Fast approximation of the maximum area convex subset for star-shaped polygons

D. Coeurjolly[1] and J.-M. Chassery[2]

[1]Laboratoire LIRIS, CNRS FRE 2672
Université Claude Bernard Lyon 1,
43, Bd du 11 novembre 1918,
F-69622 Villeurbanne, France

[2]Laboratoire LIS, CNRS UMR 5083
961, rue de la Houille Blanche - BP46,
F-38402 St Martin d'Hères, France

Abstract

Access to the shape by its exterior is classically solved using convex hull. Many algorithms have been proposed in that way. This contribution addresses the problem of the access of the shape by its interior. More precisely, we present a simple algorithm to approximate the maximum convex subset of star-shaped polygons.

1 Introduction

Access to the shape by its exterior is classically solved using convex hull. Many algorithms have been proposed in that way. This contribution addresses the problem of the access of the shape by its interior. The computation of the best shape according to a criterion included in a given one has been studied in many few occasions in the continuous plane.

The problem can be presented as follows: given a non-convex polygon, how to extract the maximum area subset included in that polygon ? In [8], Goodman calls this problem the *potato-peeling problem*. More generally, Chang and Yap [4] define the polygon *inclusion* problem class $Inc(\mathcal{P}, \mathcal{Q}, \mu)$: given a general polygon $P \in \mathcal{P}$, find the μ-largest $Q \in \mathcal{Q}$

contained in P, where $\mathcal{P}$ is a family of polygons, $\mathcal{Q}$ the set of solutions and μ a real function on $\mathcal{Q}$ elements such that

$$\forall Q' \in \mathcal{Q}, \quad Q' \subseteq Q \Rightarrow \mu(Q') \leq \mu(Q). \tag{1}$$

The maximum area convex subset is an inclusion problem where $\mathcal{Q}$ is the family of convex sets and μ gives the area of a solution Q in $\mathcal{Q}$. The inclusion problem arises in many applications where a quick internal approximation of the shape is needed [2, 6].

If we restrict the set $\mathcal{Q}$, efficient solutions exist. For example, if $\mathcal{Q}$ is the set of axis-parallel rectangles, the inclusion problem can be solved in $O(n \cdot \log^2 n)$ if n is the size of P [6]. If $\mathcal{Q}$ is the set of Euclidean disks, the inclusion problem can be related to the skeleton or the medial axis of P [3]. Hence, once the medial axis obtained using [5] for example in $O(n)$, the maximal disk Q is the vertex of the skeleton with the maximal distance value .

In the general case where $\mathcal{Q}$ is the set of convex subsets of P, the first statement of the problem has been given by Goodman [8]. A solution is presented if $n \leq 5$ but the article ends with an open problem concerning the general case. Chang and Yap [4] prove that, in that case, the problem can be solved in $O(n^7)$. However, the proposed algorithm is not tractable in practical applications.

In this article, we present a fast approximation of the maximal convex subset of a polygon if $\mathcal{P}$ is the family of star-shaped polygons. This simple algorithm extracts the maximal convex subset in $O(k \cdot n)$ in the worst case if n is the size of P and k its number of reflex points.

In section 2, we introduce the Chang and Yap's optimal solution definitions that will be used in the rest of the presentation. In section 3, we present the proposed algorithm based on classical and simple geometric tools. Finally, experiments are given.

2 Preliminaries and Exact Polynomial Solution

For the rest of the presentation, we consider a polygon $P = (v_0, v_1, \ldots, v_{n-1})$ with n vertices. We denote by $R = (r_0, r_1, \ldots, r_{k-1})$ the k reflex vertices (or concave vertices) of P (maybe empty). The potato-peeling problem can be expressed as follows,

Problem 1 *Find the maximum area convex subset (MACS for short) Q contained in P.*

In [8], Goodman proves that Q is a convex polygon and that if P is convex (*i.e.* $k = 0$) then Q is equal to P. Furthermore, he presents explicit solutions for $n \leq 5$ and leaves the problem unsolved in the general case.

In [4], Chang and Yap prove that the potato-peeling problem can be solved in polynomial time in the general case. More precisely, they detail an $O(n^7)$ time algorithm to extract Q from P. Since this algorithm uses complex geometric concepts and dynamic programming in several key steps, it is not tractable in practical applications.

Let us present some elements of the Chang and Yap's algorithm. First of all, we define a *chord* of P by a maximal segment fully contained in P. A chord is said to be *extremal* if it contains two or more vertices of P. In particular, an edge of P is always enclosed in an extremal chord. Let $C_1, C_2, \ldots, C_m$ be chords of P with $m \leq k$ such that each C_i passing through reflex vertices of P. We first consider the chords going through a unique reflex point. Let us associate to each such a chord C_i passing through u in R, the closed half-plane C_i^+ defined by C_i and such that at least one or two adjacent vertices to u does not belong to C_i^+. If C_i passes through more than one reflex vertex, the choice of the half-plane can be made in similar ways (see figure 1). They prove that the maximum area convex polygon Q is given by the intersection of P and a set of half-planes defined by a set of so-called optimal chords (see [4]). Hence, to solve the potato-peeling problem, we have to find the appropriate set of optimal chords associated to the reflex vertices.

