Recursive implementation of erosions and dilations along discrete lines at arbitrary angles

Pierre Soille, Edmond J. Breen, and Ronald Jones

Abstract

Van Herk has shown that the erosion/dilation operator with a linear structuring element of an arbitrary length can be implemented in only 3 min/max operations per pixel. In this paper, the algorithm is generalized to erosions and dilations along discrete lines at arbitrary angles. We also address the padding problem; so that the operation can be performed in place without copying the pixels to and from an intermediate buffer. Applications to image filtering and to radial decompositions of discs are presented.

Keywords. Mathematical morphology, image filtering, algorithms, recursivity, line and periodic structuring elements, radial decompositions.

Name and address of corresponding author:

Dr. Pierre Soille
LGI2P
Ecole des Mines d'Alès
Parc scientifique Georges Besse
F–30000 Nîmes
France

Ph.: int+33- 66 38 70 22 Fax: int+33- 66 38 70 74 Email: soille@eerie.fr

Pierre Soille is with the Laboratoire de Génie Informatique et d'Ingénierie de Production of the Ecole des Mines d'Alès, Parc scientifique Georges Besse, F–30000 Nîmes, France.
Edmond Breen and Ronald Jones are with the CSIRO Division of Mathematics and Statistics, Institute of Information Science and Engineering, Locked bag 17, North Ryde NSW 2113, Australia.

I. Introduction

Erosion and dilation form the basis of all mathematical morphology operators [11], [7]. Therefore, there has been a considerable amount of literature reporting techniques that enhance the speed of these fundamental operators [17], [9], [8], [4], [16], [5], [10], [14], [6]. Van Herk [15] has shown how to perform grey-scale erosions and dilations with a linear structuring element (SE) of arbitrary length using only 3 min/max operations per pixel. However, van Herk's algorithm only treats horizontal, vertical, and diagonal structuring elements. This is a severe limitation, it does not permit the extraction of linear image structures with arbitrary orientations. In this paper, we address this limitation and propose an extension of van Herk's algorithm for erosions and dilations along lines at arbitrary angles.

The paper is organized as follows. Section II summarizes van Herk's algorithm. Section III introduces our algorithm for performing erosions and dilations along lines at arbitrary angles. Section IV discusses the algorithm from both a theoretical and a practical point of view. Before concluding, Section V presents applications of the algorithm to image filtering and to the generation of circular structuring elements from radial decompositions.

II. The algorithm of van Herk [15]

Erosion $\ominus$ and dilation $\oplus$ of a grey-scale image or function f by a flat SE B may be defined as [7]: $(f \ominus B)(x) = \min_{z \in B} \ f(x+z)$ and $(f \oplus B)(x) = \max_{z \in B} \ f(x-z)$. Traditionally, erosions and dilations are implemented on a grey-scale image f by computing either the local minimum or the local maximum around each point x in the image. Van Herk has proposed a much more efficient procedure for computing elementary morphological operations with horizontal and vertical SE's. In his implementation, a 1–D input array f of length nx is divided into blocks of size k, where k is the length of the SE in number of pixels. The elements of f are indexed by indices running from 0 to $nx-1$. It is also assumed that k is of odd extent and that nx is a multiple of k, i.e., $nx = mk$. Two temporary buffers g and h of length nx are also required.

In the case of dilation, the maximum is taken recursively inside the blocks in both the right and left directions. The results are stored in buffers called g and h. For example, in the right hand direction, the results are stored in buffer g. The algorithm is initialized for each block by setting $g(x)$, where x is the coordinate on the right of the block boundary, to $f(x)$. The result for $g(x+1)$ is then given by the maximum of $f(x+1)$ and $g(x)$. The result at $g(x+2)$ is given by the maximum of $f(x+2)$ and $g(x+1)$, and so on up to the block boundary. In such a way, the result is generated recursively through each block. The same procedure is used in the left direction

to construct the buffer h. When both g and h have been constructed, the result for the dilation r at any coordinate x is given by considering the maximum value between g at position $x + k/2$ and h at position $x - k/2$. This recursive dilation algorithm is summarized below (unless otherwise stated, all divisions are integer divisions in this paper):

$$
\begin{aligned}
g(x) &= \begin{cases} f(x) \quad \text{if } x = 0, k, \ldots, (m-1)k, & /\star \text{ start block positions } \star/ \\ \max[g(x-1), f(x)] \quad \text{otherwise.} & /\star \text{ recursive maximum from left to right } \star/ \end{cases} \\
h(x) &= \begin{cases} f(x) \quad \text{if } x = mk-1, (m-1)k-1, \ldots, k-1, & /\star \text{ start block positions } \star/ \\ \max[h(x+1), f(x)] \quad \text{otherwise.} & /\star \text{ recursive maximum from right to left } \star/ \end{cases} \\
r(x) &= \max[g(x + k/2), h(x - k/2)].
\end{aligned}
$$

The advantage of this algorithm is that it requires only 3 maximum operations per result pixel, is independent of SE length k, and in practice the results r can be stored directly back into f. However, the algorithm is limited by the constraint that the buffer length nx must be a multiple of the SE length k, and by the fact that it does not consider the case when the SE is longer than the input array. In both cases, van Herk proposes to pad the end of the buffer with large negative values. Another limitation of the algorithm is that intrinsically it is only defined for 1–D arrays. It follows that morphological operations on columns of 2–D images stored in raster scan order require successive mappings of each column into a 1–D array and then back into the original image.

