Electronic Notes in Theoretical Computer Science 46 (2001)
URL: http://www.elsevier.nl/locate/entcs/volume46.html 16 pages

An Elementary Algorithm for Digital Arc Segmentation

David Coeurjolly and Laure Tougne [1]

Laboratoire ERIC
Université Lumière Lyon 2
5 av. Pierre Mendès-France
69676 Bron, France

Yan Gérard and Jean-Pierre Reveillès [2]

Laboratoire LLAIC1
IUT Clermont 1
Université d'Auvergne Clermont 1
B.P. 86
63172 Aubière
France

Abstract

This paper is concerned with the *digital circle recognition problem* and more precisely with the *circular separating algorithm*. It tries to go further in implementation details, giving pseudo-code algorithms for the main points, and avoids using the sophisticated machinery coming either from Computational Geometry or from Linear Programming found in previous papers on this subject. After recalling the geometrical meaning of the separating circle problem, we present an incremental algorithm to segment a discrete curve into digital arc.

Keywords: digital arc recognition, arc separability, digital circle, digital curvature.

1 Introduction

Euclidean shape recognition is an important topic of Discrete Geometry. Many works have been done for straight lines [12,5], planes [4,17] and algorithms become more and more efficient. Some solutions exist for higher order objects such as conics [14] or general polynoms [8] but a lot of further developments remains to be done.

[1] Email: {dcoeurjo,ltougne}@univ-lyon2.fr
[2] Email: {reveil,gerard}@llaic.u-clermont1.fr

This paper is concerned with the *digital circle recognition problem* and more precisely with the *circular separating algorithm*. It tries to go further in implementation details, giving pseudo-code algorithms for the main points, and avoids using the sophisticated machinery coming either from Computational Goemetry or from Linear Programming found in previous papers on this subject.

After recalling the geometrical meaning of the separating circle problem, we present an elementary algorithm based on duality, the formal approach to Hough transform. This algorithm is then applied to find a partition of any 8-connected curve in digital circular arcs. Such an algorithm allows us to define and compute the local curvature to a digital curve.

2 The Separating Arc Problem

Let S and T be two finite sets of points of $\mathbb{Z}^2$, we say that they are *circularly separable* if there exists an Euclidean circle $C(\omega, R)$ centered at ω and with radius R, such that:

$$\forall s \in S,\ \forall t \in T, \text{ we have } s \in C(\omega, R) \text{ and } t \notin C(\omega, R)$$

If such a circle exists for given S and T, its center ω necessarily satisfies the following inequalities:

$$\forall s \in S,\ \forall t \in T,\ dist(\omega, s) < dist(\omega, t)$$

where $dist(a, b)$ denotes usual Euclidean distance between points a and b.

This means that the set $acd(S, T)$ (*i.e.* arc center domain) of centers of circles separating sets S and T is equal to the intersection of half planes defined by perpendicular bisectors of points s and t containing points of S. If for all s in S and for all t in T, $H(s, t)$ denotes the half-plane bounded by bisector of s and t and containing s, this convex set is thus defined by

$$acd(S, T) = \bigcap_{s \in S, t \in T} H(s, t)$$

Such a convex set is also called generalized Voronoi cell. The figure 1 shows an example of such a set $acd(S, T)$ where the points of S are white and the points of T are black.

Various algorithms exist for the construction of set $acd(S, T)$ [7] [11] [2] [3], but we shall present a very elementary one using *duality*, (the theoretical content of the *Hough transform*), which could reveal useful for the Imagery applications which require the lightest possible implementations.

2.1 Points-lines duality

In order to simplify our use of duality let us suppose that the rightmost point R of convex set $acd(S, T)$ is unique, and the same for its leftmost point L.

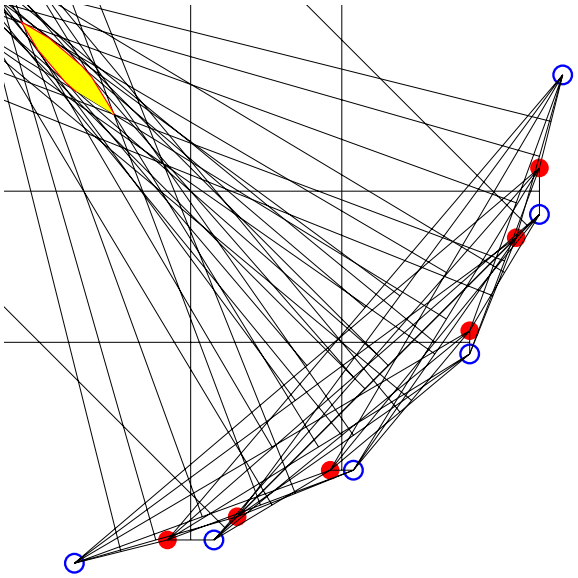

Fig. 1. Set $acd(S,T)$ of centers of separating circles.

