An elementary algorithm for digital arc segmentation

David Coeurjolly, Yan Gerard, Jean-Pierre Reveillès, Laure Tougne

An Elementary Algorithm for Digital Arc Segmentation

D. Coeurjolly[a], Y. Gérard[b], J.-P. Reveillès[b] and L. Tougne[a]

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

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

This paper concerns the digital circle recognition problem, especially in the form of the circular separation problem. General fundamentals, based on classical tools, as well as algorithmic details are given (the latter by providing pseudo-code for major steps of the algorithm). After recalling the geometrical meaning of the separating circle problem, we present an incremental algorithm to segment a discrete curve into digital arcs.

1. Introduction

Euclidean shape recognition is an important topic of Discrete Geometry. Many studies have been done for straight lines [8,23], planes [6,26] and algorithms become more and more efficient. Even if some solutions are proposed for higher order objects such as conics [24] or general polynoms [11], further developments remain to be done.

This paper concerns the *digital circle recognition problem* and more precisely the *circular separating algorithm*. This paper reports on 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.

In the literature, first results on digital circle recognition have been proposed in the same time as digital circle or disk definitions. Hence Nakamura *et al.* [22] propose an recursive algorithm to detect the center of an digital circle. However, the computational cost of such an algorithm is exponential in a general case. In [17], Kim proposes an algorithm to detect if a set of grid points in a $N \times N$ image is a digital disk. The complexity bound is $O(N^3)$. In [18], Kim *et al.* propose an optimization of this in order to reduce the complexity to $O(N^2)$.

Based on links between digital circle and the classical separating arc problem in computation geometry, Fisk presents an digital disk recognition algorithm $O(N^2)$ using simplier

geometric structure than Kim [10]. This algorithm also constructs the complete solution domain, so called *arc center domain* in the following. Based on a similar approach, Kovalevsky [19], presents a digital circle recognition algorithm of a digital curve. However, complexity bounds of this algorithm are not provided and one can estimate computational cost in $O(n^2 \cdot \log(n))$ where n is the number of points of the digital curve.

More recently, Damaschke [3] presents a linear in time digital circle detection algorithm based on linear programming tools. More precisely, Damaschke proves that the recognition problem is equivalent to solve a set of n inequalities in dimension 3 (where n is the number of pixels of the digital curve). Hence, the Megiddo's algorithm [21] is used to decide if a solution exists in $O(n)$.

Finally, Worring *et al.* [29] present a digital circle estimation algorithm based on a fixed size window process. More precisely, they compute complete solution domains for digital curves whose lengths are less than a fix number r and they present a statistical process to generalize the local circle recognition to estimate the circularity of a general digital curve.

In this paper, we present both a digital circle recognition algorithm that describes all the solution domain and a segmentation process of a digital curves into pieces of digital circles.

After recalling the geometrical meaning of the separating circle problem, we present an elementary algorithm based on duality, the formal approach to Hough transform, for the digital circle recognition problem. This algorithm is then applied in order 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.

In the following, all time complexity bounds are given considering the model of computation in which number of comparisons are considered.

2. The Separating Arc Problem

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

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

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

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

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

This means that the set $acd(S, T)$ (*i.e.* arc center domain) of circle centers 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 hollow circles and the points of T are filled circles.

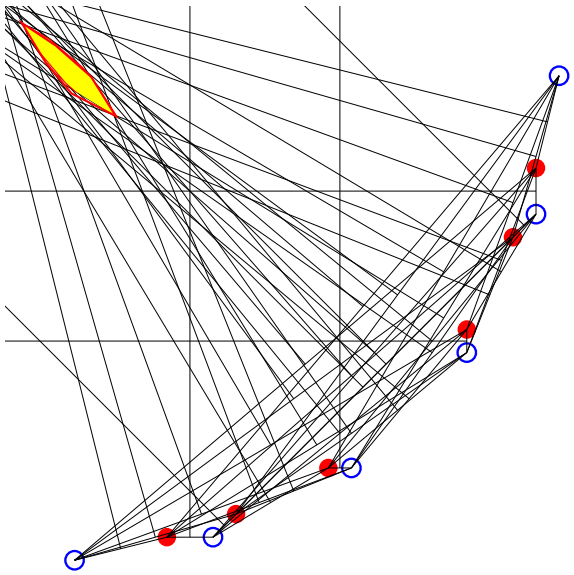

Figure 1. Set $acd(S,T)$ (upper left corner polygon) of centers of separating circles.

