On Three Constrained Versions of the Digital Circular Arc Recognition Problem

Tristan Roussillon*[1], Laure Tougne[1], and Isabelle Sivignon[2]

[1] Université de Lyon,
Université Lyon 2, LIRIS, UMR5205, F-69676, FRANCE
{tristan.roussillon, laure.tougne}@liris.cnrs.fr
[2] Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, FRANCE
isabelle.sivignon@liris.cnrs.fr

Abstract. In this paper, the problem of digital circular arcs recognition is investigated in a new way. The main contribution is a simple and linear-time algorithm for solving three subproblems: online recognition of digital circular arcs coming from the digitization of a disk having either a given radius, a boundary that is incident with a given point, or a center that is on a given straight line. Solving these subproblems is interesting in itself, but also for the recognition of digital circular arcs. Indeed the proposed algorithm can be iteratively used for the recognition of circular arcs. Moreover, since the algorithm is online, it provides a way for segmenting digital curves.

1 Introduction

This paper deals with three constrained versions of a well-known problem: the recognition of digital circular arcs (DCAs for short). Many authors have proposed a solution to the recognition of digital circles [7, 11, 6, 9, 12, 16, 5, 3, 2, 14]. Some techniques are not adapted for DCAs, like [16], and only a few ones are online [9, 2]. Even if a linear algorithm has been proposed for a long time [11], using a sophisticated machinery coming from linear programming [10], no solution is known to be truly fast and easy to implement. That's why further research on the topic is needed.

We opt for an original approach of the problem. Indeed, we study three constrained versions of the DCA recognition problem: (i) the case of disks of given radius, (ii) the case of disks whose boundary is incident with a given point, (iii) the case of disks whose center is on a given straight line. We show that deciding whether a part of a digital boundary is the digitization of one of these disks is done with a simple, online and linear-time algorithm.

Solving these constrained problems is interesting in itself. For instance, the proposed algorithm can be applied to the case of disks of infinite radius in order to provide a way for recognizing digital straight segments. Thus, our method is a

* Author supported by a grant from the DGA

nice tool to highlight what makes the problems studied in this paper be similar or different from the problem of digital straight segments recognition.

Moreover solving such constrained problems is also useful to solve the unconstrained problem. Indeed the proposed algorithm can be iteratively used for the recognition of DCAs. This new technique may be coarsely described as follows: if a new foreground (resp. background) point is located outside (resp. inside) the current smallest separating disk, then either the new point is on the new smallest separating disk, or the sets of foreground and background points are not circularly separable at all. In the aim of deciding between these two alternatives, the proposed algorithm can be used in the case (ii), *i.e.* when the boundary of the disks must be incident with a given point.

Section 2 is made up of formal definitions and a brief review of the literature. The main results are presented in Section 3. The main algorithm is described and proved in Section 3.3. We show how to use it for recognition of DCAs in Section 4.

2 Preliminaries

2.1 Digital Boundary and Digital Contour

A binary image I is viewed as a subset of points of $\mathbb{Z}^2$ that are located inside a rectangle of size $M \times N$. A digital object O is defined as a 4-connected subset of $\mathbb{Z}^2$ (Fig. 1.a). Its complementary set $\bar{O} = I \backslash O$, which is assumed to be connected, is the so-called background. The digital boundary B (resp. $\bar{B}$) of O (resp. $\bar{O}$) is defined as the 8-connected clockwise-oriented list of the digital points having at least one 4-neighbour in $\bar{O}$ (resp. O). (Fig. 1.b).

Let us assume that each digital point of O is considered as the center of a closed square of size 1×1. The topological border of the union of these squares defines the digital contour C of O (Fig. 1.c). C is a 4-connected clockwise-oriented list of points with half-integer coordinates (Fig. 1.c).

Each point of C is numbered according to its position in the list. The starting point, which is arbitrarily chosen, is denoted by C_0 and any arbitrary point of the list is denoted by C_k. A part (C_iC_j) of C is the list of points that are ordered increasingly from index i to j (Fig. 1.d).

An elementary part bounded by two consecutive points (C_kC_{k+1}) separates a point of B (on its right side) from a point of $\bar{B}$ (on its left side) (Fig. 2.a). Let us denote by $B_{(C_iC_j)}$ (resp. $\bar{B}_{(C_iC_j)}$) the list of digital points of B (resp. $\bar{B}$) that are located on the right (resp. left) side of each elementary part (C_kC_{k+1}) of (C_iC_j) with $i \leq k < j$.

