[Proc. SPIE Vol. 1818, "Visual Communications and Image Processing", pp. 862–876, Boston MA, November 1992]

Euclidean Skeletons and Conditional Bisectors

Hugues Talbot
Department of Materials Science and Engineering
Massachusetts Institute of Technology, Cambridge, MA 02139

Luc Vincent
Xerox Imaging Systems
9 Centennial Drive, Peabody, MA 01960

Abstract

This paper deals with the determination of skeletons and conditional bisectors in discrete binary images using the *Euclidean metrics.* The algorithm, derived from [18], proceeds in two steps: first, the Centers of the Euclidean Maximal Discs (CMD) included in the set to skeletonize are characterized and robustly identified. Second, a firefront propagation is simulated starting from the set boundaries, in which pixels which are not centers of maximal discs and are not crucial to homotopy preservation are removed. Not only is the resulting algorithm fast and accurate, it allows the computation of a vast variety of skeletons. Furthermore, it can be extended to provide *conditional bisectors* of any angular parameter θ. This leads to the introduction of a new morphological transformation, the *bisector function,* which synthesizes the information contained in all the θ-conditional bisectors. The interest of all these skeleton-like transformations is illustrated on the segmentation of binary images of glass fibers.

1 Introduction

The skeleton transformation is a widely used transformation in the field of image processing and mathematical morphology [13]. The skeleton $S(X)$ of a set $X \in \mathbb{R}^2$ can be described intuitively as the set of the median lines of X, i.e., the lines which are equally distant from two parts of the boundary of X. From $S(X)$, one can derive its end points, multiple points, length, loops, etc, all being important features for set description and characterization. This is the reason why skeletonization is successfully applied to various image processing problems, such as optical character recognition, biomedical imaging, materials science, etc.

The notion of skeleton was introduced by Blum in 1961 [1] as *medial axis transform.* It is based on the concept of grassfire: assuming a fire is propagating within X at a uniform speed starting from its boundaries, the medial axis of X is the set of the points where different firefronts meet (see Fig. 1). A more formal definition of the skeleton was proposed by Calabi in 1966 [4], which relies on the concept of *maximal ball.* A ball (i.e., a disc in the 2D case) B included in X is said to be maximal if and only if there exist no other ball included in X and containing B:

$$\forall B' \text{ ball}, \qquad B \subseteq B' \subseteq X \Longrightarrow B' = B. \tag{1}$$

The skeleton $S(X)$ of a set X is then defined as the set of the centers of its maximal balls:

$$S(X) = \{p \in X, \exists r \geq 0, B(p,r) \text{ maximal ball of } X\}. \tag{2}$$

From these definitions, one can see that the notion of skeleton is closely related to that of distance: the successive firefronts of definition 1 constitute the successive level lines of the *distance function* of the considered set [12] (Recall

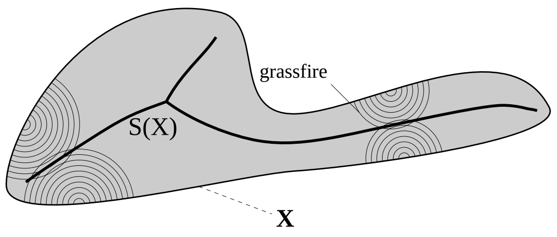


Figure 1 : *The skeleton is the locus of the pixels where "firefronts" meet.*

that the distance function $dist_X$ of a set X associate with each pixel p of X its distance d to the closest background pixel). Similarly, the balls of definition 2 are also defined with respect to a given metrics d.

In the discrete plane, the easiest and most widely used distances are the 4- and 8-connected distances, also called *city-block* and *chessboard* respectively [3]. These distances (together with hexagonal, dodecagonal and octagonal distances) are used in the vast majority of published skeleton algorithms. However, even for the simplest shapes, skeletons computed according to these metrics exhibit erroneous branches, as illustrated by Fig. 2.

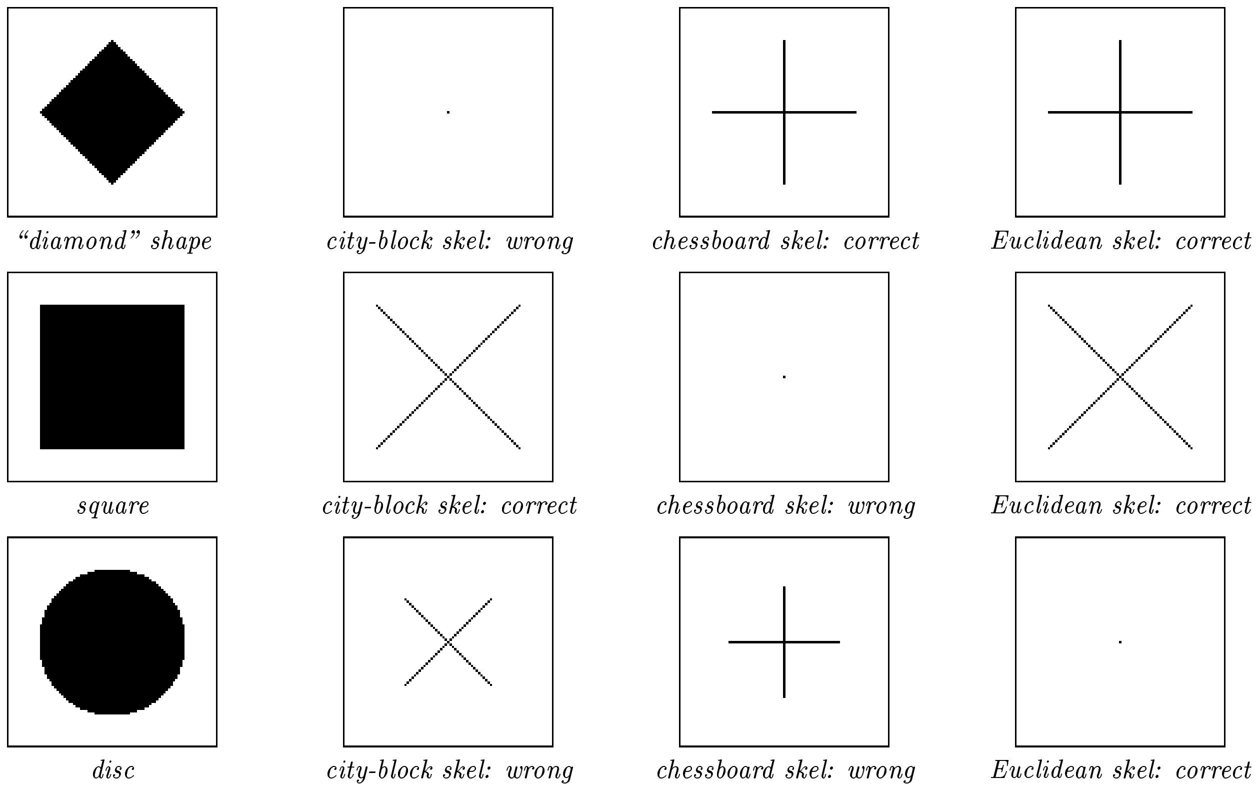


Figure 2 : *Examples of skeletons computed in different metrics for simple shapes.*

In this paper, we propose a new algorithm for computing *Euclidean* skeletons in binary images. Contrary to the existing Euclidean skeleton approaches, our technique is fast and flexible and yields accurate skeletons. It relies on both the above definitions and is derived from an algorithm published in [17, 18]. Furthermore, the distance function it is based on allows to accurately extract conditional bisectors as well (see § 4). Several examples are shown.