Various algorithms exist for the construction of set $acd(S,T)$ [10] [19] [3] [5], we present in this article a very elementary algorithm using *duality*, (the theoretical content of the *Hough transform*), which could be useful for the Imagery applications that 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 and the leftmost point L of convex set $acd(S,T)$ is unique. 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 $\{v_i^-\}_{i=0,\dots,m}$ the vertices of L^-, where $v_0^- = R$ and $v_m^- = L$ and similarly by $\{v_i^+\}_{i=0,\dots,n}$, the vertices of L^+ where, this time, $v_0^+ = L$ and $v_m^+ = R$. In the same way edges of L^- are denoted by $\{e_i^-\}_{i=1,\dots,m}$ and those of L^+ are denoted by $\{e_i^+\}_{i=1,\dots,n}$.

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 chords $\overrightarrow{st}, s$ belonging to S and t to 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. Even if lines of the larger family $Ineqs^+$ are easily selected among bisectors of chords $\overrightarrow{st}$, the main problem to determine L^+ is to find those which appear in their lower envelope.

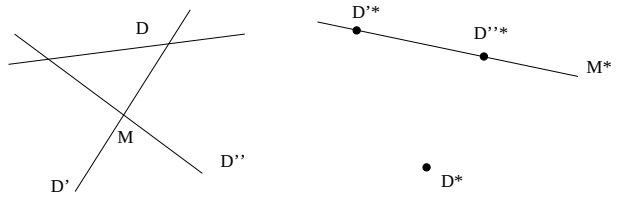

Figure 2. Illustration of Proposition 1.

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 a line whose equation is $y=ax+b$ to the point $(a,-b)$. 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 1 *Point $M=(u,v)$ of primal space belongs to half-plane $ax+by+c \leq 0$, 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 $au+bv+c \leq 0$.

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

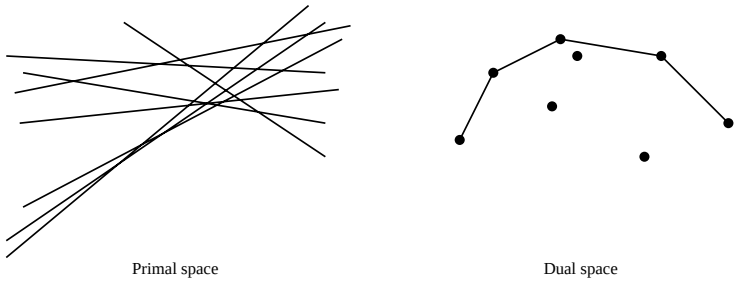

Figure 3. The lower envelope of a family of lines corresponds to the vertices of the upper part of the convex hull of their dual transform.

The following result is an immediate consequence of this observation.

Proposition 2 *Let $\mathcal{F}$ be a set of lines (none is parallel to the Oy axis) and $\mathcal{F}^*$ the set of points which are the line dual transform of $\mathcal{F}$. Then duality maps lines of the lower envelope of $\mathcal{F}$ one-to-one into the vertices of the upper part of the convex hull of $\mathcal{F}^*$.*

Using duality, the lower envelope of a line family 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 1 *The set $acd(S, T)$ can be obtained in $\mathcal{O}(card(S) \cdot card(T) \cdot \log(card(S) \cdot card(T))$ time using only elementary primitives like points-lines duality and 2D convex hull.*

From Proposition 2, it follows that construction of the upper boundary L^+ of $acd(S, T)$ is clearly obtained in $\mathcal{O}(card(S) \cdot card(T) \cdot \log(card(S) \cdot card(T))$ time; the same can be said for its lower boundary L^-. Intersection of these polygonal curves takes at most $\mathcal{O}(card(S) \cdot card(T))$ time; the result follows.

For the sake of clarity, we denote (a, b, c) the inequality $ax + by + c \leq 0$. We 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 $\forall t \in T$
$Ineqs^+ = \{(a,b,c) \in Ineqs \quad / \quad b < 0\}$
$Ineqs^- = \{(a,b,c) \in Ineqs \quad / \quad b > 0\}$
$C^+ =$ convexhull$(dual(Ineqs^+))$
$C^- =$ 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. As the cost of this processing is $\mathcal{O}(card(S) \cdot \log(card(S)))$, this solution does not increase the complexity of the overall algorithm.

2.2.2. Reducing T

In the same way let us consider two points t and 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', this circle does not contain t.

