On-line Recognition of Digital Arcs

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. This paper focuses on the on-line recognition of digital arcs. The main contribution is to propose a simple and linear algorithm for three subproblems: on-line recognition of digital arcs coming from the digitization of a disk having a fixed radius, a boundary that contacts a fixed point and a center that belongs to a fixed straight line.
Solving such subproblems is interesting in itself, but also for the recognition of digital arcs. Indeed the proposed algorithm can be used as an oracle in multidimensional search techniques or can be iteratively used in a naive manner. Moreover, since the algorithm is on-line, it is a means of segmenting digital curves in a very fast way.

1 Introduction

This paper deals with the on-line recognition of digital arcs. Many authors have proposed a solution to the recognition of digital circles: [Kim84], [OKM86], [Fis86], [Kov90], [Pha92], [Sau93], [EG94], [Dam95], [CGRT04], [RST08]. Some techniques are not adapted for digital arcs, like [Sau93], and only a few ones are on-line: [Kov90,CGRT04]. Even if a linear algorithm has been proposed for a long time [OKM86], using a sophisticated machinery coming from linear programming [Meg84], 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 digital arc recognition problem: (i) the case of disks of fixed radius, (ii) the case of disks whose boundary contacts a fixed point, (iii) the case of disks whose center belongs to a fixed 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, on-line and linear-in-time algorithm.

Solving such constrained problems is interesting in itself. For instance, if the radius is fixed at infinite, the proposed algorithm is a means of recognizing digital straight segments. Nevertheless, solving such constrained problems is also useful to efficiently solve the unconstrained problem. On the one hand, the proposed algorithm can be used as an oracle in multidimensional search techniques such

* Author supported by a grant from the DGA

as the Megiddo's algorithm [Meg84], which can be made on-line [Buz03]. The technique may be roughly described as follows [Meg84]: to search for an optimal solution relative to m constraints in a space of dimension d, we solve two problems each in a space of dimension $d-1$ and then our problem is reduced to one with only a fraction of the m input constraints in a space of dimension d. In the aim of solving the two subproblems, instead of recursively applying the technique of Megiddo, we can use the proposed algorithm in the case (iii), that is when the center of the disks must belong to a fixed straight line. Using our algorithm is a means of reducing the constant and the burden of the implementation, known to be the two drawbacks of the technique.

On the other hand, the proposed algorithm can be iteratively used in a less sophisticated manner. This new technique, which is easier to implement than the former one, 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 smallest separating disk passes through the new point, or the sets of foreground and background points are not circularly separable. In the aim of deciding between these two alternatives, the proposed algorithm can be used in the case (ii), that is when the boundary of the disks must contact a fixed point.

The first section 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 digital arc recognition and digital arc segmentation 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 (without hole) of $\mathbb{Z}^2$ (Fig. 1.a). Its complementary set $\bar{O} = I \backslash O$ 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 whose coordinates are half-integer (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 $(C_i C_j)$ of C is the list of points that are ordered increasingly from index i to j (Fig. 1.d).

2.2 Digital Circle and Digital Arc

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

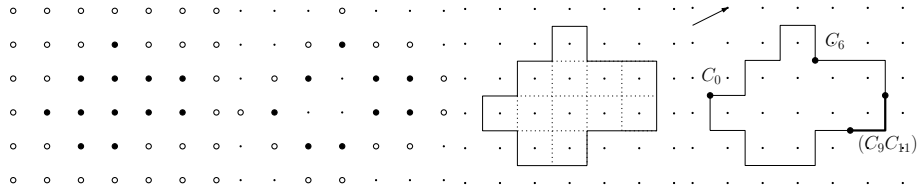


Fig. 1. (a) A digital object in black points and its complementary set in white points, (b) their digital boundaries and (c) their digital contour. (d) Notations

In order to state the analog of Definition 1 for parts of C, new notations have to be introduced. 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.b). 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 $i \leq k < j$.

Definition 2 (Digital arc (Fig. 2.c)) *A part (C_iC_j) of C is a digital 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 [Kov90].