If P has only one reflex vertex u, the optimal chord C_u that leads to the MACS Q can be easily found. First of all, Chang and Yap [4] define a *butterfly* as a sequence of

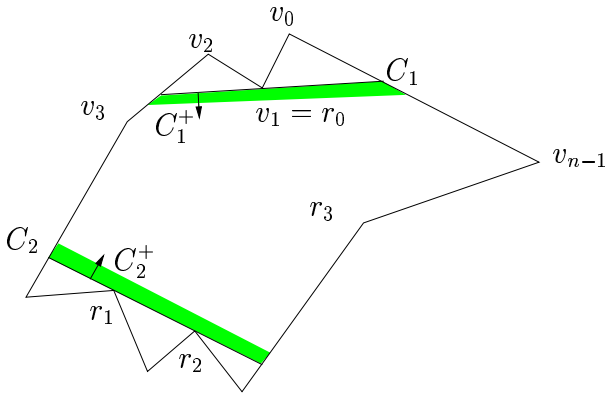

Fig. 1. Notations and illustrations of chords and half-planes generated by these chords.

points $[b', a, u, b, a']$ such that a, u and b are consecutive vertices in P with a and b adjacent vertices of u in P, and such that both $[a, u, a']$ and $[b, u, b']$ are extremal chords (see figure 2). Furthermore, the chord $[c, u, c']$ is said to be *balanced* if we have $|cu| = |uc'|$ (c and c' belonging to P). Based on these definitions, two kinds of butterflies exist according to the position of the intersection O between the straight lines $(b'a)$ and (ba'), and the polygon. More precisely, a *A-butterfly* is such that the points b', a and O (or O, b and a') appear in this order (see figure 2-(left)) in the straight line (ab) (resp. $(a'b')$). Otherwise, it is called a *V-butterfly* (see figure 2-(right)). In this one reflex corner case, Chang and Yap prove the following lemma:

Lemma 2 (Butterfly lemma [4]) *Given a butterfly B and its optimal chord C_u, if B is a V-butterfly then C_u is an extremal chord. If B is a A-butterfly, C_u is either extremal or balanced.*

We notice that in case of V-butterfly we have two possible chords, in case of A-butterfly, three choices are possible. This lemma leads to a linear in time solution to the potato-peeling problem if $k = 1$.

In a general case, Chang and Yap [4] define other geometric objects such as series of A- and V-butterflies. Based on these definitions, they present an A-lemma and a V-lemma, similar to lemma 2, that state that the optimal chords for a set of butterflies are extremal

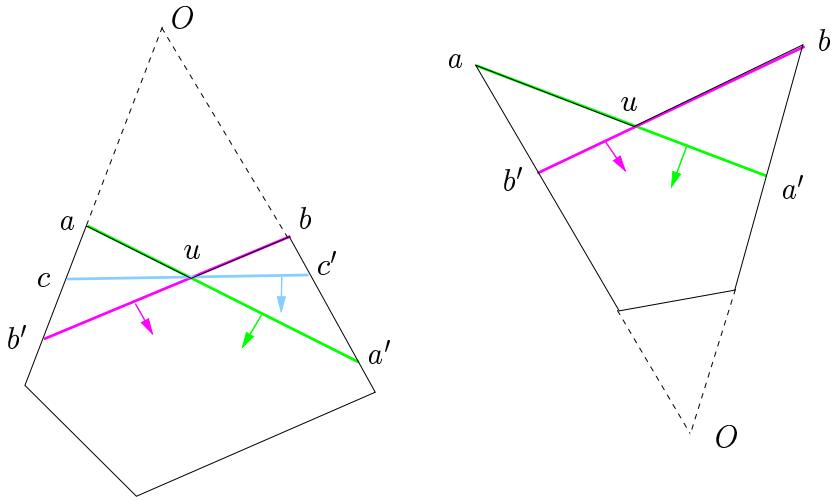

Fig. 2. One reflex vertex case: an A-butterfly (left) and a V-butterfly (right). According to lemma 2, the optimal chord of each polygon is one of the gray segments.

chords or a set of balanced chords. In the general case, the definition of a balanced chain for a butterfly sequence is more complex than in the previous case (see figure 3-(a)). Hence, the computation of such a chain uses dynamic programming and complex geometric concepts. Furthermore this step is the bottleneck of the algorithm and makes expensive the $O(n^7)$ global time complexity.