2.2 Digital Circle and Circular Arc

Definition 1 (Digital circle (Fig. 2.b)) *A digital contour C is a digital circle iff there exists a Euclidean disk $\mathcal{D}(\omega, r)$ that contains B but not $\bar{B}$.*

Definition 2 is the analog of Definition 1 for parts of C.

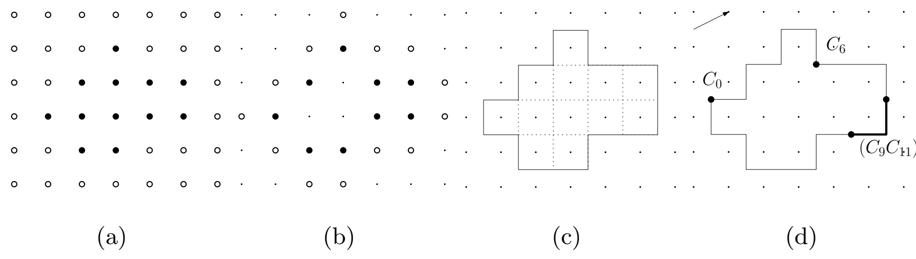


Fig. 1. (a) A digital object in black points and its complementary set in white points, (b) their digital boundaries and (c) their digital contour. (d) Notations

Definition 2 (Circular arc (Fig. 2.c)) *A part* (C_iC_j) *of* C *(with* $i < j$*) is a circular arc iff there exists a Euclidean disk* $\mathcal{D}(\omega, r)$ *that contains* $B_{(C_iC_j)}$ *but not* $\bar{B}_{(C_iC_j)}$.

This definition is equivalent to the one of Kovalevsky [9].

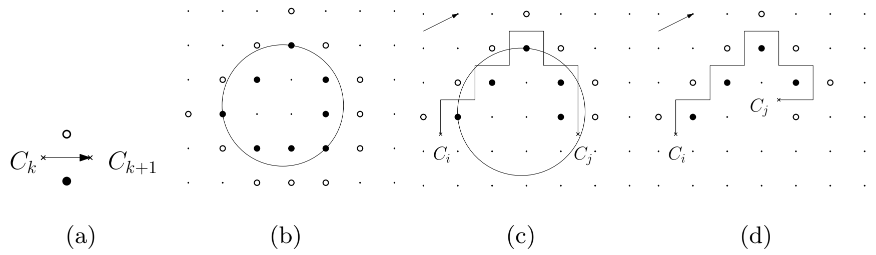


Fig. 2. (a) An elementary part. (b) A digital circle. (c) A circular arc. (d) A part that is not a circular arc

Problem 1 (DCA recognition) *Given a part* (C_iC_j) *of* C*, the DCA recognition problem consists in deciding whether* (C_iC_j) *is a DCA or not, and if so, computing the parameters of (at least) one Euclidean disk separating* $B_{(C_iC_j)}$ *from* $\bar{B}_{(C_iC_j)}$, i.e. *containing* $B_{(C_iC_j)}$ *but not* $\bar{B}_{(C_iC_j)}$.

2.3 State of Art

Three different but related approaches are used to solve Problem 1 (see Appendix A of [15]) - n is the length of the part (C_iC_j):

1. Problem 1 consists in searching for a 3D point belonging to the intersection of $2n$ half-spaces in the (ω_x, ω_y, r) parameter space: [11, 16, 3]. Therefore,