Consequently, this set contains no vertical edge; such a restriction can be simply disposed of by intersecting convex set $acd(S,T)$ with the eventual vertical lines as a last step of the algorithm.

The lower part of the positively oriented boundary of $acd(S,T)$ is a polygonal curve L^-, between points R and L, and, similarly its upper part is the line L^+ starting at L and ending at R.

Let us denote by $v_i^-, i = 0, \ldots, m$ the vertices of L^-, where $v_0^- = R$ and $v_m^- = L$ and similarly by $v_i^+, i = 0, \ldots, n$, the vertices of L^+ where, this time, $v_0^+ = L$ and $v_m^+ = R$. In the same way edges of L^- are denoted by $e_i^-, i = 1, \ldots, m$ and those of L^+ are denoted by $e_i^+, i = 1, \ldots, n$.

Our uniqueness hypothesis about west and east vertices of $acd(S,T)$ implies that all equations of the lines defined by edges of this set can be written as

$$y = a_i x + b_i$$

Due to the symmetry between upper and lower parts of $acd(S,T)$, we shall restrict our study to lines of the upper part L^+ and leave the treatment of L^- to the reader.

We now observe that polygonal curve L^+ is contained in the lower envelope of lines associated to edges e_i^+. If, moreover, equation of the line bounding half-plane $H(s,t), s \in S, t \in T$ is written $ax + by + c = 0$, giving its normal vector (a,b) the orientation of vector $\overrightarrow{st}$, implies that the half-plane

$$H(s,t) = \{M | \overrightarrow{st}.\overrightarrow{\mu M} \leq 0\}$$

where μ is the midpoint of segment st, may also be defined by inequality

$$H(s,t) = \{(x,y)|ax + by + c \leq 0\}.$$

Thus introducing the set $Ineqs$ of all inequalities $ax+by+c \leq 0$ associated with all cords $\overrightarrow{st}$, s in S,t in T, we see that upper part L^+ of $acd(S,T)$ is also defined by the subset of $Ineqs$ where $b > 0$ (once again $b = 0$ is not allowed because vertical lines are omitted); this subset is denoted $Ineqs^+$. Lines of $Ineqs^+$ having equations $ax+by+c = 0$ where $b > 0$, may be termed as *upward oriented*. Of course set $Ineqs^+$ contains the set of lines which support edges of L^+, both of them having the same lower envelope. Lines of the larger family $Ineqs^+$, are easily selected among bisectors of cords $\overrightarrow{st}$, the main problem to determine L^+ is to find those which appear in their lower envelope.

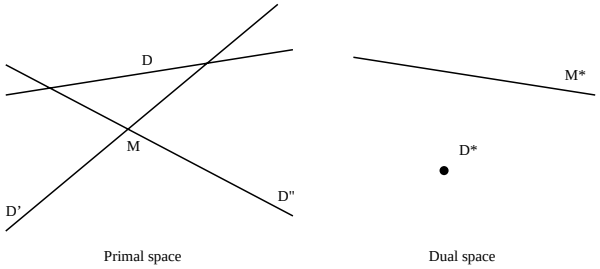

Fig. 2. Illustration of prop. 2.1.

Equations of lines of $Ineqs^+$ may be written $y = ax + b$, with convenient a and b, which allows writing inequalities of $Ineqs^+$ as $y \leq ax + b$.

Let us recall that duality is the geometrical process which maps line whose equation is $y = ax + b$ to the point $(a, -b)$ and the point (u, v) to the line $y = ux - v$. It is convenient to denote by D^* the dual of line D and by P^* the line dual of point P. The figure 2 shows an example of dual elements. It is easily proven that

Proposition 2.1 *Point $M = (u, v)$ of primal space belongs to half-plane $ax + by + c <= 0$, where $b > 0$, if and only if point $(-a/b, c/b)$ lies beneath line M^* of dual space whose equation is $y = ux - v$.*

Both conditions are in fact equivalent to inequality $au + bv + c \leq 0$.

From this it can be immediately deduced that if D, D', D" are three lines such that intersection point $M = D' \cap D$" has coordinates $M = (a, b)$, then lower envelope of D, D', D" takes value b if $x = a$ if and only if point D^* is located below the line M^* dual of point M, (see Fig. 2).

Following result is an immediate consequence of this observation.