In the following section, we introduce a new algorithm that alleviates all these problems, and is suited to erosions and dilations along lines at arbitrary angles.

III. Line erosions and dilations at arbitrary angles

When long–thin features of an image are to be extracted using morphological filters [13], horizontal, vertical and diagonal directions are not sufficient; the number of possible directions increases together with the length of the SE considered (in a square lattice of points, $2n - 2$ directions are defined for a discrete SE of odd extent n in pixels). Therefore, there is a need for an algorithm implementing erosions and dilations with SE's at arbitrary angles.

A. Definition of an array of indices: the p array

Contrary to van Herk's algorithm which deals with 1–D arrays only, we propose to process in place 2–D images f stored as a 1–D array; that is, one row after another ($nlin$ rows by $ncol$ pixels). This is achieved by defining an array of indices for accessing the successive pixels along a given line of the 2–D image. This array is denoted by p. For instance, $p(x) = x + n * ncol$ sets the

index array for dilating the nth row of f and $p(x) = x * ncol + n$ for dilating the nth column. A dilation is performed at any orientation within the 2–D array by simply loading the p array with the appropriate indices. This procedure is described in the following section.

B. Processing lines at arbitrary angles

For the general case, we propose to set the values of p using a line tracing function. This function can be Bresenham's algorithm [2], a function that makes a periodic line, or some other function defined by the user. The slope of the line is defined from two integer numbers dx and dy, where $slope = dy/dx$. Given these numbers, a discrete line with the same slope is traced in the image plane and the p array is initialized according to this line (e.g., if $dx = 1$ and $dy = 0$, $p(i) = i$ for all $i \in \{0, ncol - 1\}$). The recursive maximum filter described in Section III-C performs the dilation along the line indexed by p – the image f being stored as a 1–D array, the value of a point i along the line equals $f(p(i))$. The line is then translated and the procedure is repeated until the whole image is processed. Potentially, there are problems with this approach since there can be overlap between translations of a discrete line. We therefore add the following constraint: discrete lines with angles $\in [-45^o, +45^o]$ must not contain any group of two or more successive pixels in the vertical direction and lines with angles $\in [-90^o, -45^o[\cup]45^o, +90^o]$ must not contain any group of two or more successive pixels in the horizontal direction. Hence, by translating the line in the vertical direction for angles $\in [-45^o, +45^o]$ and in the horizontal direction for angles $\in [-90^o, -45^o[\cup]45^o, +90^o]$, we make sure that there will be no overlap: note that any Bresenham line satisfies this requirement.

The maximal length of the line is $ncol$ pixels for angles $\in [-45^o, +45^o]$ and $nlin$ pixels for angles $\in [-90^o, -45^o[\cup]45^o, 90^o]$. In order to sweep the whole image plane by translating the line in a unique direction, the line must be first drawn from an appropriate image corner. This is illustrated in Fig. 1. For instance, all lines having a slope within the range $[-90^o, -45^o[$ must be drawn from the lower-right corner (positive angles are counted clockwise as positive y image coordinates increase downwards). When translating the line from its original position, the number of pixels falling within the image plane first increases, then remains constant, and finally decreases. If the image size along the translation direction is shorter than the distance between the extremity of the line falling off the image plane and the image plane, there is no constant zone. Both situations are represented in Fig. 2.

Pixels actually falling within the image plane are directly determined after each translation by using a run length coding of the discrete line calculated within the line tracing function. When the

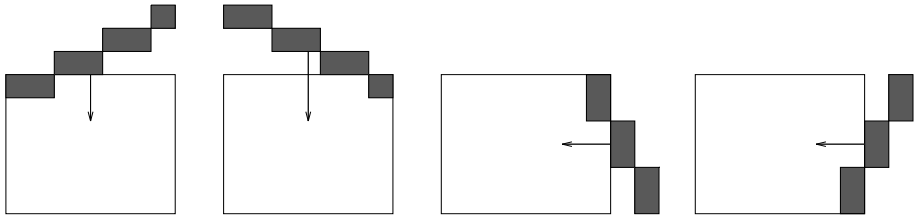


Fig. 1 : Depending on its slope the discrete line is drawn from an appropriate image corner. It can be then translated in a unique direction in order to sweep the whole image plane while avoiding overlapping pixels (the arrows indicate the translation direction).