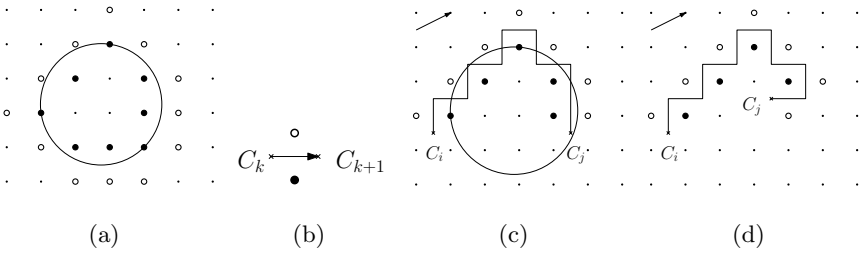


Fig. 2. (a) A digital circle. (b) An elementary part. (c) A digital arc. (d) A part that is not a digital arc

Problem 1 (Digital arc recognition) *Given a part (C_iC_j) of C (with $j-i = n$), the digital arc recognition problem consists in deciding whether (C_iC_j) is a digital arc 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 can be used (see Appendix A of [RST09]):

1. Problem 1 consists in searching for a 3D point belonging to the intersection of $2n$ half-spaces in the parameters space, that is the (ω_x, ω_y, r)-space: [OKM86], [Sau93], [Dam95]. Therefore, the Megiddo's algorithm can be used [Meg84] in order to derive an algorithm in $\mathcal{O}(n)$. This algorithm may be made on-line [Buz03] but is difficult to implement.
2. If the parameters 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 has been widely used ([Fis86], [Kov90], [Pha92], [CGRT04]) but requires massive computation. That's why several authors proposed an optimization. Kovalevsky [Kov90] removes some points during the computation (in the style of the Megiddo's algorithm) but without improving the worst-case bound. Coeurjolly *et al.* [CGRT04] 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 [CGRT04], these algorithms may be made on-line with an incremental convex hull algorithm [PS85].
3. In the space that is dual to the parameters space, problem 1 consists in searching for a plane separating two sets of n 3D points [OKM86,EG94,RST08]. Using classical results about the computation of 3D convex hulls [PS85] and the computation of the vertical distance between two convex polyhedra [PS85], this approach leads to an algorithm whose time complexity is bounded by $\mathcal{O}(n \log n)$ [RST08]. Though, this algorithm is not on-line.

3 Main Results

As it is quite difficult to implement a simple and on-line solution that solves Problem 1, we study in this section three constrained versions of this problem.

3.1 Definitions

Our results hold if a prior knowledge reduces to one the cardinality of the set of the disks touching two points:

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

The set of constrained disks fulfilling one of the three conditions of Definition 3 is called a *class of constrained disks*. Problems 2, 3 and 4 hold for a specific class of contrained disks:

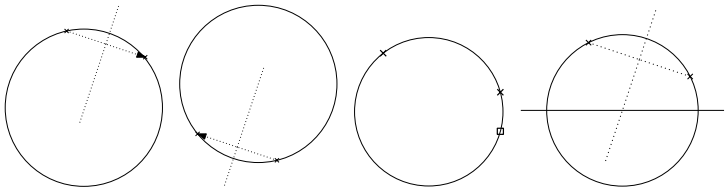


Fig. 3. One and only one disk touches the two points depicted with a cross, because a radius and an orientation have been chosen in (a) and (b), a third point depicted with a square has been fixed in (c) and the center has to belongs to the solid horizontal straight line in (d).

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.*

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 the sequel, we assume that a class of constrained disks is fixed.

3.2 Circular Hulls and Points of Support

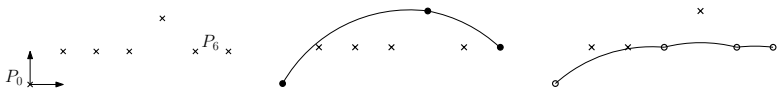
Definition 4 (Circular hull) *Let L be an ordered list of points. Its* inner *(resp.* outer*) circular hull is a list of some of the points of L such that, for each triplet of consecutive points, the third point and all the points of L belong (resp. do not belong) to the constrained disk defined by the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

The circular hull is easily computed with an on-line and linear-in-time algorithm in the style of the Graham's scan thanks to the intrinsic order of the points.