Proposition 2.2 *Let $\mathcal{F}$ be a set of lines (no one of which is parallel with Oy axis) and $\mathcal{F}^*$ the set of points which are the dual transforms of the lines of $\mathcal{F}$. Then duality maps lines of the lower envelope of $\mathcal{F}$ in a one to one way with the vertices of the upper part of the convex hull of $\mathcal{F}^*$.*

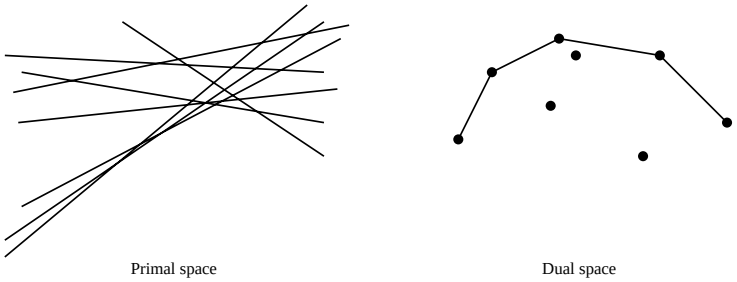

Fig. 3. Lower envelope of a family of lines corresponds to the vertices of the upper part of the convex hull of their dual transforms.

Using duality the lower envelope of a family of lines can be obtained by constructing the convex hull of their transforms. Moreover duality transform of the edges of this hull are the intersection points of the envelope lines.

Corollary 2.1 *The set $acd(S,T)$ can be obtained in*

$$card(S).card(T).log(card(S).card(T))$$

time using only elementary primitives like points-lines duality and 2D convex hull.

From prop. 2.1. it follows that construction of the upper boundary L^+ of $acd(S,T)$ is clearly obtained in $S.T.log(ST)$ time; the same can be said for its lower boundary L^-. Intersection of these polygonal curves takes at most $O(S.T)$ time; the result follows.

We can present the basic algorithm:

Algorithm 1: An elementary separating arc algorithm

Input: Two non-empty sets S and T
Output: $acd(S,T)$

Let $Ineqs$ be the set of half planes $H(s,t)$ with $\forall s \in S$ and $\forall t \in T$
$Ineqs^+ = \{ax + by + c \leq 0 \in Ineqs \quad / \quad b < 0\}$
$Ineqs^- = \{ax + by + c \leq 0 \in Ineqs \quad / \quad b > 0\}$
$C^+ = \mathsf{convexhull}(dual(Ineqs^+))$
$C^- = \mathsf{convexhull}(dual(Ineqs^-))$
Extract L^+ as the lower part of C^+
Extract L^- as the upper part of C^-
Compute the intersection between L^+ and L^-
Return the obtained polygon $acd(S,T)$

2.2 Improvements of the separating circle algorithms

Several observations are valuable if the former algorithm has to be effectively implemented.

2.2.1 Reducing S

If a circle separating sets S and T exists, this circle being convex contains convex hull of S. Thus replacing set S by the set of vertices of its convex hull does not change the solution of the circular separation problem. Cost of this processing being $O(S.log(S))$ it does not increase complexity of the overall algorithm.

2.2.2 Reducing T

In the same way let us consider two points t, t' of set T and let us suppose that the convex hull of $\{S \cup t\}$ contains t', then if there exists a circle containing S which does not contain t' then this circle does not contain t.

This leads to process T points as follows:

adjoint a point $t \in T$ to the convex hull of S and remove t out of T if $hull(S, t)$ contains another point of T. A simple $O(Tlog(T))$ algorithm can be devised to make this reduction. An even more elaborate process, presented in 3.1, will associate a set T satisfying this property to the given set S.

2.2.3 Relation with Voronoï diagrams

This point shows a very simple relation with the previous more sophisticated approach of the separating circle problem [7]. Let us suppose that set T is given and let us introduce points of S one by one; let s be such a point. In this case set $acd(\{s\}, T)$ is obviously the Voronoï cell of site s in the Voronoï diagram of $\{s\} \cup T$; let us denote by $vor(s, T)$ this convex polygon. We deduce from this remark that in the general case $acd(S, T)$ is the intersection of all $vor(s, T)$ when s takes all values of S, that is

$$acd(S, T) = \bigcap_{s \in S} vor(s, T)$$

Previous remark leads to another interesting geometric observation. Of course S and T are disjoint sets and a point s of S cannot lie on a line defined by two distinct points of T thus points of S belong to the open interior of the convex hull of T or are strictly exterior to that polygon. This shows, obviously, that the convex polygon $vor(s, T), s \in S$ is bounded *if and only if* s is an interior point of the convex hull of T. This partition of points of S into two families, those having a bounded Voronoï cell $vor(s, T)$ and the other ones, leads again to interesting coding simplification and improvement.

Let us now apply the previous results on discrete curves. The following section describes an arithmetical approach to compute the minimum number of necessary elements of S and T in order to separate the points that belong to the discrete curve (corresponding to S) from the other ones (corresponding to T).