The following concepts and notations are used throughout the paper:

– X: set of "on" pixels of binary image I to skeletonize.

– 4- and 8-connectivity: refers to the number of neighbors of each pixel in a square grid. We also use the terminology 4- and 8-neighborhood. For a given grid, the directions in which the neighbors of a pixel are located are called *principal directions.*
– 4- and 8-connected distances: metrics respectively associated with 4- and 8-connected grids (i.e., city-block and chessboard distances).
– $dist_X$: distance function of X, assigning to each pixel its distance to the background.
– $\vec{V}_X$ vector image associated with Euclidean distance function of X (see § 3, Fig. 4)
– Local maximum: pixel whose value is greater or equal than that of its neighbors (in 4- or 8-connectivity).
– CMD: Center of Maximal Euclidean Disc.

2 Computation of skeletons in digital binary images

Exhaustively listing all the different skeleton algorithms that can be found in literature would be far beyond the scope of this paper. These techniques can however be decomposed into 4 main categories:

– algorithms based on iterative morphological thinnings,
– algorithms based on the crest-lines of the distance functions,
– algorithms derived from computational geometry,
– algorithms based on contours.

While the algorithms of the first class are too slow on nonspecialized hardware and do not extend to the Euclidean case, the next two techniques are more efficient and may be adapted to produce Euclidean skeletons (see e.g. [9, 2]). However, as explained in the [17, 18], they are cumbersome, have little flexibilility and are not as fast as the contour-based techniques.

Algorithms based on contours are usually the most efficient, since they allow one to only scan each pixel a very small number of times. Among them, the skeletonization technique described in [17, 18] is based on simple concepts and may be tuned to produce many different kinds of skeletons. It consists in the following steps:

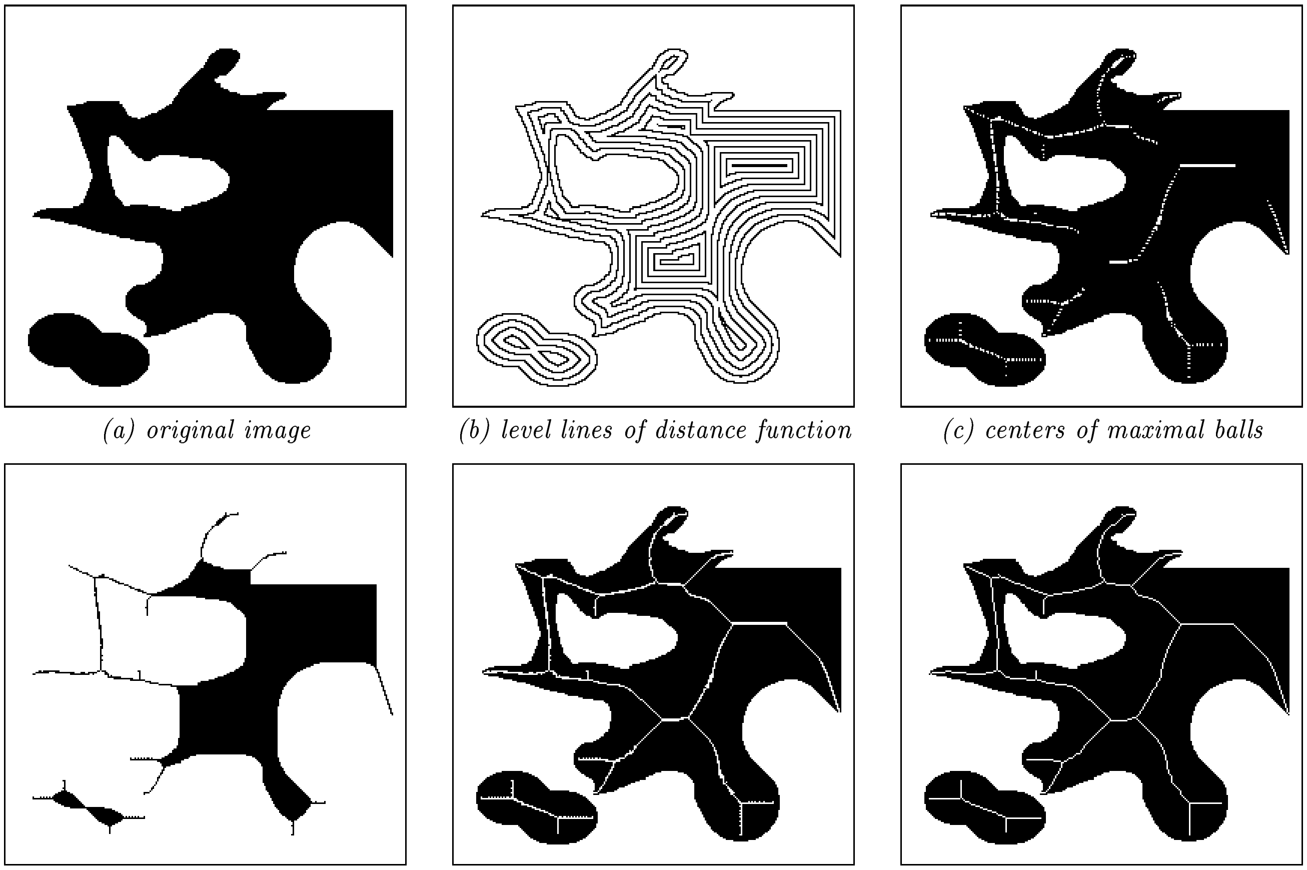
- determine the distance function (in 4-, 6- or 8-connectivity) of set X to skeletonize,
- extract the local maxima of this distance function, which are the centers of the maximal balls for the used (non-Euclidean) distance. A well-known fact is that the set of extracted pixels does not in general preserve the homotopy (connectivity) of the original object.
- starting from the boundaries of the X, progressively remove pixels from this set in a breadth-first scanning, while respecting the following two conditions:
  – none of the center of maximal balls can ever be removed; these pixels are therefore called *anchor points,*
  – the homotopy of the set shall be preserved throughout the process.