Fig. 4 displays the inner and outer circular hulls of a list of points in the fixed radius case.

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.



(a) (b) (c)

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

The following propositions, which are related to the points of support, are proved in Appendix B of [RST09]:

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

Only one algorithm 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 $B_{(C_iC_k)}$ and $\bar{B}_{(C_iC_k)}$.

Algorithm 1 gives operations done when the inner point, which is denoted by N, is added to $B_{(C_iC_k)}$. A similar algorithm may be sketched when the outer point is added to $\bar{B}_{(C_iC_k)}$.

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 1 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. The inner circular hull is updated (lines 4-6) and if N does not belong to the disk touching O_f and I_l (area 2 of Fig.5), the points of support are updated too (lines 8-10).

After that brief description of Algorithm 1, let us prove the following theorem:

Algorithm 1: Adding of an inner point (Problems 2, 3 and 4)

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

Theorem 1 *The algorithm that calls Algorithm 1 when an inner point is added to $B_{(C_iC_k)}$ and a similar algorithm when an outer point is added to $\bar{B}_{(C_iC_k)}$ correctly retrieves the set of constrained disks separating $B_{(C_iC_j)}$ from $\bar{B}_{(C_iC_j)}$ in linear time.*

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 enough to prove Theorem 1. Moreover, because the algorithm completely depends on the correctiveness of Algorithm 1 for inner points and of a modified version of Algorithm 1 for outer points, we can focus on Algorithm 1. Establishing that Algorithm 1 properly updates the points of

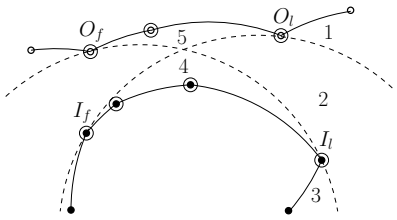


Fig. 5. The points of support are encircled. To help the reader to figure out why the role of the points of support is so important, here is a fake example where the radius of the constrained disks is equal to 5, but the same hold for other cases. The first and last points of support of each hull delineate 5 areas numbered from 1 to 5.

support requires showing that (i) 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 $\bar{B}_{(C_iC_j)}$ and (ii) Algorithm 1 correctly computes p and q. Because of the intrinsic order of the points, a new point clearly 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. In order to compute p and q, the points of the circular hulls are sequentially scanned respectively from front to back (lines 4-6) and back to front (lines 9 and 10). Knowing that (i) is true, it is clear from the design of Algorithm 1 that (ii) is true as well, which concludes the proof of correctiveness. Algorithm 1 is not constant at each adding, but is of order $\mathcal{O}(n)$ after n insertions. Indeed, each point is added and removed once at most in the inner circular hull as well as in the list of points of support.

□

Remark 1: Our algorithm holds for Problems 2, 3 and 4 because what makes each problem specific is limited to the implementation of the predicate: "is N outside the constrained disk touching P_1 and P_2 ?" (lines 1, 4, 7 and 9 in Algorithm 1). In addition, notice that in the three different implementations the computation may use integers only.

Remark 2: If $B_{(C_iC_{k+1})}$ and $\bar{B}_{(C_iC_k)}$ are not separable by a constrained disk (Algorithm 1 returns false), it is easy to know how to change the class of constrained disks so that $B_{(C_iC_{k+1})}$ and $\bar{B}_{(C_iC_k)}$ may be separated by a disk belonging to a different class of constrained disks. Suppose that $B_{(C_iC_{k+1})}$ and $\bar{B}_{(C_iC_k)}$ are as illustrated in Fig. 6.a. $B_{(C_iC_{k+1})}$ and $\bar{B}_{(C_iC_k)}$ cannot be separated by a constrained disk of fixed radius equal to 3. Because the constrained disk touching I_f and N contains O_l, if $B_{(C_iC_{k+1})}$ and $\bar{B}_{(C_iC_k)}$ are circularly separable, they only can be separated by a disk of radius strictly greater than 3, for instance equal to 5, as illustrated in Fig. 6.b.

