller even than $CH(A)$.

Review the steps of the algorithm and their time and memory complexities:

1. distribute the points of A to k sectors – time $O(N)$, memory $O(N + k)$
2. for each sector compute a maximal dot product of contained points. Because every point is multiplied only once, this step takes time $O(N+k)$ and memory $O(k)$ (to store the results and the indexes of extremal points)
3. for each sector, account the extremal points of all other sectors into o_i', requiring time $O(k^2)$ and no memory

Note that in the first step we do not need to really store the distribution of the points to sectors, because the dot product with appropriate n_i can be computed instantly. Thus the total time complexity is $O(N + k^2)$ and the extra memory (not counting the input points) is $O(k)$.

4 Error Analysis

The error of an approximation of the convex hull can be measured as a distance of the point-sets. Recall that the distance of a point x to a set S is defined as

$$dist(x, S) = \inf\{\|x - y\| : y \in S\}$$

The distance of two point sets S, T can be defined as

$$dist(S, T) = \max\left(\sup\{dist(x, T) : x \in S\},\ \sup\{dist(x, S) : x \in T\}\right)$$

which guarantees symmetry: $dist(T, S) = dist(S, T)$. Before estimating the actual error, we must respect different scales of the input set A. We cope with this requirement by assuming that A fits in a circle of radius r centered in the origin.

We begin with analysis of the ACH algorithm, because it will be useful for the subsequent error estimate of the BCH algorithm. Because

$$CH(A) \subseteq ACH_k(A)$$

the first term in $dist(CH(A), ACH_k(A))$

$$\sup\{dist(x, ACH_k(A)) : x \in CH(A)\} = 0$$

it is sufficient to bound the second term

$$\sup\{dist(x, CH(A)) : x \in ACH_k(A)\}$$

We are looking for an upper bound, so we can consider only $CH(A')$, where A' are only those points of A that are lying both on convex hull and approximate convex hull, i.e.

$$A' = A \cap \partial CH(A) \cap \partial ACH_k(A)$$

denoting the point-set boundary by ∂. This is correct because

$$dist(CH(A), ACH_k(A)) \leq dist(CH(A'), ACH_k(A))$$

Consider any two points $x, y \in A'$ such that x is a neighbor of y on the $CH(A')$. Because of the way we defined A', x lies on some line l of $ACH_k(A)$, and y lies

[3]The linearity can not be achieved if the sorting algorithm is based on comparison operations. The algorithms like radix-sort (bucket-sort) do not use comparison, so their time complexity is measured by number of other operations, such as array indexing.

[4]Consider a sector containing no points – its offset is zero and the half-space cuts-off points of A in arbitrary distance.

on some line m of $ACH_k(A)$. The lines l and m belong to neighboring normals, so they contain the angle $\pi - \alpha$. Denote the intersection point $l \cap m$ as z. The distance of line xy to $ACH_k(A)$ is actually the distance of the point $z \in ACH_k(A)$ to line xy. It can be verified easily that the maximal distance realizes when $\|x - z\| = \|y - z\|$, see Fig. 1.

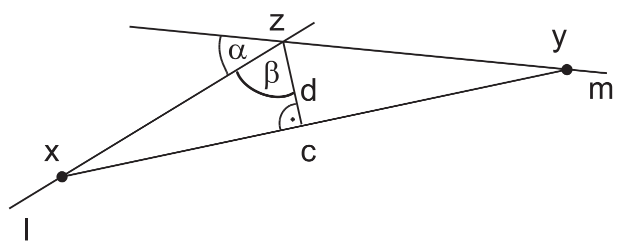

Figure 1. The maximal distance d of $ACH_k(A)$ to $CH(A')$, if the length of the line segment xy is c.