This technique is illustrated by Figs. 3a–e. The obtained skeleton may still be of non-unit width in places, but can easily be thinned, thus yielding Fig. 3f. Note that on this example, 8-connected distance was used, so that the upper right corner of the set is not detected by skeletonization.

3 Proposed Euclidean skeleton algorithm

In order or adapt the above algorithm to the Euclidean case, two problems need to be addressed:

- reliable detection of the centers of the Euclidean maximal discs (hereafter referred to as CMD)
- Euclidean propagation of the firefront to connect the CMDs into an accurate Euclidean skeleton



(a) original image *(b) level lines of distance function* *(c) centers of maximal balls*

(d) propagation of the firefront *(e) final exact skeleton* *(f) thinned skeleton*

Figure 3 : *Steps involved in the construction of an 8-connected skeleton in chessboard metrics (see [18]).*

The Euclidean distance function is at the basis of most techniques to extract CMDs. Numerous algorithm have been proposed for the accurate computation of this transform: Danielsson's algorithm [5] is certainly the most well-known, but only yields approximate results. The algorithm proposed in [19] provides an exact and efficient computation of Euclidean distance functions, but is rather complicated to implement. The technique most recently proposed by Paglerioni [11] seems to combine both advantages of [5] and [19] into a simple and exact algorithm.

No matter what algorithm is chosen, a required intermediate step is the extraction of what we refer to in the following as the *vector image* $\vec{V}_X$: given the Euclidean distance function $dist_X$ of a X, $\vec{V}_X$ is such that $\forall p \in X, p - \vec{V}_X(p)$ is (one of) the background pixel(s) closest to p. Obviously, $dist_X = ||\vec{V}_I||$. The images V_X^x and V_X^y such that $\vec{V}_X = (V_X^x, V_X^y)$ are called component images of the distance function. (see Fig. 4). $\vec{V}_X$ is at the basis of our CMD detection.

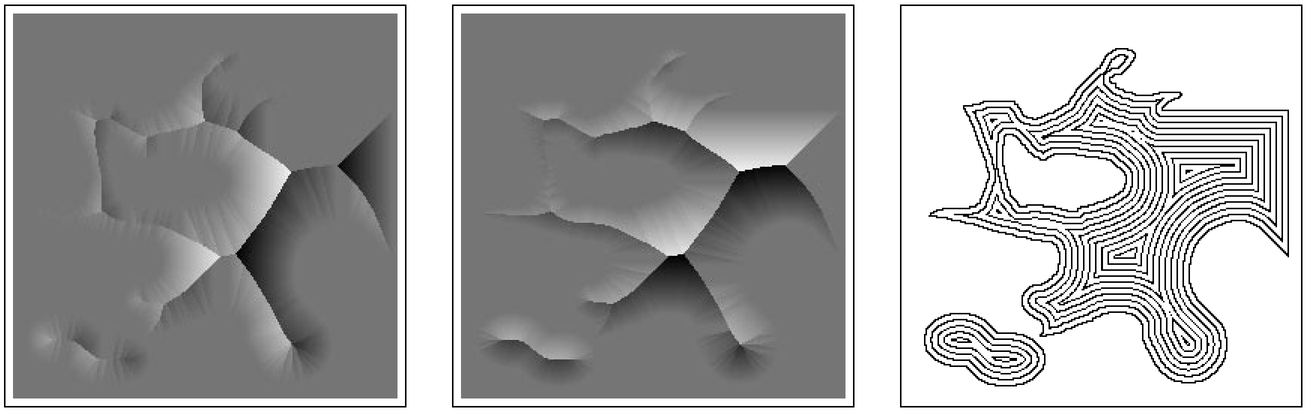

(a) x component image *(b) y component image* *(c) level lines of distance function*

Figure 4 : Components of the vector image and Euclidean distance function.

In this section, we first review some methods to extract CMDs in discrete images and illustrate their gross inaccuracies (sections 3.1 and 3.2) This leads us to propose in § 3.3 a much improved characterization of CMDs. Finally, the Euclidean grassfire propagation is implemented and allows us to easily extract a large number of different skeletons.

3.1 Extraction of CMDs: methods based on local maxima

As mentioned earlier, when using, e.g., 4- or 8-connected distances, the centers of the maximal balls can be directly extracted as the local maxima of the distance function. However, this is far from being true in the Euclidean case, contrary to one of the methods suggested in [9] by F. Meyer. The intuitive reason for this is that a maximal Euclidean disc is not necessarily centered on a grid pixel: the local maxima of a Euclidean distance function are therefore a sparse subset of the set of CMDs. This is illustrated by Fig. 5, where the set of CMDs extracted as the local maxima of the Euclidean distance function of the "diamond" is reduced to a single pixel no matter what connectivity is used.

In order to extract more CMDs, it has been proposed to compute the local maxima of an approximate Euclidean distance function, obtained by taking the integer part of the true Euclidean distance function. This usually produces far too many CMDs when local maxima are computed in 4-connectivity. In 8-connectivity however, some configurations are still missed, as illustrated by Fig. 10.

3.2 Extraction of CMDs: upstream interpolation methods

In [9], a very different technique is proposed to extract the CMDs. In the continuous case, it can be stated as follows: Let X be an open set in the Euclidean space, $dist_X$ the Euclidean distance function associated with X. The *upstream*

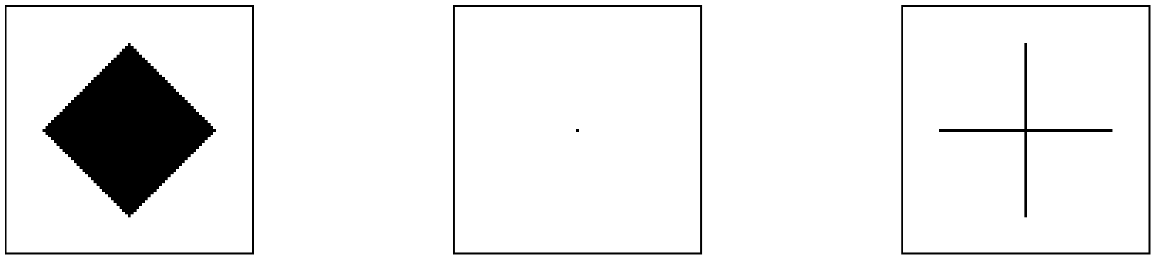


Figure 5 : *(a) "diamond" shape; (b) local maxima of Euclidean distance function; (c) desired centers of the Euclidean maximal discs.*

of a point x inside X is the set of points $y \in X$ that satisfy:

$$dist_X(y) = dist_X(x) + d(x,y) \tag{3}$$

(see [14, Chap. 11 & 12]). In a symmetric fashion, we say that y belongs to the upstream of x and that x belongs to the downstream of y. If x does not belong to the skeleton, then the upstream and the downstream of x form a unique line segment s which goes from the boundary of X to the skeleton. The maximal ball whose center is located at the intersection between this segment and the skeleton has s as one of its radii (see Fig. 6). One can show that *a point has no upstream if and only if it is a CMD.*