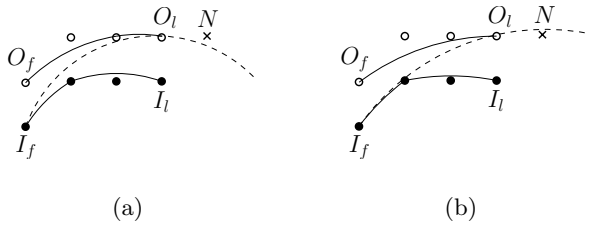


Fig. 6. $B_{(C_iC_{k+1})}$ and $\bar{B}_{(C_iC_k)}$ cannot be separated by a constrained disk of fixed radius equal to 3 (a), but are separable by a constrained disk of fixed radius equal to 5 (b)

This last remark plays an important role in the first technique presented in the next section in order to solve the digital arc recognition problem.

4 Digital Arc Recognition

In this section, we show two techniques that iteratively solve one of the three constrained problems to solve the unconstrained problem, that is Problem 1.

4.1 Linear-in-time Algorithm

As stated in Section 2.3, the Megiddo's technique can be used [Meg84,Buz03] in order to find the smallest disk that separates $B_{(C_iC_j)}$ from $\bar{B}_{(C_iC_j)}$ in $\mathcal{O}(n)$.

The technique [Meg84] may be roughly described as follows: to find the smallest separating disk relative to m constraints in the parameters space of dimension 3, we solve two problems each in a space of dimension 2 and then our problem is reduced to one with $\frac{m}{16}$ constraints in the parameters space of dimension 3. We can show that the global cost is bounded by $\mathcal{O}(m)$ ($m = 2n$ in our case).

Actually, for each subproblem, we want to know if one of the disks whose center belongs to a fixed straight line is the smallest separating disk. If not, we want to know on which side of the line the center of the smallest separating disk lies. The two straight lines of the two subproblems, are chosen in linear time such that the localization of the center of the smallest separating disk with respect the two lines implies the removal of $\frac{1}{16}$ of the constraints [Meg84]. In the aim of solving the two subproblems, instead of recursively applying the Megiddo's algorithm, we can use the fast algorithm proposed in the previous section.

Using our algorithm is a means of reducing the linearity coefficient and the burden of the implementation, known to be the two drawbacks of the technique. As these drawbacks are reduced but remain valid, we introduce an elementary algorithm in the next section.

4.2 Elementary Algorithm

The following technique is another means of computing the smallest disk that separates $B_{(C_iC_j)}$ from $\bar{B}_{(C_iC_j)}$. Thanks to the convexity of the objective function, the following property holds [dBvKOS00]: if a new inner (resp. outer) point is located outside (resp. inside) the current smallest separating disk, then either the smallest separating disk passes through the new point, or the sets of inner and outer points are not circularly separable. In the aim of deciding between these two alternatives, the algorithm presented in Section 3.3 can be used in the case where the disks have to touch a fixed point.

Similarly to Problems 2, 3 and 4, the points of a part (C_iC_j) are processed one by one. Algorithm 2 is used when the inner point defined by the elementary part (C_kC_{k+1}), which is denoted by N, is added to $B_{(C_iC_k)}$. A similar algorithm may be sketched when the outer point defined by (C_kC_{k+1}) is added to $\bar{B}_{(C_iC_k)}$.

If the inner point is located inside the current smallest separating disk, then Algorithm 2 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 (line 4). We look over all the inner and outer

Algorithm 2: Adding of an inner point (Problem 1)