Megiddo's algorithm can be used [10] in order to derive an algorithm in $\mathcal{O}(n)$. An online version of this algorithm exists [1] but is difficult to implement.
2. If the parameter space is projected along the r-axis onto the (ω_x, ω_y)-plane, Problem 1 consists in searching for a 2D point belonging to the intersection of n^2 half-planes. This approach, which requires massive computation, has been widely used [6, 9, 12, 2] and several optimizations have been proposed. Kovalevsky [9] removes some points during the computation but without improving the worst-case bound. Coeurjolly *et al.* [2] proposed a preprocessing stage using the arithmetic properties of digital curves so that the time complexity of their algorithm goes down from $\mathcal{O}(n^2 \log n)$ to $\mathcal{O}(n^{4/3} \log n)$. As noticed in [2], these algorithms may be made online with an incremental convex hull algorithm [13].
3. In the space that is dual to the parameter space, Problem 1 consists in searching for a plane separating two sets of n 3D points [11, 5, 14]. The quadratic algorithm of Kim [8] can be straightforwardly interpreted in this way. Using classical results about the computation of 3D convex hulls [13] and the computation of the vertical distance between two convex polyhedra [13], this approach leads to an algorithm whose time complexity is bounded by $\mathcal{O}(n \log n)$ [14]. Though, this algorithm is not online.

3 Main Results

In this section, we study three constrained versions of Problem 1.

3.1 Definitions

We define three classes of constrained disks that fulfill the following property: a constrained disk is uniquely defined by two points. The set of constrained disks fulfilling one of the three conditions of Definition 3 is called a *class of constrained disks*.

Definition 3 (Constrained disks) *A* constrained *disk is such that one of the three following conditions is fulfilled: (i) it has a given radius and an orientation is arbitrarily chosen (Fig. 3.a and Fig. 3.b), (ii) its boundary is incident with a third point (Fig. 3.c), (iii) its center is on a given straight line (Fig. 3.d).*

Problems 2, 3 and 4 are the analog of Problem 1 for a specific class of contrained disks:

Problem 2 *Computing the parameters of the set of Euclidean disks $\mathcal{D}(\omega, r = r_0)$ separating $B_{(C_iC_j)}$ from $\bar{B}_{(C_iC_j)}$, where r_0 is fixed and given as input.*

Problem 3 *Computing the parameters of the set of Euclidean disks $\mathcal{D}(\omega, r)$ separating $B_{(C_iC_j)}$ from $\bar{B}_{(C_iC_j)}$, such that $\mathcal{D}$ touches a fixed point P_0 given as input.*

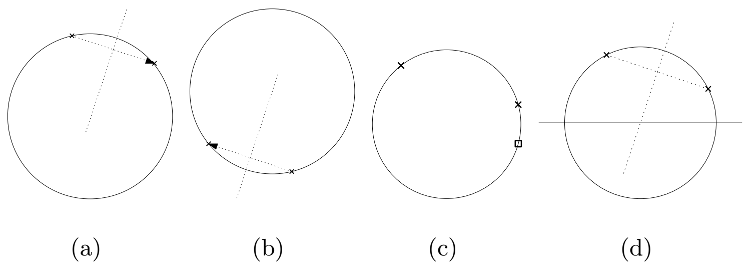


Fig. 3. One and only one disk is incident with the two points, labeled by a cross each, because a radius and an orientation have been chosen in (a) and (b), a third point depicted with a square has been given in (c) and the center has to be on the solid horizontal straight line in (d).

Problem 4 *Computing the parameters of the set of Euclidean disks $\mathcal{D}(\omega, r)$ separating $B_{(C_iC_j)}$ from $\bar{B}_{(C_iC_j)}$, such that ω belongs to a fixed straight line $\mathcal{L}_0$ given as input.*

In Sections 3.2 and 3.3, we assume that a class of constrained disks is given.

3.2 Circular Hulls and Points of Support

Definition 4 (Circular hull) *Let L be an ordered list of points. Its* inner *(resp.* outer*) circular hull is a list of some points of L such that, for each pair of consecutive points, all the points of L belong (resp. do not belong) to the constrained disk defined by the two points.*

Fig. 4 displays the inner and outer circular hulls of a list of points in the case of a given radius. Notice that the intrinsic order of the points determines a natural orientation so that one and only one disk of a given radius is defined by two points. If the radius is infinite, the circular hull of L is a part of the convex hull of L. As a consequence, the circular hull is easily computed with an online and linear-time algorithm in the style of the Graham's scan thanks to the intrinsic order of the points.

In order to solve Problems 2, 3 and 4, the constrained disks separating $B_{(C_iC_j)}$ from $\bar{B}_{(C_iC_j)}$ have to be computed. Some special points, called *points of support*, play a key role in the computation.

