Creating Convex Hulls in $\mathbb{E}^2$ Using Dual Representation

Ivanka Kolingerová
Department of Computer Science
University of West Bohemia
Americká 42, 306 14 Plzeň
Czech Republic
e-mail kolinger@kiv.zcu.cz

Abstract

The dual representation of points, lines and polygons introduced in [Gun88] can also be used for computing convex hulls of a set of points in $\mathbb{E}^2$. The main principles of the dual representation and a sketch of the algorithm for convex hull computation are given in this paper. Algorithm can be used both for statical and semi-dynamical case. More details can be seen in [Kol94].

1. Introduction

While working on acceleration of the line - polyhedron intersection computation by the means of the dual representation, some other application areas for this type of representation appeared. One of them is the problem of convex hulls construction. It is one of the most frequently solved problems in computer graphics, many algorithms were proposed to it, the complexity of the worst case proved ([Yao81], [Ben83]) - the convex hull of a set of points can be found in time $O(N \log N)$ in the worst case and no substantial efficiency improvement can be done. But, maybe, the solution by the means of the dual representation, as not so frequently used access, could be inspiring for other problems solutions.

We concentrate on construction of the convex hull of a set of points in $\mathbb{E}^2$ in two cases: 1. all data are available at one time (statical problem)
2. the points are coming one after another (semi-dynamical problem)

Probably the best already known statical algorithm was done by [Kis86] and is output-sensitive; that means, that its expected time complexity is closer to $O(N \log K)$ than to $O(N \log N)$, where $N$ is the total number of points and $K$ is the total number of points in convex hull.

The content of the paper is as follows. Chapter 2 gives a short survey of mathematical background. Chapter 3 shows the principles of tests used in the algorithm and the differences between statical and semi-dynamical version. Chapter 4 lists the algorithm. Chapter 5 concludes the paper. Chapter 6 brings references.

2. Mathematical background

The dual representation scheme used in this work was proposed in [Gun88].

Let's denote

$$p = [p_1, p_2]^T$$
and

$$L(a, c) = \{x \in \mathbb{E}^2 : x^T a = c\}, \quad a \in \mathbb{E}^2 \setminus \{0\}, \quad c \in \mathbb{E}, \quad a_2 = 0$$

a non-vertical line in $\mathbb{E}^2$.

Then the dual image $D(p)$ of the point $p \in \mathbb{E}^2$ is the line.

Its equation can be written as

$$x_2 = -p_1 x_1 + p_2,$$
that means that the coordinates of a point are used as coefficients in line equation in a dual space representation.

The dual image $D(L)$ of the line $L$ is the point with coordinates

$$x_1 = a_1 / a_2$$
$$x_2 = c / a_2$$

The dual representation of a convex polygon $P$ in $\mathbb{E}^2$ are two functions named $\text{TOP}^P$ and $\text{BOT}^P : D(\mathbb{E}^2) \rightarrow D(\mathbb{E}^3)$.
These functions can be proved to be piecewise linear, continuous and convex.

More precisely,

$$\text{TOP}^P(x_1^i) = \max_{x_1^i} D(x_1^i)(x_1^i)$$
$$\text{BOT}^P(x_1^i) = \min_{x_1^i} D(x_1^i)(x_1^i)$$

where $D(x_1^i), i=1,2,\ldots,N$ are dual images of the polygon vertices $x_1^i$ (line equations).

See also examples in Fig.1,3.

The dual representation defined in this way has two substantial properties: it doesn't change coincidence and vertical distances (that means, the distances on the vertical
axis). These properties can be utilized for computer graphics problems solution.

For the purposes of our algorithm, we need to decide whether the given point \( x \) lies inside the given polygon \( P \). In dual representation, from the point \( x \) we obtain a line \( D(x) \) and from \( P \) functions \( \top^F \) and \( \bot^F \). The character of these functions and their mutual position imply that \( D(x) \) can have maximally two intersections with them (special cases are not considered, in more details see [Ko194]).

The following situations can appear (see Fig.2):
1. \( x \) lies inside \( P \), \( D(x) \) has no intersections with either \( \top^F \) or \( \bot^F \).
2. \( x \) lies outside \( P \), \( D(x) \) has two intersections with \( \top^F \) or two intersections with \( \bot^F \) or one intersection with both.

With this basic knowledge about dual representation we can advance to the convex hull problem.

3. The main principles used for convex hull construction

The convex hull \( CH(S) \) of a set of points \( S = \{ x_k, k=1,2,...,N \} \) is in fact a polygon the vertices of which are some points from the given set. All the other points stay inside, no point can be outside the area of the polygon \( CH(S) \).

If the convex hull of some of the points (we will denote it \( CH(S_{i-1}) \) for the points \( \{ x_k, k=1,2,...,i-1 \} \)) is given, new point can be inserted or denied according to its position inside/outside \( CH(S_{i-1}) \). This is the problem of point in polygon test which can be done in dual representation.