Proposition 3 *If a point t in T is such that the convex hull of points $S \cup \{t\}$ contains another point $t' \in T$, then t can be removed out of T for the separating arc problem.*

This leads to process T points as followed: adjoint a point $t \in T$ to the convex hull of S and remove t out of T if $convex\_hull(S,t)$ contains another point of T. A simple $\mathcal{O}(card(T) \cdot \log(card(T))$ 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 [10]. 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:

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

This 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,*i.e* those having a bounded Voronoï cell $vor(s,T)$ and the others, 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. Quasi-Circular polyline in $\mathbb{Z}^2$

We first consider two points in $\mathbb{Z}^2$ denoted by m and n. For the sake of clarity, we consider that a segment $[mn]$ is in the first octant (see figure 4). The problem is to separate all pixels above $[mn]$ from m and n.

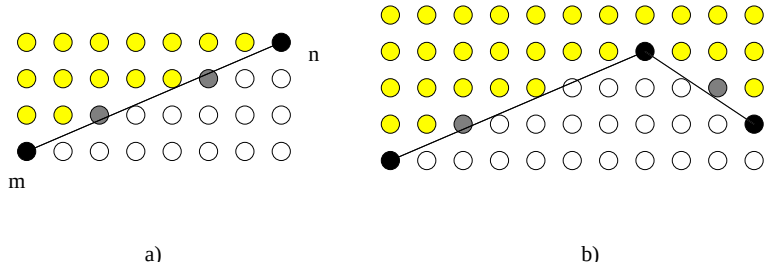

Figure 4. a) Bezout points associated to a segment $[mn]$, b) Bezout points associated to a convex polyline.

We consider an arithmetical description of the segment $[mn]$, *i.e* if we have the directional vector of $[mn]$ denoted by $\overrightarrow{u} = (a,b)^T$ with $a, b \in \mathbb{Z}$ and $gcd(a,b) = 1$.

Definition 1 (Bezout point) *A point $\mathcal{B}$ is a Bezout point of a discrete segment defined by an arithmetical directional vector $\overrightarrow{u}$ if and only if:*

$$\overrightarrow{m\mathcal{B}} = \overrightarrow{v} + k\overrightarrow{u} \quad with \quad k \in \mathbb{Z} \quad and \quad det(\overrightarrow{u}, \overrightarrow{v}) = \pm 1$$

and $\mathcal{B}$ is the closest point to the middle of $[mn]$.

Such grid point is called *Bezout point* because $\overrightarrow{v}$ is given by the classical Bezout's theorem in number theory. According to the sign of the determinant, two Bezout points can be associated to a segment, one point is above $[mn]$ and the other one is beneath $[mn]$. Figure 5 illustrates construction of one of these points.

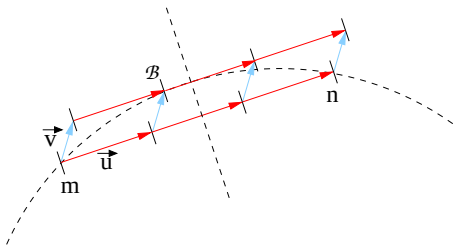

Figure 5. The point $\mathcal{B}$ is a Bezout point associated to $[mn]$.

Theorem 1 *Let us consider $[mn]$ in the first octant, T the set of grid points above $[mn]$ and $t \in T$. Then, all triangles (m, n, t) contain another point $t' \in T$ except for Bezout points.*

Proof: without loss of generality, we consider $[mn]$ such that $\overrightarrow{mn} = \overrightarrow{u} = (a, b)^T$. First of all, we consider a Bezout point t. Hence, according to previous definition, $det(\overrightarrow{mn}, \overrightarrow{mt}) = \pm 1$. Thus no grid point belong to the parallelogram defines by $\overrightarrow{mn}$ and $\overrightarrow{mt}$. Finally, no grid points are contained inside the triangle (m, n, t).

If t is not a Bezout point, and since $det(\overrightarrow{u}, \overrightarrow{u}) = \pm 1$, we have:

$$\overrightarrow{mt} = k\overrightarrow{u} + k'\overrightarrow{v}$$

with $k, k' \in \mathbb{Z}$ and $k' > 1$.

To complete the proof, we use the Pick's formula (see [12]):

$$I = A - \frac{B}{2} + 1$$

where I is the number of grid points inside a polygon which vertices are grid points, A denotes the area of such polygon and B the number of grid points in the polygon boundary.