Definition 5 (Point of support) *A point of $B_{(C_iC_j)}$ or $\bar{B}_{(C_iC_j)}$ that is located on the boundary of a constrained disk separating $B_{(C_iC_j)}$ from $\bar{B}_{(C_iC_j)}$ is called* point of support*.*

The following propositions, which are related to the points of support, are proved in Appendix B of [15]:

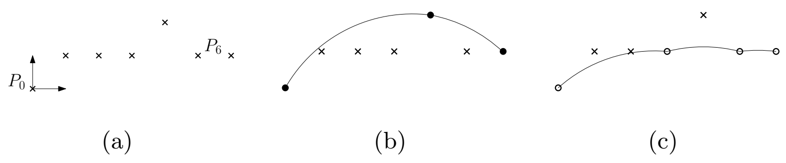


Fig. 4. Inner (b) and outer (c) circular hull of a list of points (a) when the radius of the disks is fixed ($r_0 = 4$).

Proposition 1 *$B_{(C_iC_j)}$ and $\bar{B}_{(C_iC_j)}$ are separable by a constrained disk iff $B_{(C_iC_j)}$ and $\bar{B}_{(C_iC_j)}$ contain at least one point of support.*

Proposition 2 *The points of support of $B_{(C_iC_j)}$ (resp. $\bar{B}_{(C_iC_j)}$) are consecutive points of the inner circular hull of $B_{(C_iC_j)}$ (resp. outer circular hull of $\bar{B}_{(C_iC_j)}$).*

Proposition 3 *The points of support of $B_{(C_iC_j)}$ and $\bar{B}_{(C_iC_j)}$ define the whole set of separating constrained disks.*

The first and last points of support of the inner circular hull of $B_{(C_iC_j)}$, respectively denoted by I_f and I_l, as well as the first and last points of support of the outer circular hull of $\bar{B}_{(C_iC_j)}$, respectively denoted by O_f and O_l, play a key role in the algorithm that checks the separability of $B_{(C_iC_j)}$ and $\bar{B}_{(C_iC_j)}$.

3.3 Separability

Algorithm 1 solves Problems 2, 3 and 4. The points of a part (C_iC_j) are processed one by one. Assume that the k first points have already been processed. When a new point C_{k+1} is taken into consideration, the inner and outer points defined by the elementary part (C_kC_{k+1}) (Fig. 2.b) are respectively added to the lists $B_{(C_iC_k)}$ (lines 4 and 5) and $\bar{B}_{(C_iC_k)}$ (lines 7 and 8).

Algorithm 1 calls Algorithm 2 on lines 5 and 8. It updates the inner and outer circular hulls as well as the points of support when a new point N is added.

Let us assume that N is an inner point. The case where N is an outer point is similar. If N does not belong to the constrained disk touching I_f and O_l (area 1 of Fig.5), $B_{(C_iC_{k+1})}$ and $\bar{B}_{(C_iC_k)}$ cannot be separated by a constrained disk and the Algorithm 2 returns false (lines 1 and 2).

If N belongs to the constrained disk touching I_f and O_l (areas 2 and 3 of Fig.5), $B_{(C_iC_{k+1})}$ and $\bar{B}_{(C_iC_k)}$ are still separable. If N does not belong to the disk touching O_f and I_l (area 2 of Fig.5), the inner circular hull is updated (lines 5-7) and the points of support are updated too (lines 8-10).

After this brief description of Algorithm 1 and 2, let us prove the following theorem:

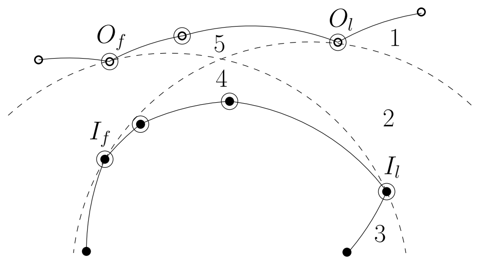
Theorem 1 *Algorithm 1 correctly retrieves the set of constrained disks separating $B_{(C_iC_j)}$ from $\bar{B}_{(C_iC_j)}$ in linear time.*