3 An Arithmetical Approach

3.1 Input description and Algorithms

We shall now present an incremental algorithm to cut a 8-connected discrete curve $\mathcal{C}$ into pieces of circular digital arcs. Let us briefly describe this algorithm before we present the arithmetic properties of $\mathbb{Z}^2$ which will lead to crucial improvements on the complexity of this algorithm.

A first treatment, made with Debled's linear polygonalization algorithm, divides $\mathcal{C}$ into digital line segments which are then gathered into strictly convex or concave (SCoC) parts denoted $\{\mathcal{C}_i\}_{1,\ldots,k}$. This preprocess can be done in time $O(n)$ where n denotes the number of pixels of $\mathcal{C}$.

Then we compute the classical convex hull of each $\mathcal{C}_i$. As a classical discretization of ordinary convex continuous curves satisfies the Weakly Externally Visible property [15] these convex hulls, denoted $\{\Gamma_i\}_{1,\ldots,k}$ can be obtained in linear time, for example by the Three Coins algorithm [9,13].

Without reducing the problem we are thus led to consider 8-connected curves which are the discretization of convex polygonal lines. These are cut into one or several pieces of digital circular arcs (or *quasi-circular* arcs for short). Let Γ be one such convex polygonal curve; its partition into quasi-circular arcs is done defining suitable sets S, T of interior and exterior points associated to Γ. Sets S, T are initialized so that $acd(S,T) \neq \emptyset$ and are increased until this separating domain becomes empty, where a new circular arc is started.

Let $v_1, v_2, \ldots, v_r$ denote $\Gamma's$ positively ordered vertices and f_i denotes the linear functional induced by successive vertices v_i, v_{i+1} such that $\Gamma's$ other vertices satisfy $f_i(v) \leq 0$. Integer points x which satisfy one of the inequalities $f_1(x) > 0, f_2(x) > 0, \ldots, f_{n-1}(x) > 0$ are said to be exterior to Γ; interior integer points of Γ are defined as usual. Remark that functional f_n defined by v_n, v_1, is missing because points satisfying $f_n(x) > 0$ do not need to be considered. Of course S should be composed of interior points of Γ and T of its exterior points but, if it is obvious that S can be reduced to Γ vertices (cf. 2.2.1), reducing T efficiently is a little more subtle than what is indicated in 2.2.2 and needs considering special points lying close to any integer vector.

Let us denote by $\overrightarrow{u} = (a,b)^T$ a directional vector of a given f_i such that $gcd(a,b) = 1$. Let us consider the vector $\overrightarrow{v}$ be a unimodular vector to $\overrightarrow{u}$ (*i.e.* $det(\overrightarrow{u}, \overrightarrow{v}) = \pm 1$).

Definition 1 *A point of integer coordinates (x,y) is a weakly exterior point (or a Bezout's point) of the discrete line defined by $\overrightarrow{u}$ if and only if:*

$$\begin{pmatrix} x \\ y \end{pmatrix} = \overrightarrow{v} + k\overrightarrow{u} \quad with \quad k \in \mathbb{Z} \quad and \quad det(\overrightarrow{u}, \overrightarrow{v}) = \pm 1$$

and (x,y) is exterior to Γ.

As we constraint the separating arc to go through the extremities, we can remark that it is sufficient to be sure that the weakly exterior point nearest from the middle of the edge is over the separating arc to conclude that all the others are (see Figure 4). Consequently, we reduce the number of constraints to one, whatever the length of the segment is. Thus, for a given edge with extremities v_i and v_{i+1}, and the excluded exterior point E, the convex set solution is defined by $acd(\{v_i, v_{i+1}\}, \{E\})$. The figure 5 shows an example of piece of discrete curve, its associated convex hull and weakly exterior point.

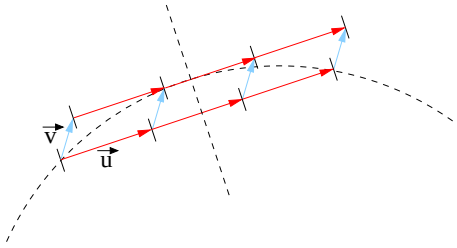

Fig. 4. Illustration of the choice of the nearest exterior point from the middle of an edge. All other points are also excluded.