In our case, we have:

$$A = \frac{|det(\overrightarrow{mt}, \overrightarrow{mn})|}{2}$$

Since $|det(\overrightarrow{u}, \overrightarrow{u})| = 1$ and $k' \geq 2$, we obtain:

$$2I \geq 2|k| - B + 2$$

If a grid point t' belongs to the triangle boundary, the proof is done. Otherwise, we have $B = 3$ and thus:

$$2I \geq 2|k| - 1$$

If $k = 0$ then there exists at least one grid point t' in the edge $[mt]$. So suppose $k \neq 0$. Hence $k' > 2$ leads to a triangle (m, n, t) with a point inside. So we just have to consider

the case where $k' = 2$. In such a case, if $k = 1$ there is exactly one grid point belonging to $[nt]$. Similarly, if $k = -1$ there is exactly one grid point on $[nt]$. Hence we have $|k| > 1$ and finally:

$$2I \geq 3$$

which concludes the proof.□

Hence, according to Proposition 4, if we want to separate all pixels above $[mn]$ from m and n, we just have to consider points m, n and the Bezout point above $[mn]$ (see figure 4).

If we consider a convex polyline that is a sequence of points in $\mathbb{Z}^2$ such that the polygonal arc is convex, we can define the quasi-circularity property on such a curve:

Definition 2 *A convex polyline Γ in $\mathbb{Z}^2$ is quasi-circular if we can separate by a circle all the grid points inside the convex hull of Γ from all the exterior points to the hull.*

Based on both previous definitions, we have the important property:

Proposition 4 *A convex polyline $\Gamma = \{v_i\}_{i=1..n}$ is quasi-circular if and only if:*

$$acd(\{v_i\}_{i=1..n}, \{\mathcal{B}(v_i, v_{i+1})\}_{i=1..n-1}) \neq \emptyset$$

where $\mathcal{B}(v_i, v_{i+1})$ denotes the Bezout point of the edge $[v_i, v_{i+1}]$ exterior to Γ.

3.2. Quasi-circular digital curve

Based on the previous definitions, we can define the quasi-circularity property on digital convex curves. In the following, we call a digital curve a finite simple 8-path [23]. We also use the classical convexity definition [15,16]: a set S of connected grid points is convex if there exits a convex region R in $\mathbb{R}^2$ such that:

$$S = \mathbb{Z}^2 \cap R$$

Moreover, a digital curve is convex if it is a piece of a convex digital set boundary.

First of all, we remind an important property that describes links between digital curves and Euclidean arcs (see [14] for a survey on digitization schemes and formal definition of the Object Boundary Quantization):

Proposition 5 *A digital convex curve is the digitization of an Euclidean arc, using the Object Boundary Quantization (OBQ) digitization scheme, if and only if we can separate curve's pixels from exterior pixels by a circle.*

According to the proposition 4, we have the property:

Proposition 6 *The following properties are equivalent:*

- *a digital convex curve is the digitization of an Euclidean circle*
- *the polyline $\Gamma = \{v_i\}_{i=1..n}$ defined by the convex hull of the digital curve is quasi-circular*
- $acd(\{v_i\}_{i=1..n}, \{\mathcal{B}(v_i, v_{i+1})\}_{i=1..n-1}) \neq \emptyset$

The proof of this proposition is straightforward according to definitions and the quasi-circularity property.

3.2.1. General process

We can detail in a few words the general process of both algorithms we present in order to decide if a digital curve is quasi-circular or to segment the curve into maximum quasi-circular parts.

So, the first step of the process is to compute the global polyline associated to a general digital curve. If this curve is convex, we compute the convex hull of the curve and we remove the edge between the first and the last points of the polyline (see figure 6). If the digital curve is not convex, we use a classical digital straight line segmentation algorithm in order to extract Strictly Convex or Concave parts (SCoC for short). Many linear in time algorithms exist for this segmentation process, see [8] or [20] for example. According to this SCoC segmentation, we can compute convex hulls on each part and thus construct the global polyline. SCoC parts can be treated independently because it is a pre-segmentation of the curve into digital circle.

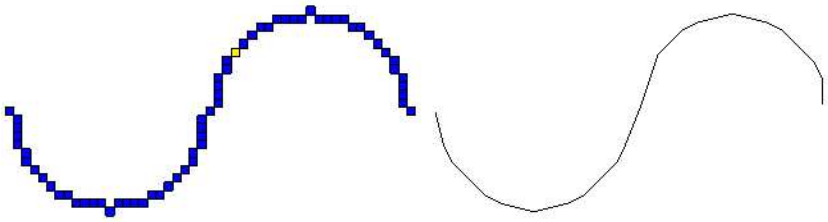

Figure 6. A SCoC segmentation of a S curve and the global associated polyline.

3.2.2. Arithmetical convex hull

In classical computational geometry, the convex hull computation of N points requires an $\mathcal{O}(N \cdot \log(N))$ time. In the digital case, some algorithms exist to compute such a hull in a linear time [27]. Hence, the global polyline construction is linear in the number of pixels.