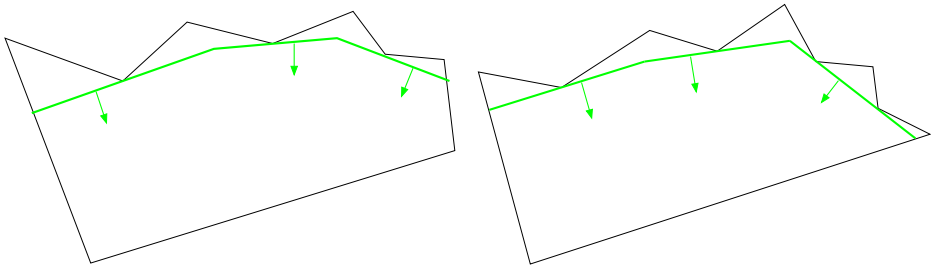

Fig. 3. Examples of balanced chains of A-butterflies series: (a) a balanced chain given by single-pivot chords and (b) given by both single- and double-pivot chords.

To end with definitions, Chang and Yap divide the set of balanced chords into two classes: a balanced chord is called *single-pivot* if it contains only one reflex vertex (chord C_1 in figure

1) and *double-pivot* if it contains two distinct reflex points (chord C_2 in figure 1). Finally, the optimal set of chords that defines the MACS of P contains extremal, balanced single-pivot and balanced double-pivot chords. In figure 3-(a), the solution is composed by only single-pivot chords, in figure 3-(b) both single- and double-pivot chords. Note that for a reflex vertex r_i, from the set of extremal chords associated to r_i, we just have to consider the two extremal chords induced by the two adjacent edges to r_i. In the following, we restrict the definition of an extremal chord to a chord that contains an edge of P incident to a reflex vertex.

Our contribution starts here. In the next section, we present a fast algorithm to approximate the maximum area convex subset star-shaped polygons that uses Chang and Yap's analysis.

3 Fast approximation algorithm

3.1 Kernel dilatation based heuristic

In this section, we assume that P is a star-shaped polygon. First of all, we remind basic definitions of such a polygon. P is a *star-shaped polygon* if there exist a point q in P such that $q\bar{v}_i$ lies inside P for all vertices v_i of P. The set of points q satisfying this property is called the *kernel* of P [10]. Using our definitions and [10], we have:

Proposition 3 *The kernel of P is given by the intersection between P and the half-planes C_i^+ defined by all extremal chords C_i associated to all reflex vertices.*

Figure 4 is an illustration of such proposition. We have the following theorem.

Theorem 4 *Let P be a star-shaped polygon, then its kernel is a subset of the maximum area convex subset of P.*

PROOF. Let C_i be the optimal chord associated to a reflex point r_i of P. We consider the closed space K defined by the intersection between P and the two extremal chords of r_i.

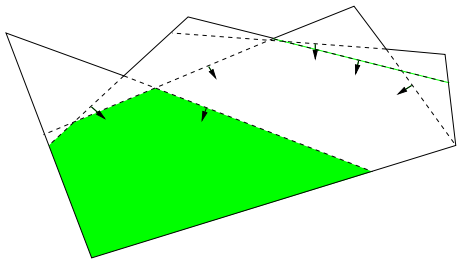

Fig. 4. Illustration of the kernel computation of the example in figure 3-(a).

If C_i is an extremal chord, it is clear that $K \subseteq (C_i^+ \cap P)$. If C_i is a single or double-pivot balanced chord, the slope of C_i is strictly bounded by the slopes of the two extremal chords. Furthermore, since all the half-planes have the same orientation according to P and r_i, we also have $K \subseteq (C_i^+ \cap P)$ (see figure 2-(left) for example). Finally, since the maximum area convex polygon is the intersection between P and the set of half-planes defined by optimal chords, and since the two extremal chords always define a subset to the associated optimal chord, the intersection of all extremal chords is a subset of the MACS of P. With proposition 3, the kernel of P is a convex subset of the MACS. □

In other words, there exists a continuous deformation that transforms the kernel to the MACS. In the following, the strategy we choose to approximate the MACS is to consider the deformation as an Euclidean dilatation of the kernel. Based on this heuristic, several observations can be made: the reflex vertices must be taken into account in the order in which they are reached by the dilatation wavefront. More formally, we consider the list $\mathcal{O}$ of reflex vertices such that the points are sorted according to their minimum distance to the kernel polygon. When a reflex vertex is analyzed, we fix the possible chords and introduce new definitions of chords as follows:

- the chord may be an extremal one as defined by Chang and Yap;
- the chord may be a single-pivot chord such that its slope is tangent to the wavefront (this point will be detailed in the next section);

– the chord may be a double-pivot chord. In that case, the second reflex vertex that belongs to the chord is necessary. It must correspond to the next reflex point in the order $\mathcal{O}$.