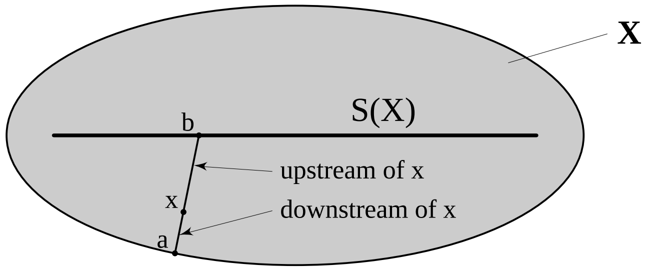


Figure 6 : *Upstream and downstream of a pixel x on the distance function*

As can be expected, transposing these notions from a continous to a discrete framework is not an easy task and approximations have to be made. In a straightforward approach to applying eq. (3), Meyer [9] suggests the following:

- get upstream direction $\vec{V}_X(p)$ of each pixel p
- extract grid principal direction $\vec{V}^d_X(p)$ that is closest to $\vec{V}_X(p)$ and neighbor q of p in this direction.
- compute expected distance function value $dist^E_X(q)$ of q by interpolating from $dist_(p)$ using eq. (3): $dist^E_X(q) = dist_X(p) + ||\overrightarrow{pq}||$, where $||\overrightarrow{pq}|| = 1$ or $\sqrt{2}$ depending on whether q is in the 4-neighborhood of p or only in the 8-neighborhood.
- If $dist_X(q) < dist^E_X(q)$, then eq. (3) is not respected. Pixel p has thus no upstream and therefore it is a CMD.

Unfortunately, this simplistic method does not work properly: since upstream directions almost never fall exactly on one of the 8 principal directions of the grid, q almost never lies exactly in the upstream of p. Its value on the distance function, $dist_X(q)$, is thus rarely as high as the expected one. Therefore, far too many CMDs are extracted by this method, as illustrated Fig. 7.

To get a better understanding of what is happening, let us magnify in Fig. 8 the border of the set shown in Fig. 7. As we see, the actual upstream direction is not one of the 8 principal directions. In this figure, line AB is parallel to the border and the arrow indicates upstream direction. We consider the white pixel p_0 from which the arrow originates: the two highlighted pixels p_1 and p_2 are the neighbors of p such that $\overrightarrow{p_0p_1}$ and $\overrightarrow{p_0p_2}$ are closest to

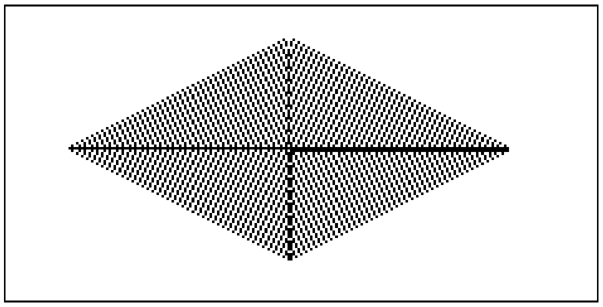


Figure 7: *Example of wrong CMD detection by interpolation. The differences between the right and left sides of set come from the non-unicity of the vector image.*

the upstream direction. The interpolated squared distance function for p_1 is $dist_X^E(p_1)^2 = ||2+1, 0+0|| = 9$, which is greater than $||2,2||^2 = 8$, the square of the actual distance function at p_1. Similarly, the interpolated distance function at p_2 is $dist_X^E(p_2)^2 = ||2+1, 0+1|| = 10$, greater than $||3,0||^2 = 9$. Therefore, pixel p_0 is considered a CMD, which is incorrect.

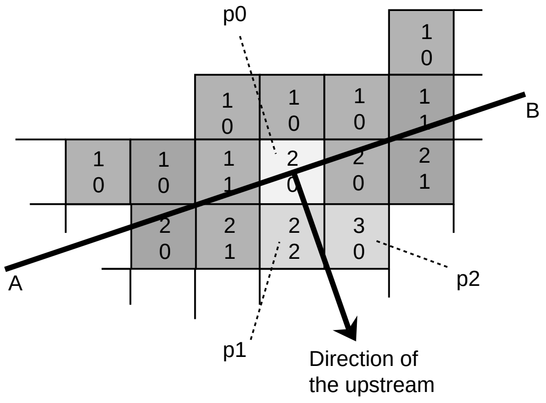


Figure 8: *Magnification of the boundary of the set in Fig. 7. Each pixel p (square) contains the components of vector image $\vec{V}_X$.*

3.3 Extraction of the centers of the Euclidean maximal discs: proposed method

Our approach is also based on the upstream characterization of CMDs. However, contrary to the above method, we impose stricter conditions to decide that a pixel is a CMD. The problem of the previously described method arises from the fact that, to decide whether or not a pixel p is a CMD, we look at the value of the neighbor q of p which is in the grid direction $\vec{V}_X^d(p)$ *closest* to the upstream direction $\vec{V}_X(p)$. Along direction $\vec{V}_X^d(p)$ however, the slope of the distance function at p has no reason to be equal to 1.

Let $s_X(p)$ denote the slope of the distance function in the *discrete* direction of $\vec{V}_X^d(p)$ and let $\theta_X(p)$ be the angle between vectors $\vec{V}_X^d(p)$ and $\vec{V}_X(p)$ (these notions are illustrated on Fig. 9. Obviously, $s_X(p) \leq 1$. For our CMD extraction, we approximate $s_X(p)$ by:

$$s_X(p) \approx \cos(\theta_X(p)) \tag{4}$$

To summarize, we declare p a center of Euclidean maximal disc if and only if the following condition is required:

$$dist_X(q) < dist_X(p) + \cos(\theta(p)) \times ||\vec{pq}||. \tag{5}$$

The implementation of this idea is straightforward and provides very reliable detection of centers of Euclidean maximal discs in discrete images. Note that for pixels p belonging to the boundary of X, upstream direction $\vec{V}_X(p)$

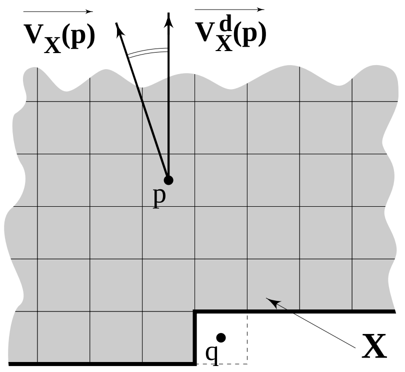


Figure 9 : *Vectors $\vec{V}_X(p)$ and $\vec{V}_X^d(p)$ associated with pixel $p \in X$. $\theta_X(p)$ is the angle between these two vectors, and $\vec{qp} = \vec{V}_X(p)$.*

is highly undetermined. This can thus lead to wrong detection of CMDs of radius 0. The remedy we used is to apply a special treatment to these pixel: look-up tables of pixel configurations are used in the algorithm to decide whether or not they are CMDs. Even for pixels located *inside* set X, similar problems can occurr, but to a much lesser extent and with almost no consequence on the resulting detections.