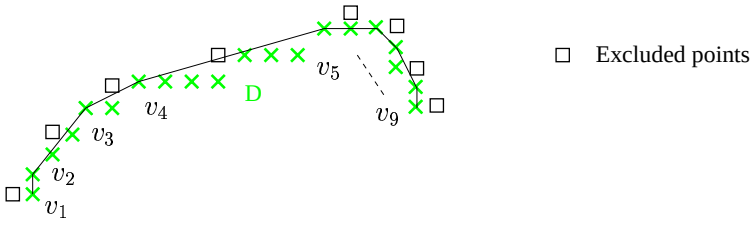

Fig. 5. An example of piece of discrete curve and its associated convex hull and the weakly exterior points

In the following parts, we first present a elementary algorithm to construct the arc center domain associated to a SCoC part and then we present an incremental algorithm to construct the maximum domains of a general discrete curve.

3.1.1 Strictly Convex or Concave (SCoC) part case

Let us first consider a sequence of edges of a SCoC $\mathcal{C}_i$ (Figure 5 shows an example of a SCoC). We consider all the edges at the same time. In order to construct the convex set associated to them, we have to consider the union of the constraints generated by each one. The algorithm is the following one:

Algorithm 2: A global separating arc algorithm on a SCoC curve

Input: a sequence of edges of Γ.
Output: the convex set $acd(S, T)$ (maybe empty)

$S = \emptyset$

$T = \emptyset$
For each edge of the exterior part of the convex hull of Γ
  Let (a, b, v_i, v_{i+1}) denotes the parameters of such a segment
    Let compute E the weakly exterior point
    $S := S + \{v_i, v_{i+1}\}$
    $T := T + \{E\}$
end for
Compute $acd(S, T)$ using the algorithm 1
Return $acd(S, T)$

If we denote by N the number of edge of the convex hull, we have to consider :

- $(N + 1)$ extremity points,
- N weakly exterior points.

Consequently, we are leaded to consider the respective sets of points S and T such that $card(S) = N + 1$ and $card(T) = N$. We finally obtain $O(N^2)$ constraints.

3.1.2 Incremental algorithm

In this case, we apply the Debled's algorithm to split the curve into SCoC parts. On each such parts, we incrementally consider the edges of the convex hull taking them one by one in order to compute the maximum arc center domain. When we go from SCoC piece to another, or when the next edge will lead to an empty domain, we store the current convex set as the maximum arc center domain and we continue with new one. We have the algorithm:

Algorithm 3: An incremental separating arc algorithm on a discrete curve

Input: A discrete curve $\mathcal{C}$
Output: An array of all the convex polygons

$i := 0$
For each $\mathcal{C}_i$ part given by the polygonalisation algorithm on $\mathcal{C}$
  $S := \emptyset$
  $T := \emptyset$
  Γ_i:=convexhull(C_i)
  For each exterior edges $v_j v_{j+1}$ of Γ_i
    Let (a, b, v_j, v_{j+1}) denotes the parameters of such a segment
    Let compute E the weakly exterior point
    $S := S + \{v_j, v_{j+1}\}$
    $T := T + \{E\}$
    Compute $new_{acd} := acd(S, T)$ using algorithm 1
    If $new_{acd} = \emptyset$ **then** we define a new domain
      $S := \emptyset$; $T := \emptyset$
      $acd[i] := prev_{acd}$
      $i := i + 1$
    else
      $prev_{acd} := new_{acd}$

end for
$acd[i] := prev_{acd}$
i:=i+1;
end for
Return *acd*

Just remark that in this case, *a priori*, for each generated new segment we have to consider all the intersections between the half planes of the current segment and all the previous ones. As we have $O(N^2)$ constraints at each of the N steps, we obtain $O(N^3)$ constraints.

Using the part 2, let us do a complexity analysis of the previous algorithms.

3.2 Complexity analysis

If we remind the previous notations, n denotes the number of pixels of the discrete curve $\mathcal{C}$ and N the number of edges of the convex hull of the curve. First, we use the well-known property:

Proposition 3.1 *On a convex discrete curve, the number of edges of the convex hull is bounded by* $O(\sqrt{n}log(n))$.

Using this proposition, we can give the complexity of previous algorithms.

Global algorithm complexity

Since we have shown that the number of constraints is in $O(N^2)$, this leads to a linear number of constraints, with respect to n. In the following, we only consider SCoC discrete curve, indeed if the curve has got inflection points, the global separating arc convex is empty. This hypothesis can be checked in $O(n)$ as a preprocessing. Hence, we can apply the algorithm 1 to compute the vertices of the convex set, maybe empty, in $O(nlog^3(n))$.

Incremental algorithm complexity

We have the following proposition:

Proposition 3.2 *Let n denote the number of pixels of a discrete curve, the complexity of the incremental algorithm (Alg. 3) is* $O(nlog^3(n))$.