Input: $\mathcal{D}_{min}$, the smallest separating disk and a new inner point N
Output: a boolean, updated $\mathcal{D}_{min}$

```
1  if N is inside D_min then
2      return true;
3  else
4      Consider the class of constrained disks touching N;
5      The first and last inner (resp. outer) points of support are set to the inner
       (resp. outer) point defined by (C_i C_{i+1}) ;          /* Initialization */
6      flag ← true; l ← i + 1;
7      while flag and l < k do                                  /* scan */
8          Let P (resp. Q) be the inner (resp. outer) point defined by (C_l C_{l+1});
9          flag ← Add P with Algorithm 1;
10         if flag then
11             flag ← Add Q with a version of Algorithm 1 valid for outer points;
12         l ← l + 1;
13     if flag then                                  /* update of D_min */
14         D_min ← D_new, the smallest separating constrained disk;
15     return flag;
```

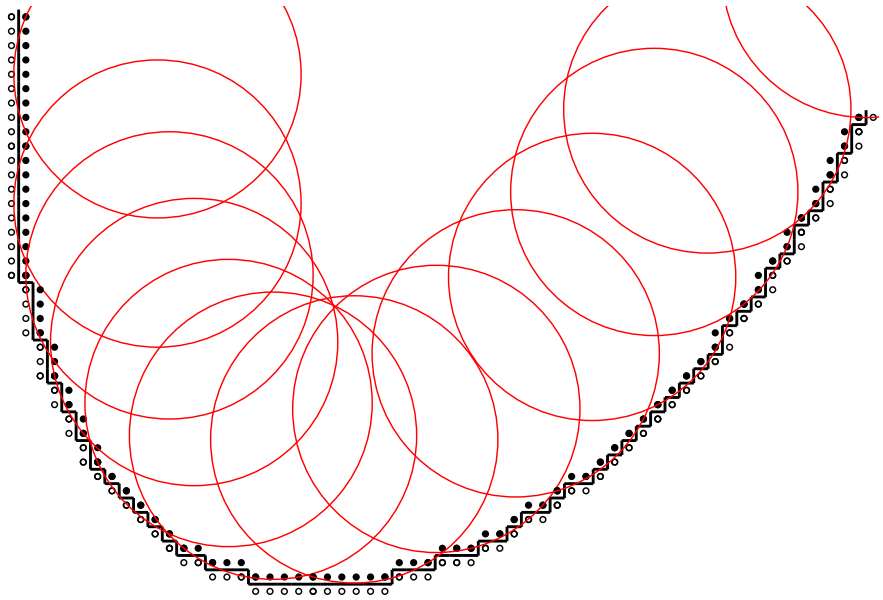
points that have been already processed (lines 5-12). If $B_{(C_iC_k)}$ and $\bar{B}_{(C_iC_k)}$ cannot be separated by a constrained disk touching the new point, then it is a classical result of quadratic programming and computational geometry [dBvKOS00] that $B_{(C_iC_k)}$ and $\bar{B}_{(C_iC_k)}$ are not circularly separable at all. Otherwise, the set of points of support defines the set of Euclidean disks that contact the new point and separate $B_{(C_iC_k)}$ from $\bar{B}_{(C_iC_k)}$ according to Proposition 3. Among all these disks, finding the smallest one is obviously done in linear time and the current smallest separating disk is thus updated in linear time (line 14).

In view of the fact that all the points are scanned at each new insertion in the worst case, it is clear that the whole algorithm is quadratic. However, we can use the preprocessing stage proposed by Coeurjolly *et al.* [CGRT04] so that the time complexity of the algorithm goes down from $\mathcal{O}(n^2)$ to $\mathcal{O}(n^{4/3})$.