Remarks:

- We cannot guarantee that *all* the CMDs are detected by this method, which is after all based on discrete approximations. However, practical results show that the obtained set is very complete.
- This method is rather straightforward to implement. However, it can be simplified even further by replacing the $\cos(\theta(p)) \times \|\vec{pq}\|$ in eq. (5) by 0. A pixel p is declared CMD if and only if:

$$dist_X(p + \vec{V}_X^d(p)) < dist_X(p)$$

  This latter equation results in even less CMDs, but in practice, the meaningful ones are still preserved.

This CMD extraction method is illustrated by Fig. 10 and opposed to other techniques. One can see that our method is the only one which detects the upper right part of the set (square angle) without producing any misdetections.

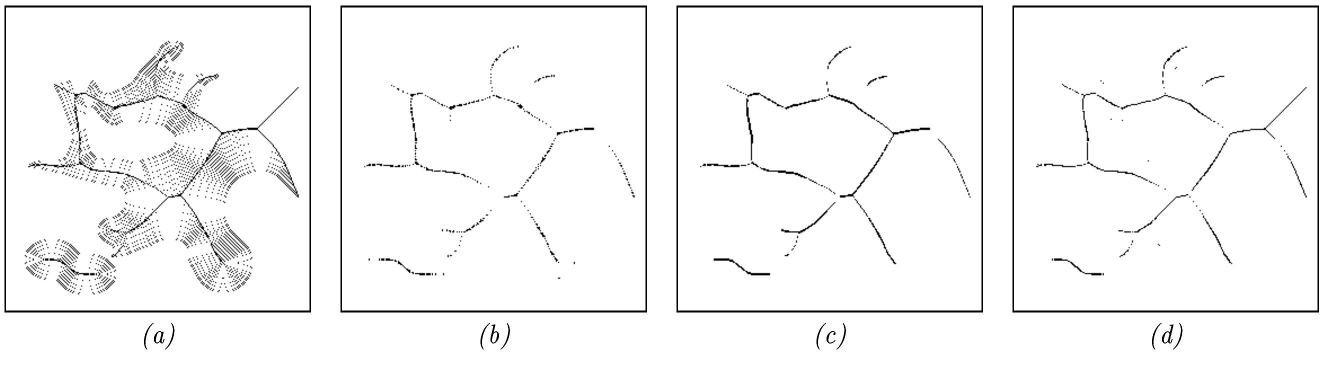


Figure 10 : *Compared CMD extraction: (a)–(b) local maxima of the integer part of the Euclidean distance function, in 4- and 8-connectivity; (c) Meyer's interpolation-based method; (d) proposed method.*

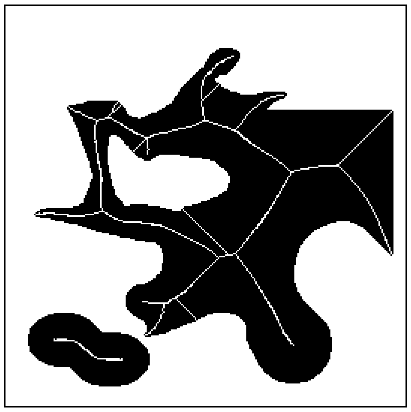
3.4 Euclidean propagation of the grassfire

Once the CMDs are extracted, we proceed to the firefront propagation in a manner very similar to that proposed in [17, 18]. Let us call the extracted CMDs *anchor points.* Starting from the boundary pixels, we progressively peel the set under study until stability while respecting the following constraints:

- CMDs can never be removed: these points are referred to as *anchor points,* since the firefront will anchor itself to them
- the *homotopy* of the set remains unchanged throughout the process. In other words, the set can never be broken at any point.

The second condition can be imposed either for 4-connectivity or for 8-connectivity by using different pixel configuration look-up tables [18]. The algorithm can thus produce 4-connected and 8-connected Euclidean skeletons equally well.

The 8-connected Euclidean skeleton thus obtained for the shape of Fig. 3a using the anchor points of Fig. 10d is shown in Fig. 11a. Again, this skeleton can be thinned, resulting in the unit pixel thickness Euclidean skeleton shown in Fig. 11b.


(a) Euclidean skeleton

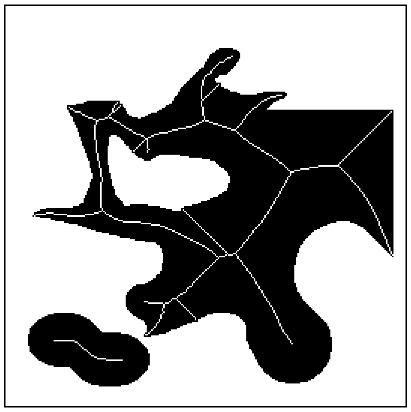

(b) thinned Euclidean skeleton

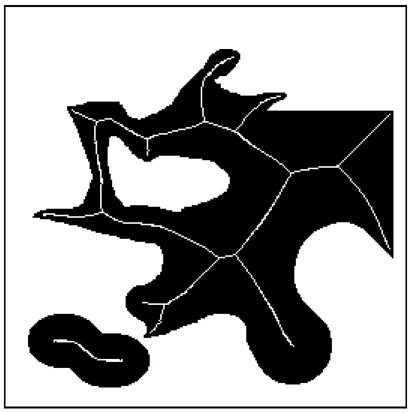
Figure 11 : *8-connected Euclidean skeletons.*

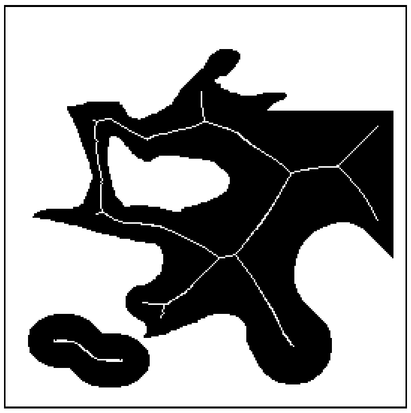
As explained in [17, 18], one of the beauties of the present skeletonization scheme is that any set of points can a priori be used as set of anchor points. For example, one can impose that the minimal radius of the Euclidean maximal discs considered is larger than a given threshold λ. Fewer anchor points are thus used in the process, resulting in a Euclidean skeleton with fewer branches called skeleton of *order* λ. The extremities of the branches of such a skeleton are located on centers of maximal discs with radius at least equal to λ. Similarly, *Euclidean minimal skeletons* can be extracted by taking as anchor points the set of the *regional maxima* [14] of the Euclidean distance function, also called *ultimate erosion.* These various skeletons are illustrated by Fig. 12