Algorithm 1: Algorithm that solves Problems 2, 3 and 4

```
   Input: A part $(C_iC_j)$ (with $i < j$).
   Output: The boolean value $areSeparable$ and $OHull$, $IHull$, $O_f$, $O_l$, $I_f$, $I_l$.
   /* Initialization                                                    */
1  Initialization of $IHull$, $I_f$, $I_l$ (resp. $OHull$, $O_f$, $O_l$) with the inner (resp. outer)
   point of $(C_iC_{i+1})$;
2  $areSeparable \leftarrow true$; $k \leftarrow i + 1$;
   /* Scan                                                              */
3  while $C_k < C_j$ and $areSeparable$ do
4      $N \leftarrow$ inner point of $(C_kC_{k+1})$;
5      $areSeparable \leftarrow$ Separability($N$,$OHull$,$IHull$,$O_f$,$O_l$,$I_f$,$I_l$);
6      if $areSeparable$ then
7          $N \leftarrow$ outer point of $(C_kC_{k+1})$;
8          $areSeparable \leftarrow$ Separability($N$,$IHull$,$OHull$,$I_f$,$I_l$,$O_f$,$O_l$);
9      $k \leftarrow k + 1$;
10 return $areSeparable$
```

Algorithm 2: Separability(N, $OHull$, $IHull$, O_f, O_l, I_f, I_l)

```
   Input: $OHull$, $IHull$, $O_f$, $O_l$, $I_f$, $I_l$ and a new point $N$
   Output: a boolean value and updated $OHull$, $IHull$, $O_f$, $O_l$, $I_f$, $I_l$
1  if $N$ is outside the constrained disk touching $I_f$ and $O_l$ then
2      return false;
3  else
4      if $N$ is outside the constrained disk touching $O_f$ and $I_l$ then
           /* update of the inner circular hull                         */
5          while $N$ is outside the constrained disk touching the last two points of
           $IHull$ do
6              The last point of $IHull$ is removed from $IHull$;
7          $N$ is added to $IHull$;
           /* update of the points of support                           */
8          $I_l \leftarrow N$;
9          while $N$ is outside the constrained disk touching the first two points of
           support of $OHull$ do
10             $O_f \leftarrow$ the point of $OHull$ that is just after $O_f$;
11     return true;
```



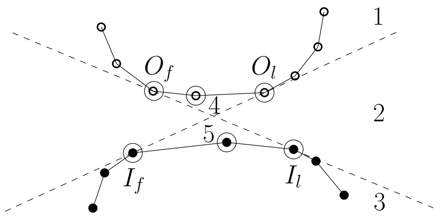


Fig. 5. The points of support are encircled. To help the reader to figure out why the role of the points of support is so important, here are two examples. In (a) the radius of the constrained disks is equal to 5, whereas in (b) the radius is infinite. The first and last points of support of each hull delineate 5 areas numbered from 1 to 5.

Proof. Thanks to Proposition 3, we know that the whole set of separating constrained disks is given by the points of support of $B_{(C_iC_j)}$ and $\bar{B}_{(C_iC_j)}$. Therefore, showing that the algorithm properly retrieves the points of support of $B_{(C_iC_j)}$ and $\bar{B}_{(C_iC_j)}$ in linear time is sufficient to prove Theorem 1. Moreover, since Algorithm 1 completely depends on the correctness of Algorithm 2, we can focus on Algorithm 2. To prove that it properly updates the points of support, we have to show that a new point involves the removal of the p last points of support of $B_{(C_iC_j)}$ and the removal of the q first points of support of $\bar{B}_{(C_iC_j)}$. Because of the intrinsic order of the points, a new point cannot be in areas 4 and 5 of Fig. 5, but only in areas 1, 2 or 3. As a consequence, the points of support lying in the middle of the list of consecutive points of support of $B_{(C_iC_j)}$ (resp. $\bar{B}_{(C_iC_j)}$) cannot be removed if those lying at the front (resp. back) are not removed too. Algorithm 2 correctly computes p and q, because the points of the circular hulls are sequentially scanned respectively from front to back (lines 5-7) and back to front (lines 9 and 10).

Each point is added and removed once at most in the inner circular hull as well as in the list of points of support, implying that Algorithm 1 is linear-time.

□

Our algorithm applies to Problems 2, 3 and 4: the only thing that change is the implementation of the predicate: "is N outside the constrained disk touching P_1 and P_2 ?" (lines 1, 4, 5 and 9 in Algorithm 2). In addition, notice that in the three different implementations the computation may use integers only.