For the *acd* problem, in order to be able to compute the Bezout points associated to an edge in a constant time, we need an arithmetical description of each edge of the convex hull, *i.e.* we must have the slope of each edge in an irreducible form. A naive approach would consist in appling the Euclide's algorithm at each edge to reduce the fractions but the cost is really expensive: the arithmetical convex hull would be in $\mathcal{O}(N \cdot \log(size_{max}(slope))$ where $size_{max}(slope)$ denotes the maximum arithmetical size of the slopes. However, arithmetical properties in $\mathbb{Z}^2$ can be used to construct efficiently such an algorithm.

For the sake of clarity, we just consider convex hull on digital straight segment (DSS for short) but this algorithm can be generalized to convex curves [7]. If we consider a DSS whose slope is $(a, b) \in \mathbb{Z}^2$ with $gcd(a, b) = 1$, we use the Klein's diagram based on the

continuous fraction analysis of a DSS slope [4,12]. This algorithm constructs, from the classical Euclide's algorithm of b/a, the set of convex hull points. Hence, in $\log(\min(a,b))$ time, we can construct the arithmetical description of all the convex hull edges of the DSS.

Since the complexity of the DSS segmentation algorithm is linear in the number of pixels and bounds the cost of the Klein's diagram, the global complexity of the arithmetical convex hull is linear in the number of pixels.

In the following parts, we first present an 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 digital curve.

3.2.3. Digital circle recognition algorithm

Let us first consider a sequence of edges of a SCoC $\mathcal{C}_i$. We consider all the edges at the same time. The following algorithm decides if a SCoC digital curve is the digitization of an Euclidean arc. If the curve is quasi-cicular, the complete description of the *acd* is returned.

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 an edge
    Let compute $\mathcal{B}$ the weakly exterior point
    $S := S + \{v_i, v_{i+1}\}$
    $T := T + \{\mathcal{B}\}$
end for
Compute $acd(S,T)$ using the algorithm 1
Return $acd(S,T)$

If we denote by N the number of edges 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 $\mathcal{O}(N^2)$ constraints.

3.2.4. Digital arc segmentation algorithm

In this case, we consider a general digital curve and its associated polyline. The maximum *acd* is an incremental implementation of the global algorithm. We add edges of the polyline one by one and we test if the *acd* is empty or not. If the *acd* is empty, we initialize a new domain.

Algorithm 3: Maximum digital arc segmentation

Input: A digital 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 an edge
    Let compute $\mathcal{B}$ the weakly exterior point
    $S := S + \{v_j, v_{j+1}\}$
    $T := T + \{\mathcal{B}\}$
    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 $\mathcal{O}(N^2)$ constraints at each of the N steps, we obtain $\mathcal{O}(N^3)$ constraints.

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

3.3. Complexity analysis

If we remind the previous notations, n denotes the number of pixels of the digital curve and N the number of edges of the polyline Γ. First, we use the theorem:

Theorem 2 (Acketa and Žunić [1]) *The maximal possible number of edges $e(m)$ of a convex digital polygon included into an $m \times m$-grid is bounded by:*

$$e(m) = \frac{12}{(4\pi^2)^{1/3}} m^{2/3} + \mathcal{O}(m^{1/3} \cdot \log(m))$$

Similar results with slightly different theorem statements may be found in [13] and [28].

Hence if n is the number of points of a digital convex curve, the number of edges N of the polyline Γ is bounded by $\mathcal{O}(n^{2/3})$. Using this bound, we can give the complexity of proposed algorithms. Note that for both algorithms, the polyline Γ is computed in $\mathcal{O}(n)$:

- Digital circle recognition algorithm: since we have shown that the number of constraints is in $\mathcal{O}(N^2)$, we can apply the Algorithm 1 to compute the vertices of the convex set, maybe empty, in $\mathcal{O}(n^{4/3} \cdot \log(n))$.

- Digital arc segmentation algorithm: since we only use classical computational geometry tools (points-lines duality, convex hulls), we can transform Algorithm 1 into a on-line algorithm without changing the complexity. Hence, the digital arc segmentation algorithm is also computed in $\mathcal{O}(n^{4/3} \cdot \log(n))$.

In Kovalevsky's approach [19], no complexity bound is provided but if the optimal Algorithm 1 is used to test the arc separability, we can bound the computational cost by $\mathcal{O}(n^2 \cdot \log(n))$. As a matter of fact, in the worst case, all exterior grid points have to be taken into account (see Figure 7).