Let $\beta = (\pi - \alpha)/2$, the angle contained by $z, x, (x+y)/2$. We see that $\tan\beta = c/2d$, therefore

$$d = \frac{c}{2\tan\beta}$$

Because the input point-set A is contained in a circle of radius r, we have the inequality $c \leq 2r$, leading to

$$d \leq \frac{r}{\tan\beta} = \frac{r}{\tan\frac{\pi-\alpha}{2}} = r\tan\frac{\alpha}{2}$$

Recall that $\alpha = \frac{2\pi}{k}$ and point out the dependence of d on k,

$$d(k) \leq r\tan\frac{\pi}{k}$$

The $d(k)$ bounds the distance from $ACH_k(A)$ to $CH(A)$. If k goes to infinity, the $ACH_k(A)$ converges to $CH(A)$, because

$$\lim_{k\to\infty} d(k) \leq \lim_{k\to\infty} r\tan\frac{\pi}{k} = 0$$

and $d(k)$ is non-negative.

4.1 Error of the Improved Algorithm

The resulting hull constructed by the BCH algorithm needs not be an outer approximation of the actual convex hull. Naturally we face the question about the maximal distance of a point from $CH(A) - BCH_k(A)$ to $BCH_k(A)$.

Let S_j be an arbitrary sector, $j = 0, \ldots, k-1$, and a_j the maximal point in the direction of this sector, i.e. satisfying

$$\langle n_j, a_j\rangle = \max_{x\in s_j}\langle n_j, x\rangle$$

By the last step of the BCH algorithm, we assured that a_j has been accounted into offsets of all half-spaces $H_i, i = 0, \ldots, k-1$. Therefore, the maximal distance of any point from $CH(A) \cap S_j$ to any half-space H_i is bounded by the length of the secant of sector S_j at distance r (the radius of the containing circle), see Fig. 2.

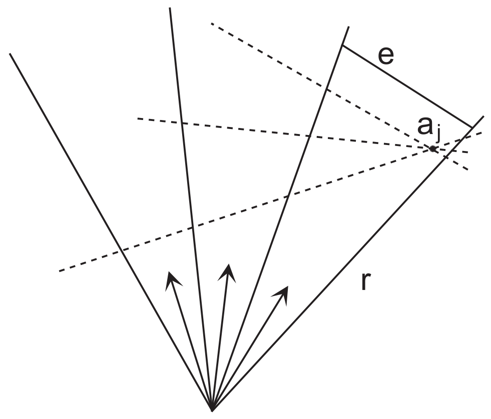

Figure 2. The length e of the secant of sector S_j at distance r from the origin bounds the distance of any point from S_j to the half-spaces (denoted by dashed lines).

The length e of the secant conforms

$$\sin\frac{\alpha}{2} = \frac{e}{2r},\ e = 2r\sin\frac{\alpha}{2}$$

The total error of the BCH algorithm can be expressed as

$$dist(BCH_k(A), CH(A))$$

To bound $dist(BCH_k(A), CH(A))$, we have to bound both

1) $\sup\{dist(x, CH(A)) : x \in BCH_k(A)\}$
2) $\sup\{dist(x, BCH_k(A)) : x \in CH(A)\}$

Intuitively, the first term measures the possible redundancy of $BCH_k(A)$ over $CH(A)$. The second term measures the distance of outliers from $CH(A)$ to $BCH_k(A)$. Because

$$BCH_k(A) \subseteq ACH_k(A)$$

the first term is bounded by

$$\begin{aligned}\sup\{dist(x, CH(A)) : x \in BCH_k(A)\} &\leq \\ \sup\{dist(x, CH(A)) : x \in ACH_k(A)\} &\leq r\tan\frac{\pi}{k}\end{aligned}$$

as shown in the ACH algorithm analysis. The second term conforms

$$\sup\{dist(x, BCH_k(A)) : x \in CH(A)\} \leq 2r\sin\frac{\alpha}{2}$$

as shown in the beginning of this section. Using $\alpha = \frac{2\pi}{k}$ we get

$$2r\sin\frac{\alpha}{2} \leq 2r\sin\frac{\pi}{k}$$

To sum up, the resulting error upper bound is

$$\max\left(r \tan\frac{\pi}{k}, 2r\sin\frac{\pi}{k}\right)$$

According to our expectation, the resulting error is proportional to the radius r of the input set A. If k approaches infinity, the $BCH_k(A)$ converges to $CH(A)$, because

$$\lim_{k\to\infty} r\tan\frac{\pi}{k} = 0$$

as well as

$$\lim_{k\to\infty} 2r\sin\frac{\pi}{k} = 0$$

5 Example Outputs

Two examples of an approximate convex hull constructed by the BCH algorithm for the same input set are presented in Fig. 3, 4. The resulting polygon is filled by gray. The sectors are delimited by dashed lines, the other lines represent the half-spaces.