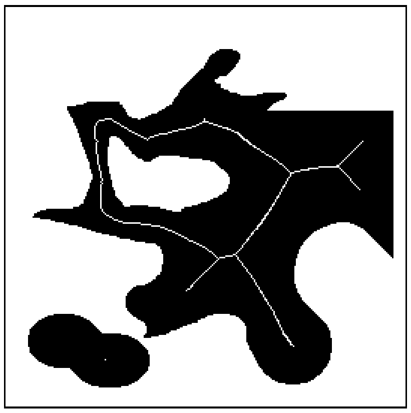
4 Conditional bisector

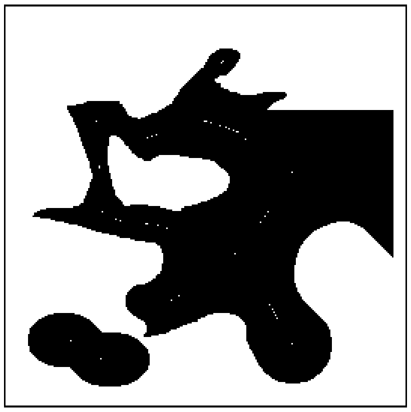
Our Euclidean distance approach also allows us to derive subsets of the skeleton called *conditional bisectors.* The conditional bisector of angular parameter θ was introduced by Meyer in 1979 [8]. It is based on the remark that, when the skeleton is constructed from a firefront propagating inside set X at uniform unit speed 1, the fire propagates along the skeleton at a speed greater or equal to 1. Meyer's definition is the following:

Definition 4.1 *The θ-conditional bisector of a set X is the set of the skeleton pixels where the fire travels at a speed greater or equal to* $1/\sin(\theta)$.


(a) skeleton of order 2


(b) skeleton of order 10


(c) skeleton of order 20


(d) ultimate erosion: anchor points for minimal skeleton

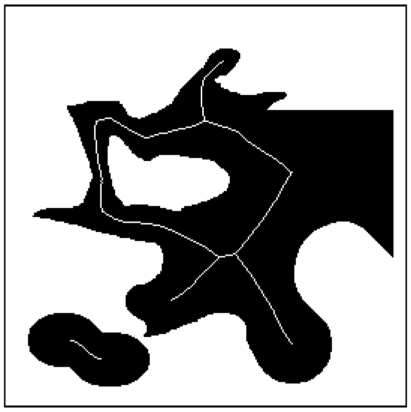

(e) minimal skeleton

Figure 12 : A variety of Euclidean skeletons.

From this definition, one can derive an approximate analytic formula for the conditional bisector in the discrete case, itself derived from Lantuéjoul's skeleton formula [7]. Let us consider for example the 8-connected distance and denote S the elementary square (9 pixels) of the grid. Also denote nS the square of radius n and γ_S the morphological opening with respect to S. Lantuéjoul's formula describes the set $CMS(X)$ of the centers of the maximal squares of set X in terms of erosions and openings:

$$CMS(X) = \bigcup_{n=0}^{+\infty} (X \ominus nS) \setminus \gamma_S(X \ominus nS). \tag{6}$$

The discrete $1/k$-conditional bisector, $k > 0$, is then given by:

$$CB_{1/k}(X) = \bigcup_{n=0}^{+\infty} (X \ominus nS) \setminus ((X \ominus (n+1)S) \oplus kS). \tag{7}$$

Note that setting $k = 1$ in this equation gives back $CMS(X)$.

From eq. (7), one can easily derive an algorithm for computing conditional bisectors of parameters $1/k$. However, this algorithm is clearly very slow. Furthermore, it provides poorly connected bisectors, is based on non-Euclidean metrics and only works for discrete values of k. These are some of the reasons why conditional bisectors have not been used very often yet for practical applications.

To introduce our algorithm, we start from a very different point of view. Recall that the speed $s_X(p)$ of the fire along the skeleton at point p is related to the angle 2θ between the firefronts that met at p by $s_X(p) = 1/\sin(\theta)$ (see Fig. 13). The following proposition thus holds:

Proposition 4.2 *The θ-conditional bisector of X is the set of the skeleton pixels p such that the angle between any two firefronts having met at p is smaller or equal to 2θ.*

For every pixel p of the skeleton—in fact, for every pixel of X—the firefront propagation direction equals the upstream direction given by vector $\vec{V}_X(p)/\|\vec{V}_X(p)\|$ (see § 3) and is perpendicular to the firefront. Thus, for every skeleton pixel p, we can straigtforwardly compute

$$\frac{1}{2}\left[\pi - \max\{\mathit{angle}(\vec{V}_X(p), \vec{V}_X(q)), q \text{ neighbor of } p\}\right],$$

Conditional bisectors of any kind are therefore accurately and rapidly determined from our Euclidean skeleton and the vector image $\vec{V}_X$. Examples of conditional bisectors of different parameters θ are shown on Fig. 14. Notice that as θ decreases, only the increasingly "sharp" structures remain detected by the bisector.

A very interesting outcome of this approach is that with every pixel of the skeleton, we directly associate the angle between the firefronts that met at this pixel. A new transformation which we called *bisector function,* is thus introduced:

Definition 4.3 *The bisector function associates with every point p of the skeleton the angle between the firefronts that met at p.*

Obviously, thresholding the bisector function between θ and 0 yields the conditional bisector of parameter $\theta/2$. An example of bisector function is shown in Fig. 15.

The interest of the bisector function obvious: first, it allows the real-time selection of a θ-bisector particularly suited to a given problem by simple thresholding. Moreover, gray-scale filters of various kinds can be applied to it prior to this thresholding stage. One can also imagine extracting the maxima and/or minima of this function. Its study would go beyond the scope of this paper and will constitute the topic of future publications.

Let us conclude this section by an example of application ilustrating the use of conditional bisectors for binary image segmentation. Fig. 16a is a binary image of a cross section of vitreous fibers. These fibers are overlapping and need to be separated. A very acceptable marking of these fibers was proposed in [16] (Fig. 16b), resulting in a first