Fig. 6.a illustrates the outcome of Algorithm 1. A part of a digital ellipse has been scanned and iteratively decomposed into DCAs of radius 10.

4 Digital Circular Arc Recognition

In this section, we propose a method that iteratively solves Problem 3 (online recognition of a DCA coming from the digitization of a disk having a boundary that is incident with a given point) to solve the unconstrained problem, *i.e.* Problem 1.

The method computes the smallest disk that separates $B_{(C_iC_j)}$ from $\bar{B}_{(C_iC_j)}$. In the (ω_x, ω_y, r)-space (Section 2.3), the smallest separating disk corresponds to a 3D point that belongs to the intersection of $2n$ half-spaces and that is the closest to the paraboloid of equation $r^2 = {\omega_x}^2 + {\omega_y}^2$. Due to the convexity of the paraboloid, a classic result of convex programming [4] implies the following property: if a new inner (resp. outer) point is located outside (resp. inside) the current smallest separating disk, then either the new point is on the new smallest separating disk, or the sets of inner and outer points are not circularly separable at all. In the aim of deciding between these two alternatives, Algorithm 1 can be used in the case where the disks have to be incident with a given point.

Similarly to Algorithm 1, the points of a part (C_iC_j) are processed one by one in Algorithm 3.

Algorithm 3: Algorithm that solves Problem 1

```
   Input: A part $(C_iC_j)$ (with $i < j$).
   Output: The boolean value $areSeparable$ and $\mathcal{D}_{min}$.
   /* Initialization                                                      */
1  $\mathcal{D}_{min}$ is set to a disk of null radius and whose center is on the inner point of
   $(C_iC_{i+1})$;
2  $areSeparable \leftarrow true$; $k \leftarrow i+1$;
   /* Scan                                                                */
3  while $C_k < C_j$ and $areSeparable$ do
4      $N \leftarrow$ inner point of $(C_kC_{k+1})$;
5      $areSeparable \leftarrow$ CircularSeparability($N$,$\mathcal{D}_{min}$,$(C_iC_k)$);
6      if $areSeparable$ then
7          $N \leftarrow$ outer point of $(C_kC_{k+1})$;
8          $areSeparable \leftarrow$ CircularSeparability($N$,$\mathcal{D}_{min}$,$(C_iC_k)$);
9      $k \leftarrow k+1$;
10 return $areSeparable$
```

Algorithm 4 is called instead of Algorithm 2 when a new inner (resp. outer) point N is added to $B_{(C_iC_k)}$ (resp. $\bar{B}_{(C_iC_k)}$) (lines 5 and 8 of Algorithm 3).

Let us assume that N is an inner point. The case where N is an outer point is similar. If the inner point is located inside the current smallest separating disk, then Algorithm 4 returns true (line 2) because the current disk is still separating. Otherwise, the constrained disks are defined as touching the new point that makes the current disk not separating. We use Algorithm 1 to decide whether $B_{(C_iC_k)}$ and $\bar{B}_{(C_iC_k)}$ are separable by a disk whose boundary is incident

Algorithm 4: CircularSeparability(N,$\mathcal{D}_{min}$,(C_iC_k))

```
Input: A new point $N$, the smallest separating disk $\mathcal{D}_{min}$ and the part $(C_iC_k)$.
Output: A boolean value and updated $\mathcal{D}_{min}$.
1 if $N$ is inside $\mathcal{D}_{min}$ then
2     return true;
3 else
      /* The boundary of the constrained disks has to be incident with
         $N$                                                            */
4     Use Algorithm 1 to decide whether $B_{(C_iC_k)}$ and $\bar{B}_{(C_iC_k)}$ are separable by a
      constrained disk whose boundary is incident with $N$;
5     if $B_{(C_iC_k)}$ and $\bar{B}_{(C_iC_k)}$ are separable then      /* update of $\mathcal{D}_{min}$ */
6         $\mathcal{D}_{min} \leftarrow \mathcal{D}_{new}$, the smallest separating disk that is incident with $N$;
7         return true;
8     else
9         return false;
```

with N (line 4). If $B_{(C_iC_k)}$ and $\bar{B}_{(C_iC_k)}$ cannot be separated by a constrained disk whose boundary is incident with N (line 5), then it is a classical result of quadratic programming [4] that $B_{(C_iC_k)}$ and $\bar{B}_{(C_iC_k)}$ are not circularly separable at all. Algorithm 4 returns false (line 9). Otherwise, the set of points of support defines the set of Euclidean disks whose boundary is incident with the new point N and that separate $B_{(C_iC_k)}$ from $\bar{B}_{(C_iC_k)}$ according to Proposition 3. Among all these disks, finding the smallest one, denoted by $\mathcal{D}_{new}$, is done in linear time and $\mathcal{D}_{min}$, the current smallest separating disk, is thus updated in linear time (line 6).