(a) (b)

Figure 7. Comparison between the Kovalevsky's approach and our proposal

3.4. Reversibility

Let us consider a piece of the digital 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 7 *Let $acd(S,T)$ be the arc center domain defined by a digital 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 [14] of $\mathcal{A}$ is contains all points of $\mathcal{C}$.*

This property is proved by the construction of the domain.

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

3.5. Application to curvature estimation

Let us suppose our goal to be the curvature estimation at each point of the digital 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 [2]; this computation can be done by 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 digital curve based on convex hull vertices and edges. Based on this segmentation, we can compute a curvature estimation at each pixel of an arc by the mean value of the *acd* computed during the segmentation. This approach leads to a global calculus of the curvature and some errors are introduced near extremities. Most of all this approach is up to the digital arc segmentation initialization. Hence, we propose a local characterization of the curvature based on the same tools but with appropriate characteristics points:

Definition 3 *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 digital 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, which 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 digital curve

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

Note : the digital 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 digital straight segment) AND ($[MR]$ is a digital straight segment) **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 $\mathcal{O}(n)$ 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 digital curve

Input: A digital 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 $\mathcal{O}(n^2)$ complexity. However, we can use digital tangents defined by [25] in order to bound the digital segment growth. As a matter of fact these digital tangents are defined by the longest digital segment at a point of $\mathcal{C}$ and thus half-tangent at M define the longest possible segment for $[ML]$ and $[MR]$. Since these digital tangents can be computed in $\mathcal{O}(n)$ at each point of the curve $\mathcal{C}$ [9] 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 $\mathcal{O}(\log(n))$ to check if the *acd* is empty or not ($i.e\,\mathcal{O}(\log(n))$ to compute the new equation of the segment and $\mathcal{O}(1)$ to test the *acd*). We finally obtain a global complexity in $\mathcal{O}(n \cdot \log(n)^2)$.

In figure 8, we display the curvature graph on a digital circle of radius 40. If we compare these results to previous algorithms (see [2] for example), the curvature estimation based on a 5-point *acd* is not enough accurate to be relevant and further work is needed. However, this approach is theoretically consistent with the Euclidean osculating circle 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 9, 12 and in table 1, we show the results obtained for a circle of radius 100 grid units. 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.

In figures 10 and 11, we present examples of the digital arc segmentation algorithm. Since both the SCoC segmentation and the maximum *acd* process are greedy, they do not guarantee a minimal number of digital arcs.

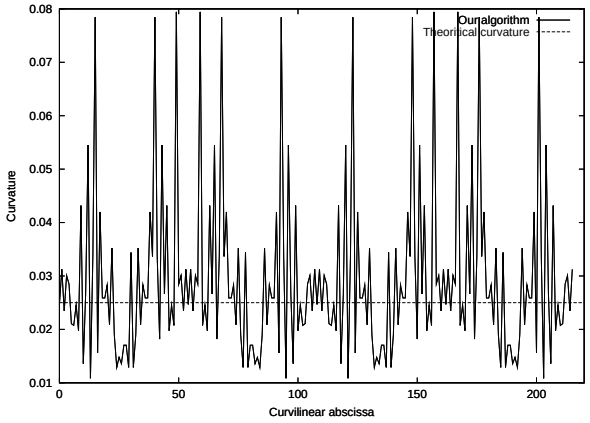

Figure 8. Curvature graph on a digital circle of radius 40.

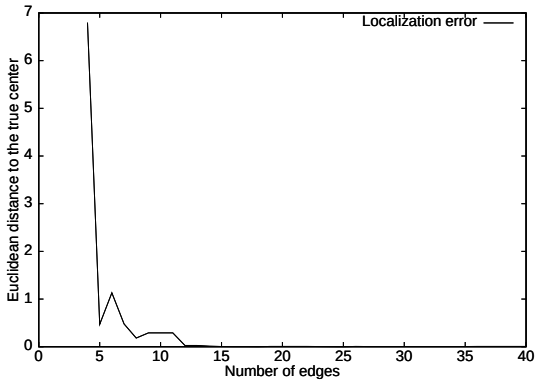

Figure 9. Euclidean distance between the center of a digital circle of radius 100 grid units and the mid point of $acd(S,T)$.