Proof: In this algorithm, since we compute the convex set for each new inserted edge, a basic complexity analysis leads to n (for a general discrete curve, the number of edges is in $O(n)$) tests on $O(N^2)$ linear constraints and thus to a complexity in $O(n^2log^3(n))$. However, we can transform the algorithm 1 into an incremental one. Furthermore, we only use classical computational geometric tools on which a lot of literature can be found [3]. So, we can reuse an incremental convex hull algorithm on which the update cost is in $O(log(n))$. Hence, if we have computed a convex set $acd(S,T)$ and if we insert a new constraint, the updates of convex hulls C^+ and C^- can be done in $O(log(card(S)card(T)))$. All other processes of algorithm 1 can

be easily designed in an incremental form keeping the update complexity in $O(log(card(S)card(T)))$. Hence, each time we test a new edge, we update the polygon $acd(S,T)$ in $log(n)$. This leads to a global complexity in $O(nlog^3(n))$.

Note that the bound $N = O(\sqrt{n}log(n))$ is a pessimistic one, we expect a bound in $O(\sqrt{n})$ that will lead to a complexity in $O(nlog(n))$ for both algorithms.

Let us notice the important reversibility property.

3.3 Reversibility

Let us consider a piece of the discrete curve $\mathcal{C}$, denoted by P, such that the corresponding convex set is not empty. This convex set characterizes all the arcs that separate P from the other points of the plane. These arcs have the following property:

Proposition 3.3 *Let $acd(S,T)$ be the arc center domain defined by a piece of a discrete curve $\mathcal{C}$ and let p be a point in $acd(S,T)$. For each Euclidean arc $\mathcal{A}$ of center p and radius $r \in [min_{X\in P}dist(c,X), min_{Y\notin P}dist(c,Y)[$, the OBQ (Object Boundary Digitization) digitization [10] of $\mathcal{A}$ is exactly the considered piece of $\mathcal{C}$.*

This property is proved by construction of the domain. To be more precise and since a lot of digitization processes can be found, we consider here a *inner* digitization, that is the set of discrete points closest to the center p.

We can use the previous algorithms to compute the curvature at each point of a discrete curve.

3.4 Application to curvature estimation

Let us suppose our goal to be the curvature estimation at each point of the discrete curve $\mathcal{C}$. A classical way to define the curvature at a point of a curve is to consider the inverse of the osculating circle radius [1], this computation can be done considering the best fitting circle locally at each point of the curve. In the previous sections, we have shown an algorithm to recognize digital arc pieces from a discrete curve based on convex hull vertices and edges. For the curvature estimation problem, we use the same tools but with specific characteristic points:

Definition 2 *Let M be a point of $\mathcal{C}$, we define $k(M)$ the curvature at M by:*

$$k(M) = sign\left(\frac{1}{min(\{dist(c,M)) \mid c \in acd(\{M,L,R\},\{L_{ext},R_{ext}\})}\right)$$

Where L, R, L_{ext}, R_{ext} and sign are defined by:

- *$[ML]$ and $[MR]$ are discrete straight lines*

- *L_{ext} the weakly exterior point associated to the segment $[ML]$ exterior to the polyline (LMR)*
- *R_{ext} the weakly exterior point associated to the segment $[MR]$ exterior to the polyline (LMR)*
- *L and R are maximal for the acd problem, it means that L (resp. R) is the farthest point on $\mathcal{C}$ from M such that the arc center domain $acd(\{M, L, R\}, \{L_{ext}, R_{ext}\})$ is not empty.*
- *sign is +1 or -1 according to the convexity or concavity of the polyline (LMR)*

This curvature is the inverse radius associated with the closest vertex of $acd(\{M, L, R\}, \{L_{ext}, R_{ext}\})$ to M, that defines the minimum curvature radius and thus the maximum curvature at M. Since M, L and R may not be convex hull points, some points of $[MR]$ or $[ML]$ might be outside the computed separating arcs. This makes the process not reversible but this is not important in the case of local curvature computation.

We have an elementary algorithm to estimate the curvature at a point of a curve.

Algorithm 4: Curvature estimation at a point of a discrete curve

Input: A discrete curve $\mathcal{C}$ and a point $M \in \mathcal{C}$
Output: the estimated curvature of M

Note : the discrete curve is stored as a double-link list

L:=M→prev
R:=M→next
Compute L_{ext} and R_{ext}
While $(acd(\{M, L, R\}, \{L_{ext}, R_{ext}\}) \neq \emptyset)$ AND ($[ML]$ is a discrete straight line) AND ($[MR]$ is a discrete straight line) **do**
    L:=L→prev
    R:=R→next
    Compute L_{ext} and R_{ext}
end while
If $acd = \emptyset$ **then** we come back to a non empty acd
    L:=L→next
    R:=R→prev
    Compute L_{ext} and R_{ext}
Let $\mathcal{A}$ be the arc defined by the closest center of the $acd(\{M, L, R\}, \{L_{ext}, R_{ext}\})$ to M
Return the inverse radius of $\mathcal{A}$