Furthermore, when a reflex vertex is analyzed, we choose the chord from this list that maximizes the area of the resulting polygon. If we denote by P' the polygon given by the intersection between P and the half-plane associated to the chosen chord, the chord must maximize the area of P'. In the algorithm, it is equivalent to minimize the area of the removed parts P/P'. Using these heuristics, the approximated MACS algorithm can be easily designed in a greedy process:

1: Compute the kernel of P
2: Compute the ordered list $\mathcal{O}$ of reflex vertices
3: Extract the first point r_1 in $\mathcal{O}$
4: **while** $\mathcal{O}$ is not empty **do**
5: Extract the first point r_2 in $\mathcal{O}$
6: Choose the best chord that maximizes the resulting polygon area with the chords (r_1, r_2)
7: Modify the polygon P accordingly
8: Update the list $\mathcal{O}$ removing reflex points excluded by the chord
9: $r_1 \leftarrow r_2$
10: **end while**

3.2 Algorithms and complexity analysis

In this section, we detail the algorithms and their computational costs. Keep in mind that n denotes the number of vertices of P and k the number of reflex vertices.

Distance-to-kernel computation First of all, the kernel of a polygon can be constructed in $O(n)$ time using the Preparata and Shamos's algorithm [10]. Note that the number of edges of the kernel is $O(k)$ (intersection of $2k$ half-planes). The problem is now to compute the distances between the reflex vertices and the kernel of P denoted $Kern(P)$. A first solution is given by the Edelsbrunner's algorithm that computes the extreme distances between two convex polygons [7]. This algorithm can compute the minimum distance between a point

and a convex polygon in $O(\log k)$ (if k is the number of edges of the convex polygon). Hence, we have a first solution in $O(k \cdot \log k)$ to compute the ordered list $\mathcal{O}$.

However, we use another geometrical tool that will be reused in the next section. Let us consider the generalized Voronoi diagram of $Kern(P)$ [10, 1]. More precisely, we are interested in the exterior part to $Kern(P)$ of the diagram (see figure 5). As $Kern(P)$ is convex, such a diagram is defined by exterior angular bisectors $\{b_i\}_{i=1..5}$ of the kernel vertices. For example in figure 5, all exterior points to $Kern(P)$ located between the angular bisectors b_0 and b_1, are closer to the edge e (extremities are included) than to all other edges of $Kern(P)$. Hence, the minimum distance between such a point and $Kern(P)$ is equal to the distance between the point and the edge e.

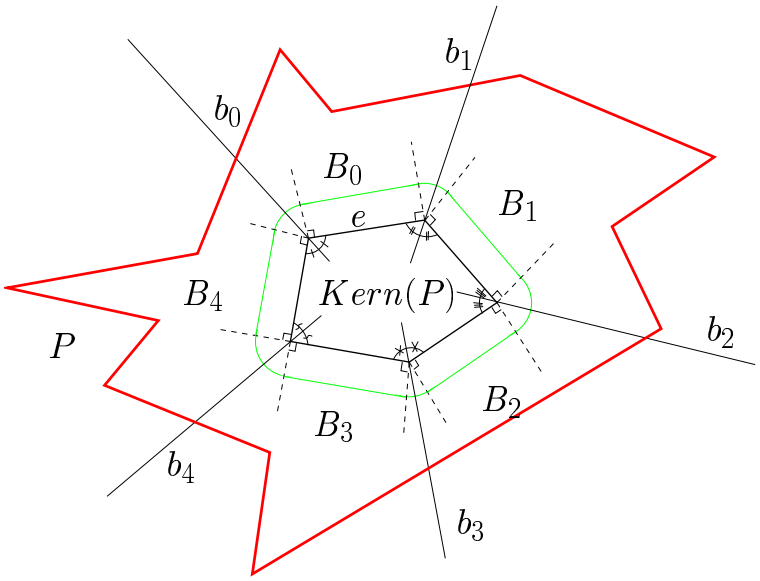

Fig. 5. Euclidean growth of a convex polygon.

To efficiently compute the distance between all reflex vertices to the kernel, we first detail some notations. Let B_m be the open space defined by $Kern(P)$ and the exterior angular bisectors b_m and b_{m+1} (if $m+1$ is equal to $|Kern(P)|$ then b_0 is considered). Hence, the distance to kernel computation step can be viewed as a labelling of vertices of P according to the cell B_m they belong to. To solve this labelling problem, we have the following proposition.

Proposition 5 *The vertices of P that belong to the cell B_m form only one connected piece of P.*

PROOF. Let us consider an half-line starting at a point of $Kern(P)$. By definition of $Kern(P)$, the intersection between such an half-line and P is either a point or an edge of P. The special case when the intersection is an edge of P only occurs when the starting point belongs to an edge of $Kern(P)$ and when the half-line is parallel to this edge (and thus parallel to an extremal chord of a reflex vertex of P). Hence, using the definition of the exterior angular bisectors and given a cell B_m, the half-line b_m (resp. b_{m+1}) crosses P at a point p_m (resp. p_{m+1}) of P. Furthermore, each vertex of P between p_m and p_{m+1} belongs to B_m, otherwise the number of intersections between P and b_m or b_{m+1} should be greater than one. Finally, only one connected piece of P belongs to B_m. □

Then the proposition implies the corollary:

Corollary 6 *The order of the P vertex labels is given by the edges of $Kern(P)$.*

Hence, we can scan both the vertices of P and the edges of $Kern(P)$ to compute the distance-to-kernel. We obtain the following simple algorithm[1]:

1: Find the closest edge e_j of $Kern(P)$ to the point $v_0 \in P$
2: **for** i from 1 to n **do**
3: **while** $d(v_i, e_j) > d(v_i, e_{j+1 \ (mod\ |Kern(P)|)})$ **do**
4: $j{:=}j+1 \ (mod\ |Kern(P)|)$
5: **end while**
6: Store the distance $d(v_i, e_j)$ to v_i
7: **end for**

To detail the computational costs, the step 1 of this algorithm is done in $O(k)$ and the cost of the **For** loop is $O(n)$. As the matter of fact, using Prop. 5 and excepted for the first edge $e_j \in Kern(P)$ of step 1, when we go from an edge $e_{j'}$ to next one, the piece of P associated to the cell B_m defined by $e_{j'}$ is completely computed. Hence, each edge of

[1] $d(v_i, e_j)$ denotes the Euclidean distance between the point v_i and the segment e_j

$Kern(P)$ is visited once during all the process. Note that the first edge e_j of step 1 may be used twice to complete the scan of P.

Finally, the minimum distances between the vertices of P and $Kern(P)$ are computed in $O(n)$. Note that since we are only interested in labelling reflex vertices, the above algorithm can transformed to have a $O(k)$ computational cost. However, a $O(n)$ scanning of P is still needed to identify reflex points. Furthermore, the sorted list $\mathcal{O}$ of reflex points according to such distance is computed in $O(k \cdot \log k)$.

Single-pivot chords computation Given a reflex point r_i of P, we have listed three possible classes of chord: extremal, single-pivot and double-pivot chords. The figure 6 reminds the possibles chords. The extremal and double-pivot chord computation is direct. However, we have to detail the single-pivot chord extraction. According to our heuristic, the single-pivot chord associated to r_i must be tangent to the wavefront propagation of the kernel dilatation.

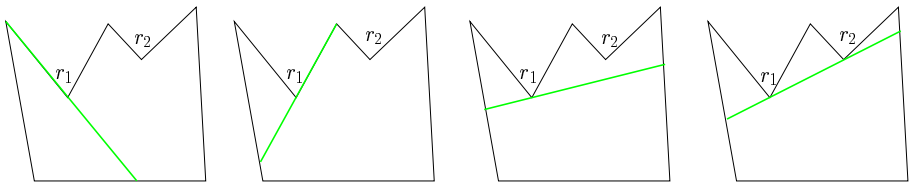

Fig. 6. All possible chords that can be associated to the reflex point r_1 (the two extremal chords, a single-pivot balanced chord and the double-pivot chord).

Using the exterior angular bisector structure we have introduced above, we can efficiently compute the slopes of such chords. In figure 7-(a), let e_1 and e_2 be two adjacent edges of $Kern(P)$ (e_1 and e_2 are incident to the vertex v). Let p (resp. q) be a point in the plane that belongs to the cell generated by e_1 (resp. e_2). We can distinguish two cases: p is closer to e_1 than to one of its extremities and q is closer to v than to e_2 (without the extremities). Hence the straight line going through p and tangent to the wave-front propagation is parallel to e_1. In the second case, the tangent to wavefront straight line going through q is tangent

to the circle of center v and radius $\|\boldsymbol{vq}\|$ (see figure 7). Moreover, two particular cases can be identified (see figure 7-(b)): the first one occurs when the distance between v and q is null, in that case the slope of the chord is the mean of the slopes of edges e_1 and e_2. The second case occurs when the single-pivot chord does not fulfill the maximal chord definition (see the right figure in 7-(b)). In that case, we do not consider this single-pivot chord in the optimal chord choice of line 6 in the main algorithm. Note that this last particular case can also occur with double-pivot chords. In such cases, the chord is not considered too.