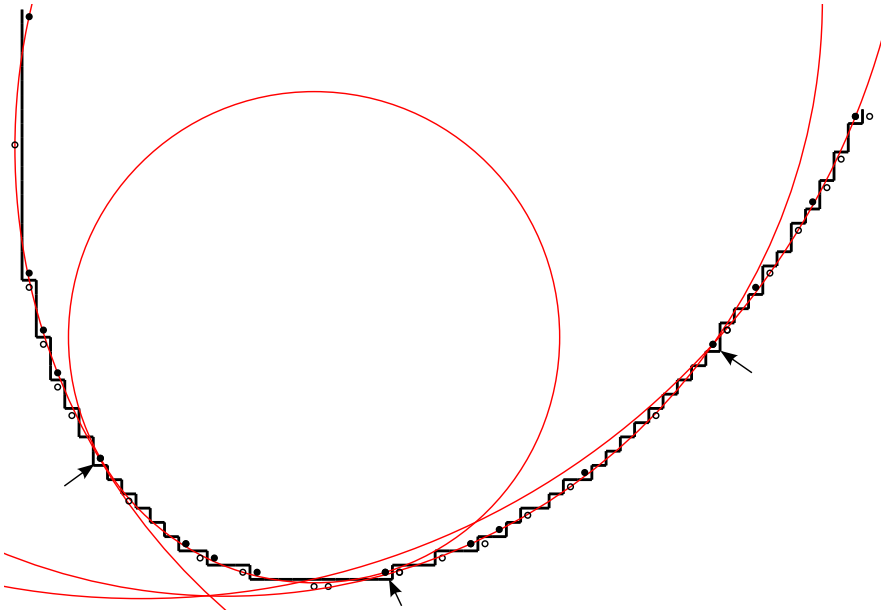
Since the algorithm is on-line, the segmentation of digital curves into digital arcs is done without any increase of the time complexity, that is in $\mathcal{O}(n^{4/3})$. This technique, which is considerably simpler than the one presented in Section 4.1, has been implemented. For instance, Fig. 7.b illustrates the segmentation of a part of a digital ellipse into digital arcs. For each digital arc, the circle drawn is the smallest separating circle.

5 Conclusion

A simple, on-line and linear-in-time algorithm is introduced to cope with three constrained problems: recognition of digital arcs coming from the digitization of



(a)



(b)

Fig. 7. Segmentation of a part of a digital ellipse into circular arcs of fixed 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 [CGRT04] has discarded a great amount of black and white points. The arrows point to the end points of the digital arcs

a disk having a fixed radius, a boundary that contacts a fixed point and a center that belongs to a fixed straight line.

Solving such constrained problems is valuable for the recognition of digital arcs. Our algorithm can be used as an oracle in multidimensional search techniques in order to recognize a digital arc in $\mathcal{O}(n)$ or can be used as a routine each time a new point is taken into consideration. Thanks to the optimization proposed in [CGRT04], this last method runs in $\mathcal{O}(n^{4/3})$, instead of being quadratic. Moreover, it is easy to implement, it may use integers only and is a means of segmenting digital curves in a fast way.

References

[Buz03] L. Buzer. A Linear Incremental Algorithm for Naive and Standard Digital Lines and Planes Recognition. *Graphical Model*, 65:61–76, 2003.

[CGRT04] 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.

[Dam95] P. Damaschke. The Linear Time Recognition of Digital Arcs. *Pattern Recognition Letters*, 16:543–548, 1995.

[dBvKOS00] M. de Berg, M. van Kreveld, M. Overmars, and O. Scharzkopf. *Computation geometry, algorithms and applications*. Springer, 2000.

[EG94] A. Efrat and C. Gotsman. Subpixel Image Registration Using Circular Fiducials. *International Journal of Computational Geometry & Applications*, 4(4):403–422, 1994.

[Fis86] 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.

[Kim84] C. E. Kim. Digital Disks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(3):372–374, 1984.

[Kov90] 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.

[Meg84] N. Megiddo. Linear Programming in Linear Time When the Dimension Is Fixed. *SIAM Journal on Computing*, 31:114–127, 1984.

[OKM86] J. O'Rourke, S. R. Kosaraju, and N. Meggido. Computing Circular Separability. *Discrete and Computational Geometry*, 1:105–113, 1986.

[Pha92] S. Pham. Digital Circles With Non-Lattice Point Centers. *The Visual Computer*, 9:1–24, 1992.

[PS85] F. P. Preparata and M. I. Shamos. *Computational geometry : an introduction*. Springer, 1985.

[RST08] 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.

[RST09] T. Roussillon, I. Sivignon, and L. Tougne. On-Line Recognition of Digital Arcs. Technical report, RR-LIRIS-2009-008, 2009.

[Sau93] P. Sauer. On the Recognition of Digital Circles in Linear Time. *Computational Geometry*, 2(5):287–302, 1993.