# 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 of the *acd* verticies on a digital circle of radius 100.

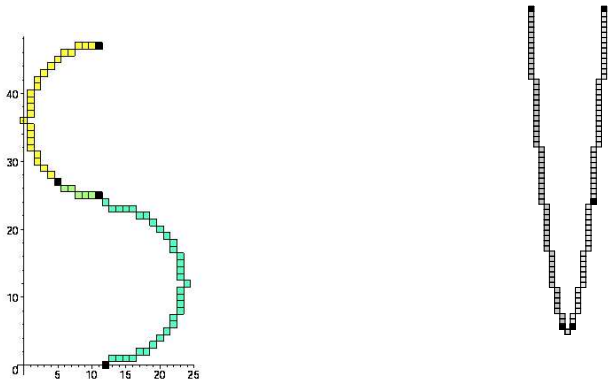

Figure 10. Examples of the digital arc segmentation: segmentation of a S shape and a parabola.

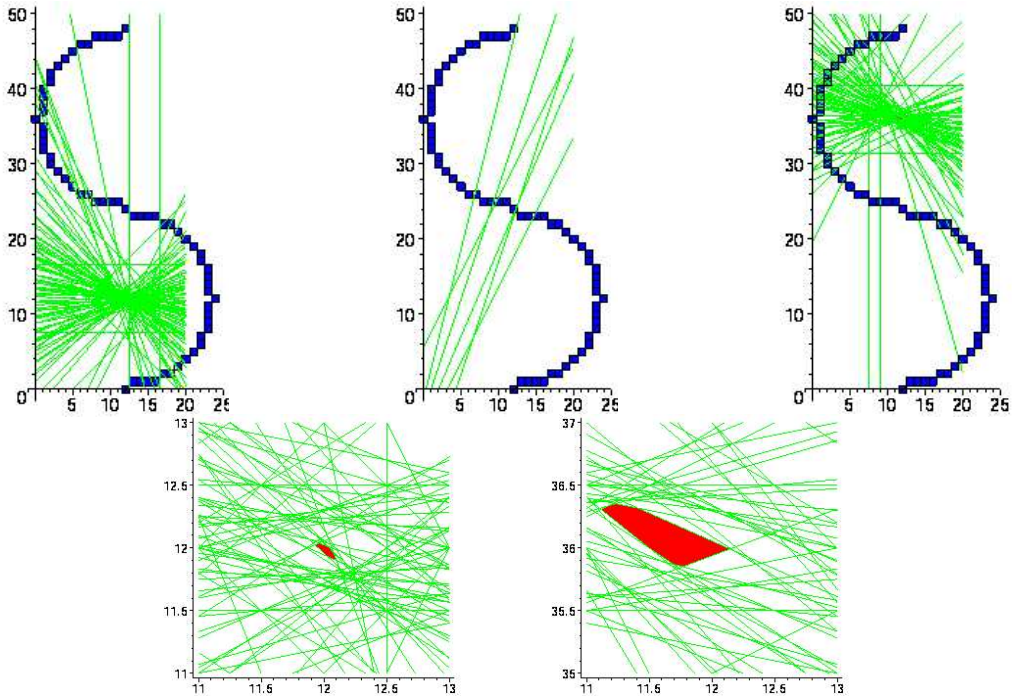

Figure 11. Details of the S shape segmentation: the first three figures represent the inequation systems attached to the digital arcs, the last two figures detail the first and the last *acd* (note that the true centers are respectively $(12, 12)$ and $(12, 36)$).

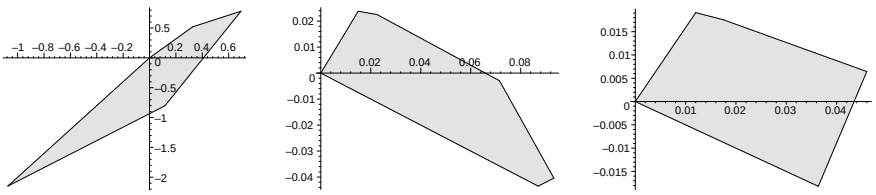

Figure 12. Some examples of the convex set evolution when we increase the number of edges on the circle of radius 100.

5. Conclusion and future works

Our purposed approach uses elementary geometrical notions only : finding circles separating two sets of points and one obvious property of $\mathbb{Z}^2$ geometry with Bezout points. This allows conceiving of various simple algorithms leading to efficient programs. If query for a single separating circle is needed, 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 the local digital curvature variations on algebraic curve discretizations, comparison of our approach with those using higher order derivatives in the definition of digital curvature, or the recognition of higher differential invariants. On the digital arc segmentation, the proposed algorithm opens an important problem: how to segment a digital curve into minimal number of digital arcs ?

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.

Acknowledgment

We would like to thank reviewers for their fruitful comments.