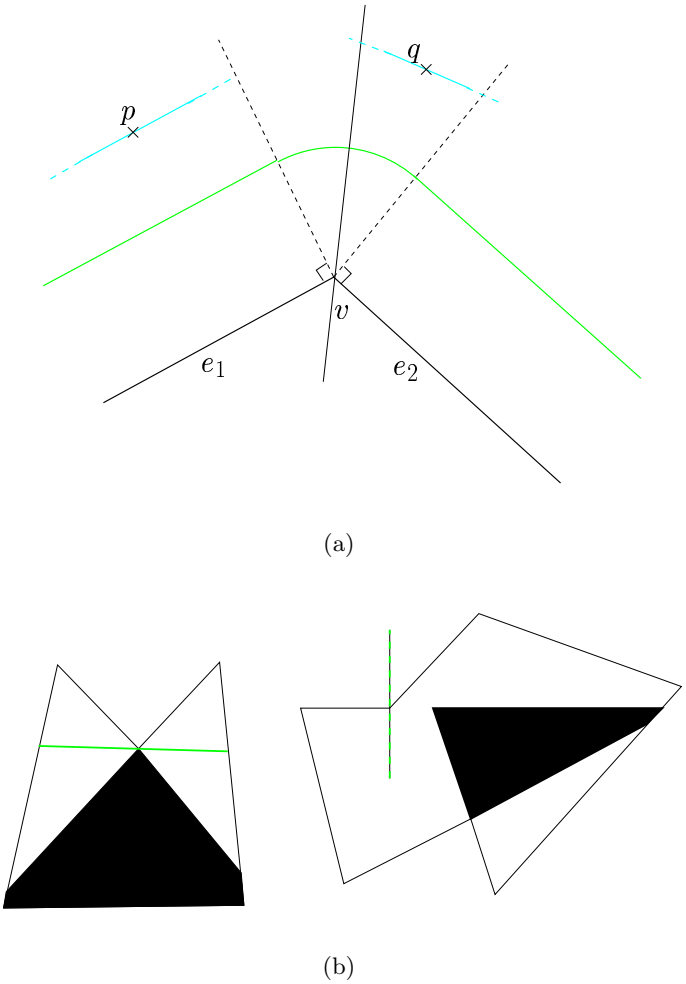

Fig. 7. Computing a chord parallel to the kernel dilatation wavefront: (a) slope computation in the general case, (b) illustration of the two particular cases: the distance between the reflex point and the kernel is null and the single-pivot chord is not a maximal chord.

Finally, if each reflex point r_i of P is labelled according to the closest edge e_i of $Kern(P)$ (extremities included), we can directly compute the single-pivot chord: if r_i is closer to e_j than one of its extremities, the chord is parallel to e_j, otherwise, the chord is tangent to a given circle.

Note that this labelling can be obtained using the previous distance-to-kernel algorithm. Finally, the computation of the k single-pivot chords is done in $O(k)$.

Polygon cut and area evaluation Given a reflex point r_i and a chord C_i either extremal, single-pivot of double-pivot, we have to compute the resulting polygon of the intersection between P and C_i^+. We suppose that the vertices of P are stored as a doubly-connected list.

The cutting algorithm is simple: starting from r_i, the vertices of P are scanned in the two directions until they belong to C_i^+ (see figure 8).

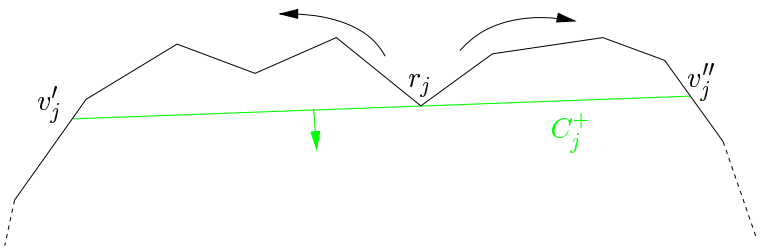

Fig. 8. Illustration of the vertex scan to compute the $(P \cap C_i^+)$ polygon.

During the process, let m be the number of removed vertices. Hence, the polygon cut according to C_i is done in $O(m)$. Furthermore, the resulting polygon has got $n - m$ vertices. Hence, given k reflex vertices and k chords, the resulting polygon is computed in $O(n)$: each vertex is visited a constant number of times.

Furthermore, the area of the removed parts can be computed without changing the computational cost. Indeed, let p be a point in the plane, the area of a polygon P is given by

$$\mathcal{A}(P) = \frac{1}{2} \sum_{i=0}^{n-1} A(p, v_i, v_{i+1 \ (mod \ n)}) \,, \tag{2}$$

where $A(p, v_i, v_j)$ denotes the signed area of the triangle (p, v_i, v_j) [9]. Hence, the area can be computed in a greedy process during the vertex removal process.

3.3 Overall computational analysis

Based on the previous analyses, we can detail the computational cost of the global algorithm presented in the section 3.1. First of all, step 1 (kernel determination) requires $O(n)$ computation using [10]. Then, we have presented a simple $O(n)$ algorithm to compute the distance-to-kernel of reflex vertices and thus the cost of step 2 (reflex vertices sorting) is $O(n + k \cdot \log k)$. The Step 3 and the step 5 in the **while** loop on reflex vertices requires $O(1)$ computations.

Given the two reflex points r_1 and r_2 of step 6, we have to decide which chord should be associated to r_1. We have two extremal chords, a single-pivot chord whose slope is computed in $O(1)$ and a double-pivot chord going through r_2. Using our heuristics, we choose the chord that minimizes the removed part of P. Hence, we compare the removed part area using the double chord and the 9 area measures of the 3×3 other possible choices for r_1 and r_2. Hence, at each step of the **while** loop, we compute 10 polygon cuts and we choose, for the r_1 chord, the chord that maximizes the resulting polygon area. Note that steps 7 and 8 are computed during the polygon cutting steps and do not change the computational cost.