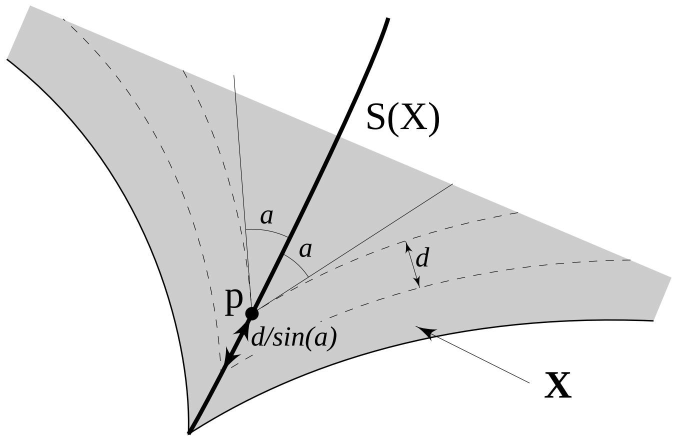
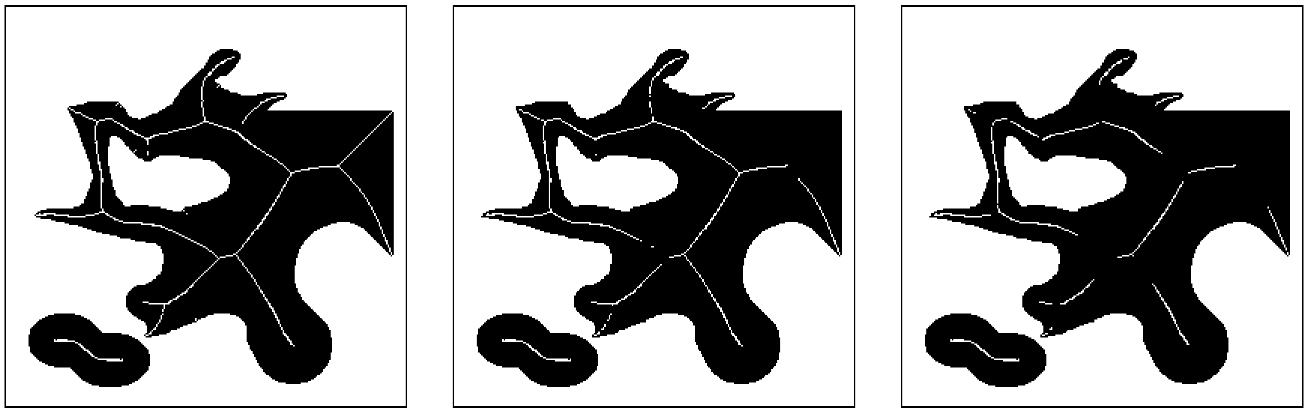


Figure 13 : *When two firefronts meet at p with angle $2a$, the speed of the firefront along the skeleton at p equals $1/\sin(a)$.*



(a) $\theta \leq \frac{\pi}{3}$ *(b)* $\theta \leq \frac{\pi}{4}$ *(c)* $\theta \leq \frac{\pi}{6}$

Figure 14 : *Euclidean conditional bisectors for different angular parameters θ.*

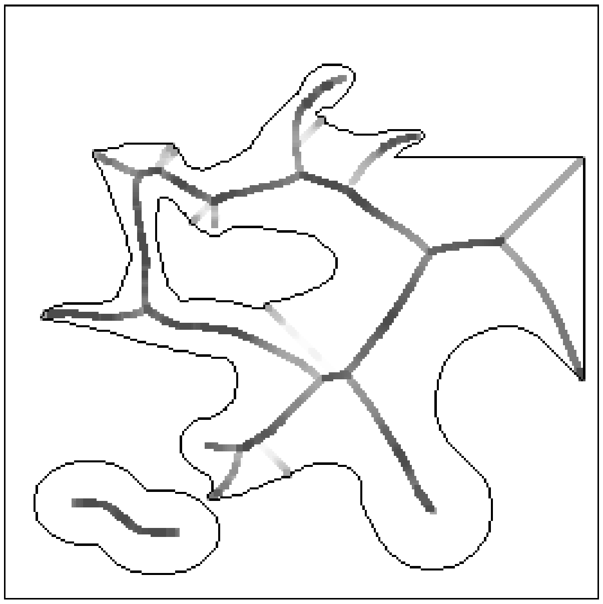


Figure 15 : *The bisector function (dilated for clarity): the darker the pixel, the larger its value.*

watershed segmentation of Fig. 16a (see Fig. 16c)—For watershed segmentation, refer to [10, 17, 20]. However, since some very overlapping particles were not marked, they are not separated in Fig. 16c (these particles are pointed at by arrows). In order ot mark them, we use a $\pi/6$-conditional bisector (Fig. 16d). Not only does it mark these fibers, it also marks some "necks" between fibers. The latter are eliminated as crossing the separating lines of Fig. 16c (see Fig. 16e). Now, since some of the resulting marker are still slightly disconnected, we use a connection technique already described in [6]: Fig. 16e is dilated by an elementary disc, resulting in Fig. 16f. The latter image is then skeletonized using Fig. 16e as set of anchor points (see § 3). This results in Fig. 16g. Adding to this image the markers of Fig. 16b yields the final marker-image shown in Fig. 16h. At this stage, a new watershed segmentation provides the (almost) perfect result of Fig. 16i.