REFERENCES

1. D.M. Acketa and J.D. Žunić. On the maximal number of edges of convex digital polygons included into a $m \times m$-grid. *Journal of Combinatorial Theory*, Serie A(69):358–368, 1995.
2. D. Coeurjolly, S. Miguet, and L. Tougne. Discrete curvature based on osculating circle estimation. In *International Workshop on Visual Form 4*, number 2059, pages 303–312. Springer Lecture Notes in Computer Science, 2059, May 2001. Capri, Italy.

3. P. Damaschke. The linear time recognition of digital arcs. *Pattern Recognition Letters*, (16):543–548, May 1995.
4. H. Davenport. *The higherArithmetic : An Introduction to the Theory of Numbers.* Cambridge University Press, 7th edition, 1952.
5. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry.* Springer-Verlag, 2000.
6. I. Debled. *Etude et reconnaissance des droites et plans discrets.* PhD thesis, Université Louis Pasteur - Strasbourg, 1995.
7. I. Debled-Rennesson, J.L. Rémy, and J. Rouyer-Degli. Detection of discrete convexity of polyominoes. In *9th International Conferencee in Discrete Geometry for Computer Imagery*, pages 491–504, Uppsala, Sewden, December 2000.
8. I. Debled-Rennesson and J. P. Reveillès. A linear algorithm for segmentation of digital curves. *Parallel Image Analysis: Theory and Applications*, pages 73–100, 1993.
9. F. Feschet and L. Tougne. Optimal time computation of the tangent of a discrete curve : Application to the curvature. In *Discrete Geometry for Computer Imagery*, pages 31–40, 1999.
10. S. Fisk. Separating points sets by circles, and the recognition of digital disks. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 8(4):554–556, 1986.
11. Y. Gérard. *Contribution à la géométrie discrète.* PhD thesis, Université d'Auvergne - Clermont-Ferrand I, 1999.
12. G.H. Hardy and E.M. Wright. *Introduction to the theory of numbers.* Oxford press, 1975.
13. I. V. Jarnik. Über die die gitterpunkte auf konvexen kurven. *Math. Zeitschrift*, 24:500–518, 1926.
14. A. Jonas and N. Kiryati. Digital representation schemes for 3d curves. *Pattern Recognition*, 30(11):1803–1816, 1997.
15. C. Kim. On the cellular convexity of complexes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3:617–625, 1981.
16. C. Kim and A. Rosenfeld. Convex digital solids. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(6):612–618, November 1982.
17. C. E. Kim. Digital disks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:372–374, 1984.
18. C. E. Kim and T. A. Anderson. Digital disks and a digital compactness measure. In *Seventh International Conference on Pattern Recognition (Montreal, Canada, July 30-August 2, 1984)*, IEEE Publ. 84CH2046-1, pages 254–257. IEEE, 1984.
19. V. A. Kovalevsky. New definition and fast recognition of digital straight segments and arcs. *Proceedings of the tenth international conference on Pattern Analysis and Machine Intelligence*, pages 31–34, June 1990.
20. M. Lindenbaum and A. Bruckstein. On recursive, o(n) partitioning of a digitized curve into digital straigth segments. *IEEE Transactions on PatternAnalysis and Machine Intelligence*, 15(9):949–953, september 1993.
21. N. Megiddo. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM Journal of Computing*, 12(4):759–776, November 1983.
22. Akira Nakamura and Kunio Aizawa. Digital circles. *Computer Vision, Graphics, and Image Processing*, 26(2):242–255, May 1984.

23. A. Rosenfeld. Digital straight lines segments. *IEEE Transactions on Computers*, pages 1264–1369, 1974.
24. N. Sladoje and J. Zunić. Ellipses estimation from their digitization. In Springer, editor, *Lecture Notes in Computer Science*, volume 1347, pages 187–198, 1997.
25. A. Vialard. Geometrical parameters extraction from discrete paths. *Discrete Geometry for Computer Imagery*, pages 24–35, 1996.
26. J. Vittone. *Caractérisation et reconnaissance de droites et de plans en géométrie discrète*. PhD thesis, Université Joseph Fourier-Grenoble 1, décembre 1999.
27. K. Voss. *Discrete Images, Objects and Functions in* $\mathbb{Z}^n$. Number 11 in Algorithms and Combinatorics. Springer-Verlag, 1993.
28. K. Voss and R. Klette. On the maximal number of edges of convex digital polygons included into a square. *Computers Artificial Intelligence*, 1:549–558, 1982.
29. M. Worring and W.M. Smeulders. Digitized circular arcs: characterization and parameter estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):587–598, jun 1995.