In the worst case, each polygon cut step is done in $O(n)$. Hence, the overall complexity of the **while** loop is $O(k \cdot n)$. Finally, the cost of the approximated MACS extraction algorithm is $O(k \cdot n)$ in the worst case. Let N be the number of removed vertices while evaluating the optimal chord at r_1 in step 6. If we suppose that the reflex vertices of P are *uniformly distributed* in the sense that each other tested chord only visits the same amount $O(N)$ of vertices. Then, at each step, $O(N)$ vertices are visited and the modified polygon has got $O(n - N)$ vertices. Hence, the **while** loop complexity becomes $O(n)$. This leads to a global cost for the approximated MACS extraction in $O(n + k \cdot \log k)$. In practice, such *uniformity* has been observed in our examples.

To speed up the algorithm, several optimizations can be done without changing the worst case complexity. For example, when chords going through r_2 are tested, the obtained polygons are propagated to the next step of the while loop in order to reduce the number of polygon cut steps. Furthermore, since the area of the removed parts is computed during the vertex scan, the process can be stopped if the area is greater than the current minimum area already computed with other chords.

3.4 Experiments

In this section, we present some results of the proposed algorithm. First of all, the figure 9 compares the results between the optimal Chang and Yap's algorithm [4] and the approximated MACS extraction process. In practical experiments, the optimal $O(n^7)$ algorithm do not lead to a direct implementation. Indeed, many complex geometrical concepts are used and the overall algorithm is not really tractable. Hence, we use a doubly-exponential process to extract the optimal MACS. The main drawback of this implementation is that we cannot extract the optimal MACS if the number of the polygon points is important. In figure 9, the first column presents the polygon, its kernel and the distance labelling of all vertices, the second row contains the optimal MACS and the third one the approximated MACS. Note that the results of the last row are identical.

If we compute the area error between the optimal and the approximated MACS on these examples, the error is less than one percent.

The figure 10 illustrates the intermediate steps of the approximated MACS algorithm and the figure 11 presents the result of the proposed algorithm on different shapes.

4 Conclusion

In this article, we have proposed a fast algorithm to extract an approximation of the maximum convex subset of a star-shaped polygon. Based on an analysis of the optimal solution detailed by Chang and Yap [4], we have defined a kernel dilatation based heuristic that use

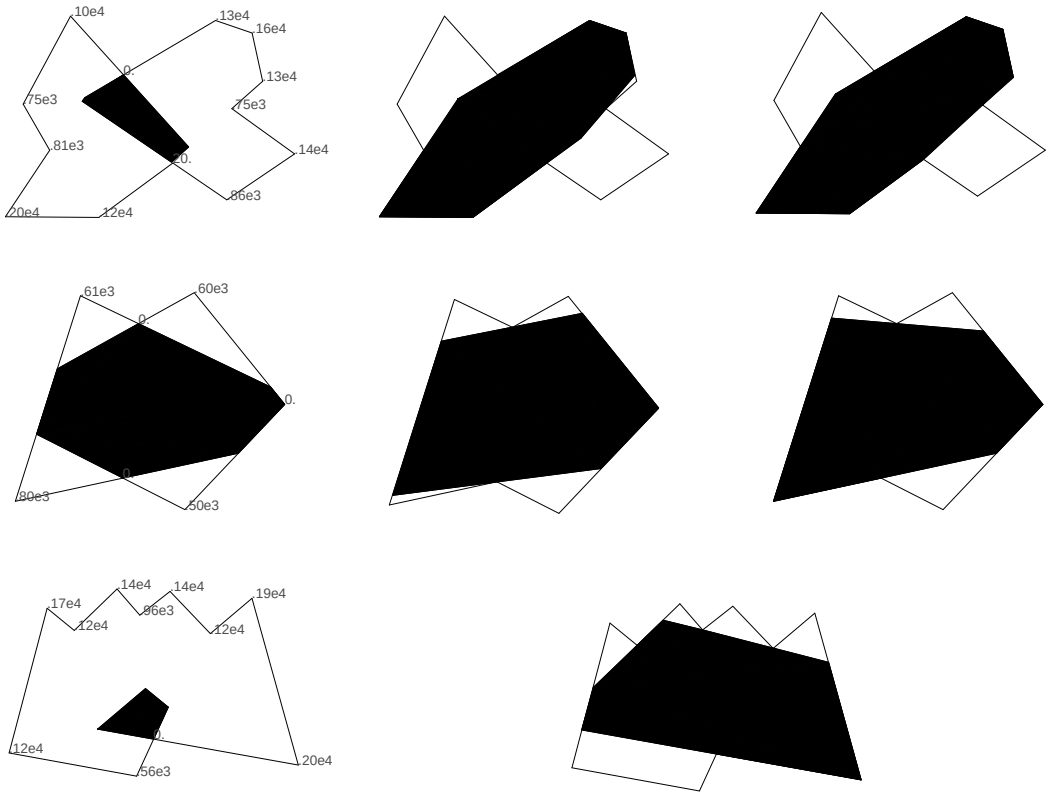

Fig. 9. Comparisons between the optimal MACS and the proposed algorithm: the first column presents the input polygons, their kernels and the distance labelling, the second column shows the results of the Chang and Yap's algorithm. The last row present the result of the proposed algorithm.