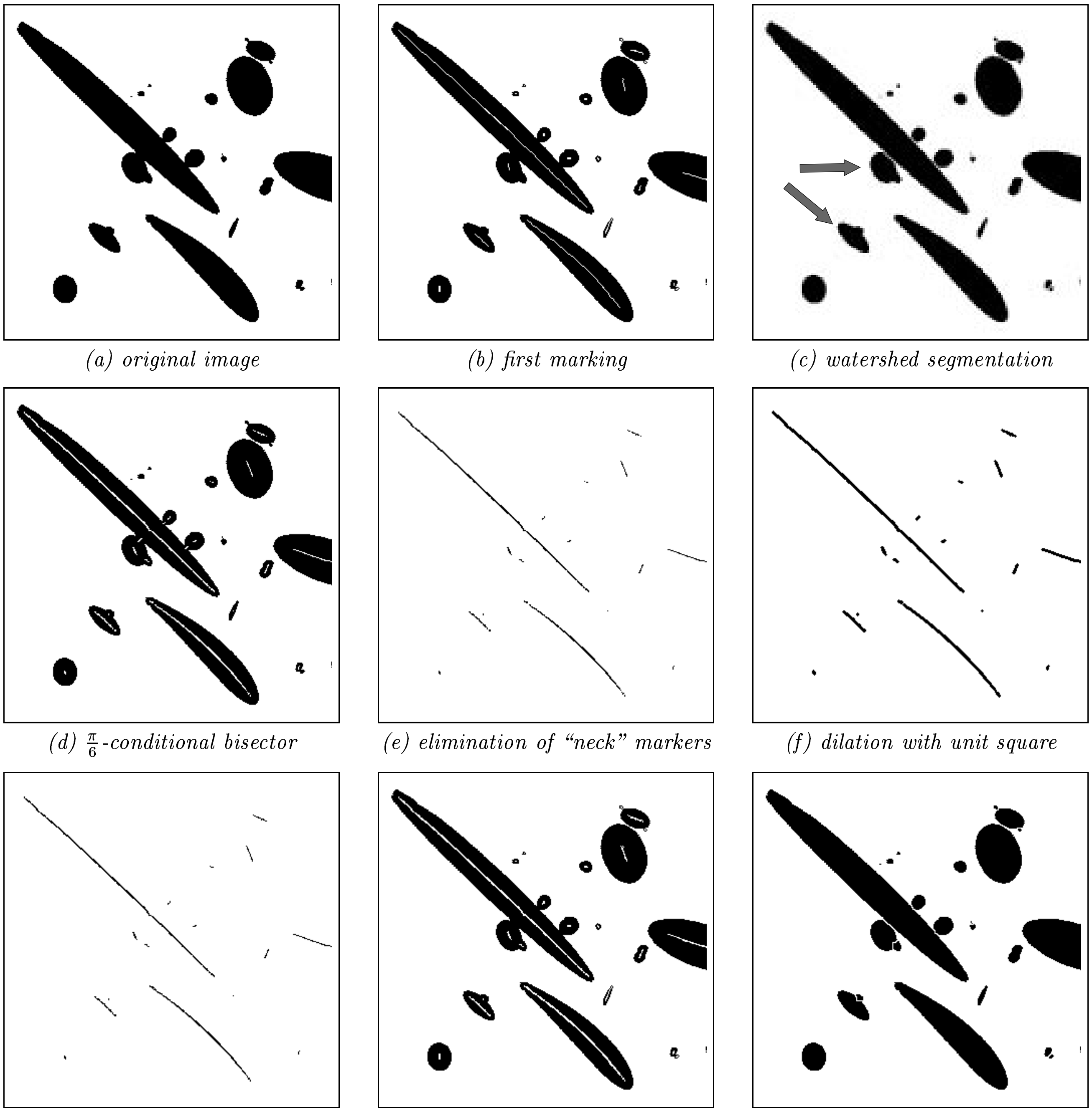
5 Conclusion

Commonly used skeletons are computed according to metrics which can be 4-connected, 8-connected, octagonal, hexagonal, etc. We showed that these skeletons either fail to detect obvious skeleton branches, or generate extraneous ones. Alternatively, skeleton computed by means of iterative homotopic thinnings yield even worse and more unpredictable results. On the contrary, the discrete Euclidean skeleton approach proposed in this paper produces beautifully accurate results, which do not vary with the orientation of the set to skeletonize with respect to the grid. This Euclidean skeleton is therefore a *better shape descriptor*

In addition, the algorithm described in this paper is very efficient, since it only requires a very small number of image scans. On a Sun SparcStation 2, it typically produces a Euclidean skeleton in a couple of seconds for a 256×256 binary image. We illustrated its ability to produce a vast variety of Euclidean skeletons, like minimal skeletons, smooth skeletons, geodesic skeletons, etc. Lastly, the algorithm produces 4-, 6- or 8-connected Euclidean skeletons equally easily.

We also introduced a new transformation, the *bisector function,* whose determination is straightforward from our Euclidean skeleton algorithm. It associates with each pixel of the skeleton the "level of sharpness" of the boudary part(s) having caused this skeleton pixel and can be seen as the "pile" of the conditional bisectors for all possible angular parameters.

We illustrated the use of these skeleton-like tools on a binary segmentation problem. It is our hope that, thanks to the algorithms described in this paper, Euclidean skeletons and conditional bisectors will become increasingly



(a) original image *(b) first marking* *(c) watershed segmentation*

(d) $\frac{\pi}{6}$-conditional bisector *(e) elimination of "neck" markers* *(f) dilation with unit square*

(g) connection of markers in (e) *(h) final markers* *(i) watershed segmentation*

Figure 16 : Use of conditional bisector for the segmentation of binary images of glass fibers.

popular image analysis tools and serve as many purposes as possible.

6 References

1. H. Blum. An associative machine for dealing with the visual field and some of its biological implications. In E. Bernard and M. Kare, editors, *Biological Prototypes and Synthetic Systems*, pages 244–260. Plenum Press, New York, 1962. Proc. second Annual Bionics Symposium, Cornell University, 1961.
2. F. L. Bookstein. The line skeleton. *Comp. Graphics and Image Processing*, 11:123–137, 1979.
3. G. Borgefors. Distance transformations in digital images. *Comp. Vis., Graphics and Image Processing*, 34:334–371, 1986.
4. L. Calabi and W. Harnett. Shape recognition, prairie fires, convex deficiencies and skeletons. Technical Report 1, Parke Math. Lab. Inc., One River Road, Carlisle MA, 1966.
5. P. Danielsson. Euclidean distance mapping. *Comp. Graphics and Image Processing*, 14:227–248, 1980.
6. G. G. Gordon and L. Vincent. Application of morphology to feature extraction for face recognition. In *SPIE/SPSE Vol. 1658, Nonlinear Image Processing III*, pages 151–164, San Jose CA, Feb. 1992.
7. C. Lantuéjoul. *La Squelettisation et son Application aux Mesures Topologiques des Mosaiques Polycristallines.* PhD thesis, Ecole des Mines, Paris, 1978.
8. F. Meyer. *Cytologie Quantitative et Morphologie Mathématique.* PhD thesis, Ecole des Mines, Paris, 1979.
9. F. Meyer. Digital Euclidean skeletons. In *SPIE Vol. 1360, Visual Communications and Image Processing*, pages 251–262, Lausanne, Switzerland, Oct. 1990.
10. F. Meyer and S. Beucher. Morphological segmentation. *Journal of Visual Communication and Image Representation*, 1:21–46, Sept. 1990.
11. D. Paglerioni. Distance transforms: Properties and machine vision applications. *Comp. Vis., Graph. Im. Proc.: Graphical Models and Image Processing*, 54(1):56–74, Jan. 1992.
12. A. Rosenfeld and J. Pfaltz. Distance functions on digital pictures. *Pattern Recognition*, 1:33–61, 1968.
13. J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, London, 1982.
14. J. Serra, editor. *Image Analysis and Mathematical Morphology, Volume 2: Theoretical Advances.* Academic Press, London, 1988.
15. P. Soille. Spatial distribution from contour lines: An efficient methodology based on distance transformations. *Journal of Visual Communication and Image Representation*, 2(2):138–150, 1991.
16. H. Talbot. Binary image segmentation using weighted skeletons. In *SPIE Vol. 1769, Image Algebra and Morphological Image Processing III*, pages 393–403, San Diego, CA, 1992.
17. L. Vincent. *Algorithmes Morphologiques à Base de Files d'Attente et de Lacets: Extension aux Graphes.* PhD thesis, Ecole des Mines, Paris, May 1990.
18. L. Vincent. Efficient computation of various types of skeletons. In *SPIE Vol. 1445, Medical Imaging V*, pages 297–311, San Jose, CA, 1991.
19. L. Vincent. Exact euclidean distance function by chain propagations. In *IEEE Int. Computer Vision and Pattern Recog. Conference*, pages 520–525, Maui, HI, June 1991.
20. L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Machine Intell.*, 13(6):583–598, June 1991.