Since the *acd* computation is only based on five points the first test of the while loop can be done in a constant time but the straight line recognition can be in $O(n)$ in a basic complexity analysis. In order to estimate the curvature at each point of the curve, we must apply the above algorithm to each point of the curve:

Algorithm 5: Curvature estimation at each point of a discrete curve

Input: A discrete curve $\mathcal{C}$
Output: the estimated curvature graph K

For each point M of the curve
  Let store in $K[M]$ the curvature estimation using *Alg. 4* at M
end for
Return K

A basic complexity analysis of *Alg. 5* will lead to an $O(n^2)$ complexity. However, we can use discrete tangents defined by [16] in order to bound the discrete segment growth. As a matter of fact these discrete tangents are defined by the longest discrete segment at a point of $\mathcal{C}$ and thus half-tangent at M define the longest possible segment for $[ML]$ and $[MR]$. Since these discrete tangents can be computed in $O(n)$ at each point of the curve $\mathcal{C}$ [6] we can bound the $[ML]$ and $[MR]$ in the same complexity.

If the extreme L and R lead to an empty *acd*, we have to backtrack on the segments using a dichotomy process in order to find appropriated L and R. Each tested points in the recursive process will take $O(log(n))$ to check if the *acd* is empty or not (*i.e* $O(log(n))$ to compute the new equation of the segment and $O(1)$ to test the *acd*). We finally obtain a global complexity in $O(nlog(n)^2)$.

The figure 6 represents an example of the *acd* obtained from the small points in black (T) and white (S). Notice that the tangents come from the circle of center $(170, 30)$ and of radius (25) and, the estimated center using the above algorithm is well located.

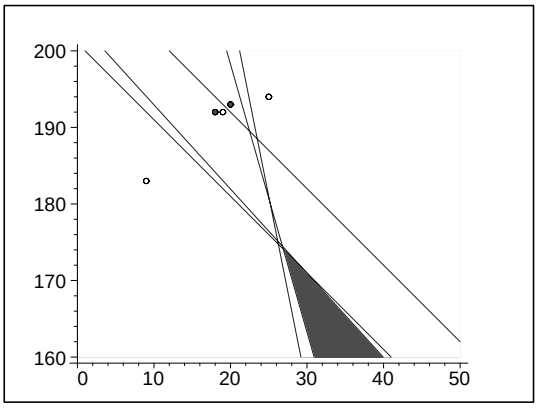

Fig. 6. An example convex set obtained from two half-tangents computed on a discrete circle of center (170,30).

As in [1], some complementary experiments would be necessary to compare this new curvature estimation algorithm to previous ones.

Let us now present some experiments we have made in order to test the accuracy of the global approach.

4 Some experiments

The first experiment concerns digital circle center recovering; a digital circle C being given we try to find a point M of $\mathbb{R}^2$ (and a real number R) such that the discretization of the circle of center M and radius R gives C back.To obtain this, we compute its convex hull and our goal is to test the number of edges that are necessary to find a *good* approximation of the radius of the circle. In figures 7, 8 and in table 1, we show the results obtained for a circle of radius 100. We extract from the shape of the convex set the mid point and the standard deviation of its vertices. We can remark that the mid point fastly converges to the real center and the standard deviation also decreases very quickly.

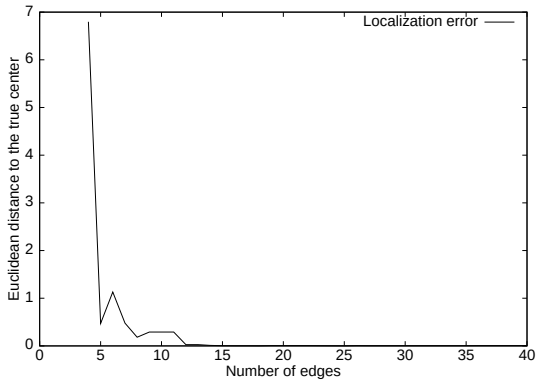

Fig. 7. Euclidean distance between the center of a discrete circle of radius 100 and the mid point of $acd(S,T)$.

# of edges	$\bar{x}$	$\bar{y}$	σ_x	σ_y
5	0.338444	.328996	2.772722	17.091599
20	0.005424	0.001096	0.000383	6.653609e-05
30	0.004237	0.000741	1.511038e-05	7.513850e-06
40	0.005606	-0.000490	1.710540e-05	1.949764e-06

Table 1
Some numerical results on the mid point localization and the standard deviation considering the convex set vertices on a circle of radius 100.