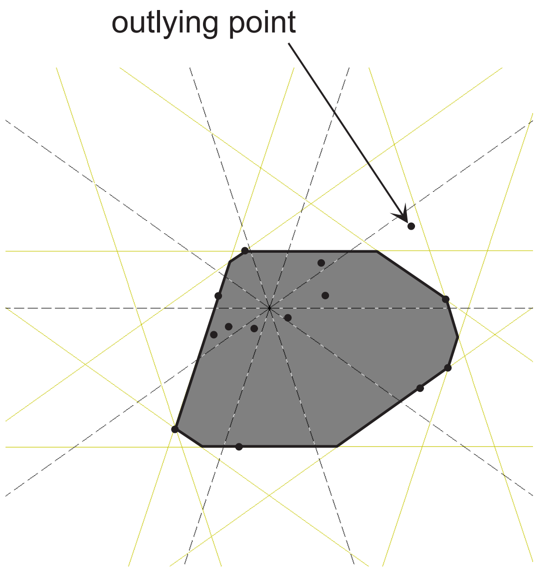

Figure 3. An example of BCH_{10} that does not contain all points of the input set.

The BCH algorithm has one more nice feature: it is an on-line algorithm, meaning that it can quickly recompute $BCH_k(A)$ to $BCH_k(A \cup \{x\})$, where x is a new point. The update can be performed as follows: first, locate the sector S_i such that $x \in S_i$. If $\langle n_i, x\rangle \le \langle n_i, a_i\rangle$, we are done, because the point x is not extremal in its sector. If $\langle n_i, x\rangle > \langle n_i, a_i\rangle$, then set a_i to x and account it into maxima

$$o'_j = \max(o'_j, \langle n_j, a_i\rangle)$$

for each $j = 0, \ldots, k-1$. Obviously, the update operation takes time $O(k)$.

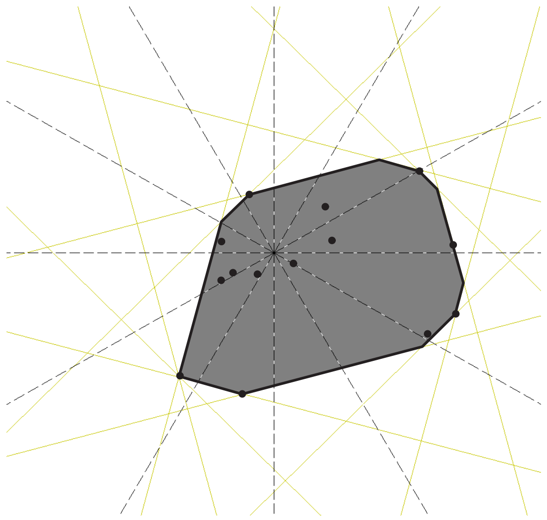

Figure 4. Already BCH_{12} contains all points of the input set. However there is a small redundancy over the accurate convex hull.

Acknowledgements

This work has been partly supported by the Ministry of Education, Youth and Sports of the Czech Republic under research program MSM 6840770014 (Research in the Area of the Prospective Information and Navigation Technologies).

References

[1] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[2] Timothy M. Chan. Output-sensitive results on convex hulls, extreme points, and related problems. In *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*, pages 10–19, New York, NY, USA, 1995. ACM Press.

[3] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, /1998.

[4] J. O'Rourke. *J. Computational Geometry in C, 2nd ed.* Cambridge, England: Cambridge University Press, 1998.

[5] F. Preparata and M. Shamos. *Computational Geometry: An Introduction*. New York: Springer-Verlag, 1985.

[6] Andrew C Yao. A lower bound to finding convex hulls. Technical report, Stanford, CA, USA, 1979.