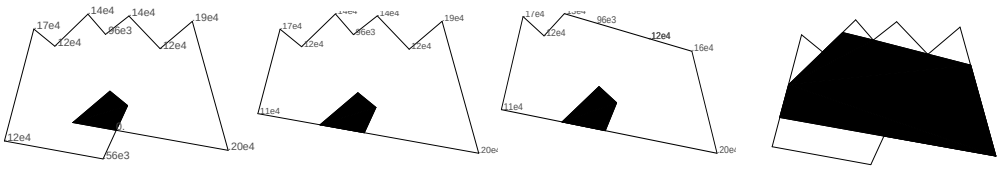

Fig. 10. Some intermediate steps of the approximated MACS algorithm.

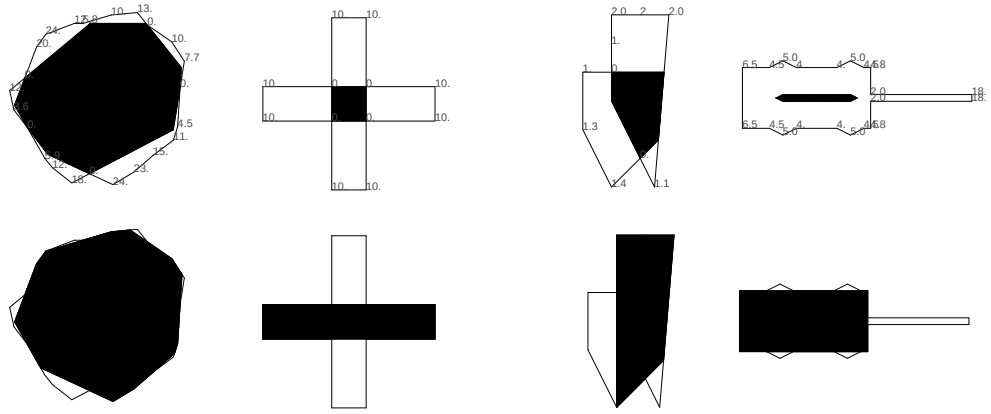

Fig. 11. Results of the proposed approximated MACS algorithm on various shapes. The first row illustrates the shapes, their kernels and the distance labelling, the second row presents the obtained solutions.

classical tools in computational geometry. The computational worst case cost of the algorithm is $O(k \cdot n)$ where n is the number of points of the polygon and k is the number of reflex vertices. However, under some hypotheses on the reflex vertex distribution, the complexity can be bounded by $O(n + k \cdot \log k)$. In our experiments, the computational behavior of the algorithm is closer to the second complexity bound than to the first one.

In future works, a first task consists in a formal comparison between the proposed approximated solution and the optimal one. However, heuristics choices make this comparison non-trivial. More generally, an optimization of the Chang and Yap's optimal algorithm is still an open problem. However, efforts should be made to extend this heuristic to general polygons.

References

1. A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor. A linear time algorithm for computing the Voronoi diagram of a convex polygon. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 39–45, New York City, 25–27 May 1987.
2. C. Andjar, C. Saona-Vzquez, and I. Navazo. LOD visibility culling and occluder synthesis. *Computer Aided Design*, 32(13):773–783, November 2000.

3. H. Blum. A transformation for extracting descriptors of shape. In *Models for the Perception of Speech and Visual Forms*, pages 362–380. MIT Press, 1967.
4. J. S. Chang and C. K. Yap. A polynomial solution for the potato-peeling problem. *Discrete & Computational Geometry*, 1:155–182, 1986.
5. F. Chin, J. Snoeyink, and C. A. Wang. Finding the medial axis of a simple polygon in linear time. In *Proc. 6th Ann. Int. Symp. Algorithms and Computation (ISAAC 95)*, number 1004 in Lecture Notes in Computer Science, pages 382–391. Springer, 1995.
6. K. Daniels, V. Milenkovic, and D. yRoth. Finding the largest area axis-parallel rectangle in a polygon. *Computational Geometry: Theory and Applications*, 7:125–148, 1997.
7. H. Edelsbrunner. Computing the extreme distances between two convex polygons. *Journal of Algorithms*, 6:213–224, 1985.
8. J. E. Goodman. On the largest convex polygon contained in a non-convex $n-$gon or how to peel a potato. *Geometricae Dedicata*, 11:99–106, 1981.
9. Joseph O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1993. ISBN 0-521-44034-3.
10. F. P. Preparata and M. I. Shamos. *Computational Geometry : An Introduction*. Springer-Verlag, 1985.