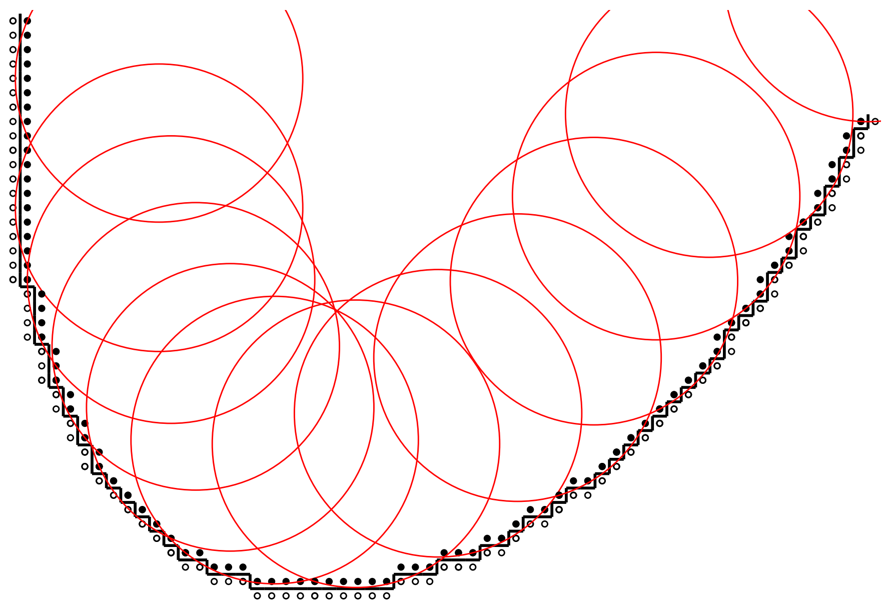
All the points of $B_{(C_iC_k)}$ and $\bar{B}_{(C_iC_k)}$, *i.e* $\mathcal{O}(n)$ points, are scanned at each call of Algorithm 4. Moreover, Algorithm 4 is called at most n times in Algorithm 3. Hence, the whole algorithm is quadratic. However, we can use the preprocessing stage proposed by Coeurjolly *et al.* [2] so that the time complexity of the algorithm goes down from $\mathcal{O}(n^2)$ to $\mathcal{O}(n^{4/3})$.

Since the algorithm is online, the segmentation of digital curves into DCAs is done without any increase of the time complexity, that is in $\mathcal{O}(n^{4/3})$. This technique has been implemented and Fig. 6.b illustrates the segmentation of a part of a digital ellipse into DCAs. For each DCA, the circle drawn is the smallest separating circle.