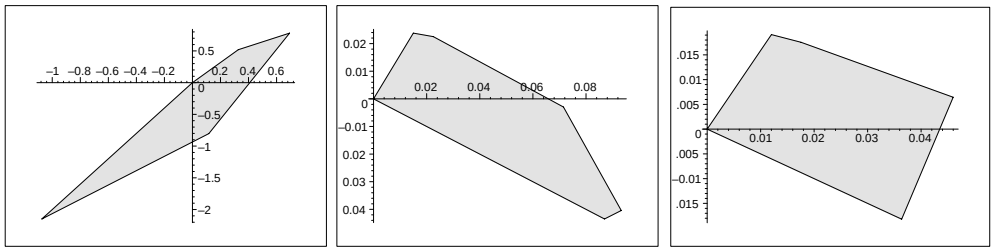

Fig. 8. Some examples of the convex set evolution when we increase the number of edges on the circle of radius 100.

5 Conclusion and future works

Purposed approach uses elementary geometrical notions only : finding circles separating two sets of points and one obvious property of $\mathbb{Z}^2$ geometry : Bezout's points. This allows conceiving of various simple algorithms leading to efficient programs. If query for a single separating circle is the need, first part of our work is enough; if segmentation of a digital curve into circular arcs is wanted, the knowledge of all arc centers domains is necessary to recognize points where it becomes empty, then our incremental version of the second part can be the tool. Let us insist, at this point, on the quite reasonable complexity increase beyond linearity of this incremental algorithm; linearity is already reached by the query of only one separating circle when, here, all these circles are computed.

Many trails seem interesting to follow after this work such as the study of variations of the local digital curvature of the discretization of algebraic curves, comparison of our approach with those using higher order derivatives in the definition of discrete curvature, or the recognition of higher differential invariants.

Beside these most urgent tasks we are interested as much by theoretical developments like higher dimensional generalizations as by the practical sides like optimizing algorithms and resulting codes.

References

[1] Coeujolly, D., S. Miguet and L. Tougne, *Discrete curvature based on osculating circle estimation*, in: *International Worshop in Visual Form IV*, 2001.

[2] Damaschke, P., *The linear time recognition of digital arcs*, Pattern Recognition Letters (1995), pp. 543–548.

[3] de Berg, M., M. van Kreveld, M. Overmars and O. Schwarzkopf, "Computational Geometry," Springer-Verlag, 2000.

[4] Debled, I., "Etude et reconnaissance des droites et plans discrets," Ph.D. thesis, Université Louis Pasteur - Strasbourg (1995).

[5] Debled-Rennesson, I. and J.-P. Reveillès, *A linear algorithm for segmentation of digital curves*, Parallel Image Analysis: Theory and Applications (1993), pp. 73–100.

[6] Feschet, F. and L. Tougne, *Optimal time computation of the tangent of a discrete curve : Application to the curvature*, in: *Discrete Geometry for Computer Imagery*, 1999.

[7] Fisk, S., *Separating points sets by circles, and the recognition of digital disks*, IEEE Transaction on Pattern Analysis and Machine Intelligence **8** (1986), pp. 554–556.

[8] Gérard, Y., "Contribution à la géométrie discrète," Ph.D. thesis, Université d'Auvergne - Clermont-Ferrand I (1999).

[9] Graham, R. L., *An efficient algorithm for determining the convex hull of a finite planar set*, Information Processing Letters **1** (1972), pp. 132–133.

[10] Jonas, A. and N. Kiryati, *Digital representation schemes for 3d curves*, Pattern Recognition **30** (1997), pp. 1803–1816.

[11] Kovalevsky, V. A., *New definition and fast recognition of digital straight segments and arcs*, Proceedings of the tenth international conference on Pattern Analysis and Machine Intelligence (1990).

[12] Rosenfeld, A., *Digital straight lines segments*, IEEE Transactions on Computers (1974), pp. 1264–1369.

[13] Sklansky, J., *Measuring concavity on a rectangular mosaic*, IEEE Trans. Comput. **21** (1972), pp. 1355–1364.

[14] Sladoje, N. and J. Zunić, *Ellipses estimation from their digitization*, , **1347**, 1997, pp. 187–198.

[15] Toussaint, G. T. and D. Avis, *On a convex hull algorithm for polygons and its application to triangulation problems*, Pattern Recognition (1982).

[16] Vialard, A., *Geometrical parameters extraction from discrete paths*, Discrete Geometry for Computer Imagery (1996).

[17] Vittone, J., "Caractérisation et reconnaissance de droites et de plans en géométrie discrète," Ph.D. thesis, Université Joseph Fourier-Grenoble 1 (1999).