In order to insert a point \( x_i \) into \( CH(S_{i-1}) \), we must find for it the right place in \( CH(S_{i-1}) \). Point insertion can cause some other point deletion, too. In the dual representation it is not a difficult task as the results of point-inside test can be utilized as:

According to the last chapter, point outside a polygon has two intersections with functions \( \top^B/\bot^B \). Let's suppose that \( D(x_i) \) has two intersections with \( \top^B(S_{i-1}) \) in line segments \( i_1 \) and \( i_2 \), see Fig.4. As \( \top^B(S_{i-1}) \) is convex, its part between both intersections always lies below \( D(x_i) \). That means that this part doesn't satisfy the definition of \( \top^B \) (maxima) and has to be replaced by the line \( D(x_i) \), see Fig.4. After this reconstruction,
if no intersection then g(o) to 12
7. if exist two intersections with $TOPCH(S_{i-1})$ in linear
parts no. k, l then
\begin{verbatim}
begin
Insert D(x_k) into $TOPCH(S_{i-1})$ between $D(x_k)$ and
D(x_l);
\end{verbatim}
if linear parts k, l don't neighbour then
\begin{verbatim}
Delete from $TOPCH(S_{i-1})$ parts from D(x_k+1) up
to D(x_{i-1})
\end{verbatim}
end;
the same as step 7 but with BOT;
9. if exists one intersection with $TOPCH(S_{i-1})$ in linear
part k then
\begin{verbatim}
{ [p_1, p_2] is the endpoint of the linear part k,
  x_k = [p_1, p_2] }
begin
if the part k is not the last one (opened) then
  left := p_1p_1 + p_2 - p_2 > 0
else
  left := p_1p_1 + p_2 - p_2 < 0;
if left then
  Delete from $TOPCH(S_{i-1})$ the left side (up to
  the part k-1) if it exists
else
  Delete from $TOPCH(S_{i-1})$ the right side (from
  the part k+1) if it exists
\end{verbatim}
end;
the same as step 9, but with BOT; in computation of
the flag "left", the relations ">" and "<" have to be
reversed;
\begin{verbatim}
p := p + 1; \{ the total number of the points in the
convex hull \}
\end{verbatim}
10. $TOPCH(S_1):= TOPCH(S_{i-1}), \ BOTCH(S_1):= BOTCH(S_{i-1})$
11. $i := i + 1$
end;
\begin{verbatim}
\{ After finishing the algorithm, the convex hull is given by
functions $TOPCH(S_N)$ and $BOTCH(S_0)$. The total number of points
in the convex hull is p. \}
\end{verbatim}

Let's stop at the complexity of the proposed method. For $N$
points in the set $S$, the time complexity will be $O(N \cdot \log N)$; $f(N)$
is given by the complexity of searching polygon-line
intersection. The solution with $f(N) = K$ is obvious. Memory
demands are $O(N)$ (for dual representation of $CH(S)$). No
preprocessing is necessary.

It is possible to reduce this complexity, if we reduce the
complexity of two critical points: of TOP/BOT update and of
polygon-line intersection computing. The first problem can be
solved simply if we use for TOP/BOT double chained list. Then TOP
and BOT update can be done in $O(1)$. As to the second point, we
need some intersection algorithm with logarithmic time
complexity. The solution from [Meh84] with $(2,4)$-tree can be
used.

With such improvements, we could construct the convex hull of
a set of points in $E^2$ in time $O(N \log K)$ with $O(N)$ preprocessing
(construction of the tree) and with $O(N \log N)$ memory
requirements (tree, chained list).

5. Conclusion

A new algorithm for computation of the convex hull of a set of
points in $E^2$ for statical and semi-dynamical data on the basis of
the dual representation is proposed.

The algorithm can be optimal in time if an algorithm of
computing line-polygon intersection with logarithmical time
complexity is used. It is sensitive to output as its complexity
depends on the number of points which are members of the convex
hull.

6. References

[Ben83] Ben-Or, M. : Lower Bounds for Algebraic Computation
1983, pp. 80-86.
Hull Algorithm \? SIAM J. Comput., Vol.15, No.1, 1986,
pp. 287-299.
[Kol94] Kolingerová, I. : Duální reprezentace a její využití


\[ p_1 = [1, 2]^T, \quad p_2 = [3, 4]^T, \quad L: x_2 - x_1 = -0.5 \]

\[ D(p_1): x_2' = -x_1' + 2, \quad D(p2): x_2' = -3x_1' + 4, \quad D(L) = [1, -0.5]^T. \]

Two points and a line segment and their dual representation

Fig. 1

The possible situations that can appear during point-in-polygon test

Fig. 2
Vertices:
\[ x_1 = [1, 0] \]
\[ x_2 = [0.5, 0.866] \]
\[ x_3 = [-0.5, 0.866] \]
\[ x_4 = [-1, 0] \]
\[ x_5 = [-0.5, -0.866] \]
\[ x_6 = [0.5, -0.866] \]

Edges:
\[ f_1: a = [1.732, -1]^T, c = 1.732 \]
\[ f_2: a = [1.732, 1]^T, c = 1.732 \]
\[ f_3: a = [0.1]^T, c = 0.866 \]
\[ f_4: a = [-1.732, 1]^T, c = 1.732 \]
\[ f_5: a = [-1.732, -1]^T, c = 1.732 \]
\[ f_6: a = [0, -1]^T, c = 0.866 \]

Dual images of the vertices:
\[ D(x_1): x_2' = -x_1' \]
\[ D(x_2): x_2' = -0.5 x_1' + 0.866 \]
\[ D(x_3): x_2' = 0.5 x_1' + 0.866 \]
\[ D(x_4): x_2' = x_1' \]
\[ D(x_5): x_2' = 0.5 x_1' - 0.866 \]
\[ D(x_6): x_2' = -0.5 x_1' - 0.866 \]

The initial convex hull for a set of points

Fig. 5

Examples of mutual position of TOP/BOT\(CH(S_4)\) to \(D(x_5), D(x_6)\)
and \(D(x_7)\)

Fig. 6