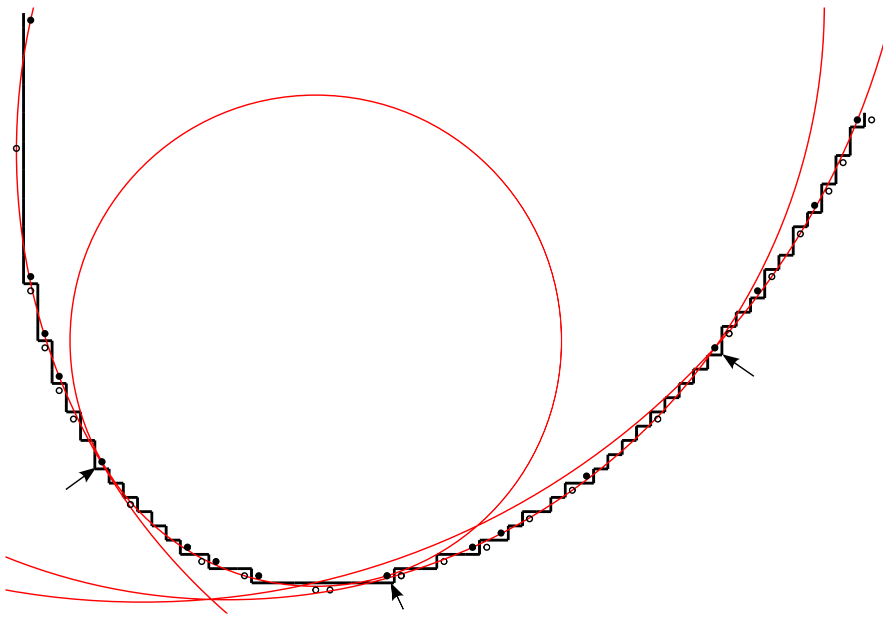
5 Conclusion

A simple, online and linear-time algorithm is introduced to cope with three constrained problems: recognition of digital circular arcs coming from the digitization of a disk having either a given radius, a boundary that is incident with a given point or a center that is on a given straight line.

In addition to its theoretical interest, solving such constrained problems is valuable for the recognition of digital circular arcs. Our algorithm can be used



(a)



(b)

Fig. 6. Segmentation of a part of a digital ellipse into DCAs of given radius ($r = 10$) in (a) and of any radius in (b). The black polygonal line depicts the digital contour. The black and white points are those retained for the computation. In (b), the preprocessing stage proposed in [2] has discarded a great amount of black and white points. The arrows point to the end points of the DCAs

as a routine each time a new point is taken into consideration. Due to the optimization proposed in [2], this last method runs in $\mathcal{O}(n^{4/3})$, instead of being quadratic. Moreover, it is easy to implement, may use integers only and provides a way for fastly segmenting digital curves.

References

1. L. Buzer. A Linear Incremental Algorithm for Naive and Standard Digital Lines and Planes Recognition. *Graphical Model*, 65:61–76, 2003.
2. D. Coeurjolly, Y. Gérard, J-P. Reveillès, and L. Tougne. An Elementary Algorithm for Digital Arc Segmentation. *Discrete Applied Mathematics*, 139(1-3):31–50, 2004.
3. P. Damaschke. The Linear Time Recognition of Digital Arcs. *Pattern Recognition Letters*, 16:543–548, 1995.
4. M. de Berg, M. van Kreveld, M. Overmars, and O. Scharzkopf. *Computation geometry, algorithms and applications*. Springer, 2000.
5. A. Efrat and C. Gotsman. Subpixel Image Registration Using Circular Fiducials. *International Journal of Computational Geometry & Applications*, 4(4):403–422, 1994.
6. S. Fisk. Separating Points Sets by Circles, and the Recognition of Digital Disks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:554–556, 1986.
7. C. E. Kim. Digital Disks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(3):372–374, 1984.
8. C. E. Kim and T. A. Anderson. Digital Disks and a Digital Compactness Measure. In *Annual ACM Symposium on Theory of Computing*, pages 117–124, 1984.
9. V. A. Kovalevsky. New Definition and Fast Recognition of Digital Straight Segments and Arcs. In *Internation Conference on Pattern Analysis and Machine Intelligence*, pages 31–34, 1990.
10. N. Megiddo. Linear Programming in Linear Time When the Dimension Is Fixed. *SIAM Journal on Computing*, 31:114–127, 1984.
11. J. O'Rourke, S. R. Kosaraju, and N. Meggido. Computing Circular Separability. *Discrete and Computational Geometry*, 1:105–113, 1986.
12. S. Pham. Digital Circles With Non-Lattice Point Centers. *The Visual Computer*, 9:1–24, 1992.
13. F. P. Preparata and M. I. Shamos. *Computational geometry : an introduction*. Springer, 1985.
14. T. Roussillon, I. Sivignon, and L. Tougne. Test of Circularity and Measure of Circularity for Digital Curves. In *International Conference on Image Processing and Computer Vision*, pages 518–524, 2008.
15. T. Roussillon, I. Sivignon, and L. Tougne. On-Line Recognition of Digital Arcs. Technical report, RR-LIRIS-2009-008, 2009.
16. P. Sauer. On the Recognition of Digital Circles in Linear Time. *Computational Geometry*, 2(5):287–302, 1993.