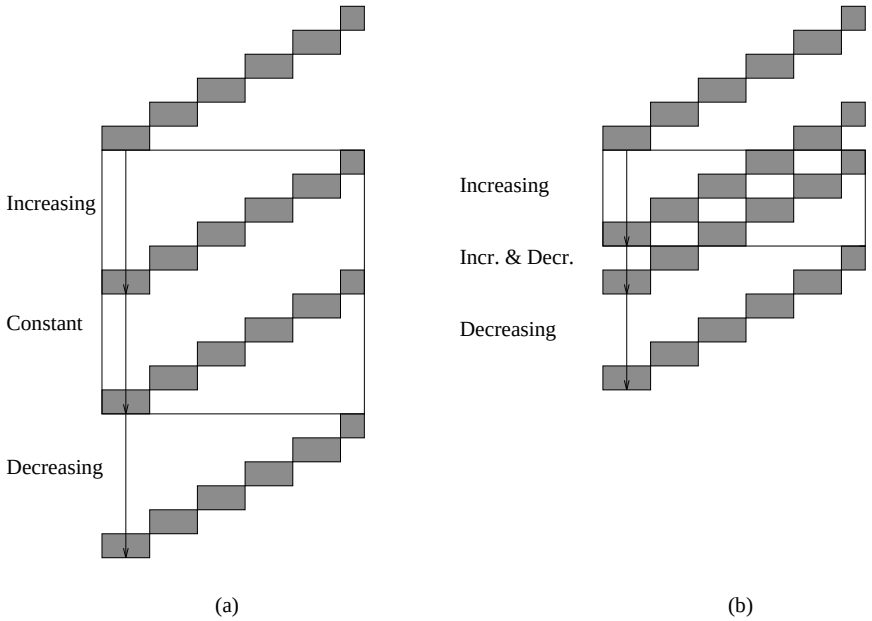


Fig. 2 : (a) When translating the discrete line from its initial position, the number of pixels falling within the image plane successively increases, remains constant, and then decreases. (b) In some cases, the zone where the line is of constant length is replaced by a still increasing line at one end while already decreasing at the other.

line is decreasing at one end (i.e., pixels at the beginning of the line fall off the image plane after a translation), the pointer to the p array must also be updated to avoid pointing outside of the image. This operation is automatic due to the run-length coding of the original line. In practice, we avoid updating the p array by simply incrementing the pointer to the image data f (e.g., pointer to f plus $ncol$ for a translation along the vertical direction). The procedure terminates when no part of the line lies on the image (i.e., once all pixels have been processed).

C. Processing lines of arbitrary length

There are problems using van Herk's recursive procedure for computing dilations along a given line indexed by the p array because the original procedure assumed that the buffer length would be a multiple of the SE size. Figure 3 illustrates two situations where padding would normally be required to fill the buffers. In Fig. 3a, the structuring element lies over the left hand edge of

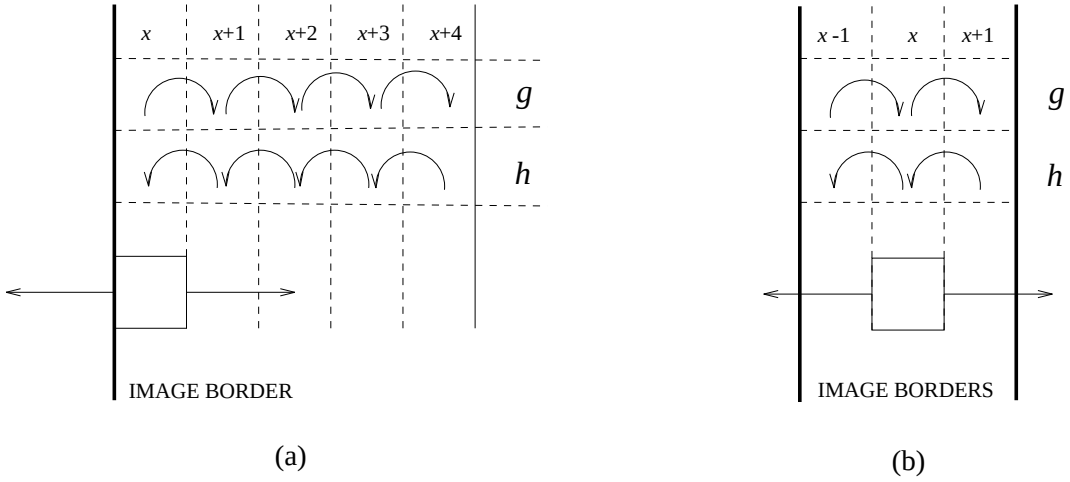


Fig. 3 : Cases where padding is normally required (the width of the SE is 5 pixels). (a) The SE lies off the border of the image. (b) The SE exceeds image at both borders.

the image. If we consider the case of dilation, the result is undefined at position x because the maximum of $g(x + k/2)$ and $h(x - k/2)$ is undefined when $h(x - k/2)$ is undefined. Rather than padding the buffer h with large negative values, it is more direct to set Result$(x) = g(x + k/2)$, as the maximum of $g(x + k/2)$ and a large negative value will always be $g(x + k/2)$. In the case of erosion, the minimum of $g(x + k/2)$ and $h(x - k/2)$ is also undefined when $h(x - k/2)$ is undefined. If $h(x - k/2)$ is defined as a large negative value, say $-N$, then Result$(x) = -N$. However, negative grey-scales are undesirable in the image and we use instead Result$(x) = g(x + k/2)$, which would be the running minimum in the case of erosion. The right hand border is treated in much the same way, only we use the h array instead of g. Figure 3b illustrates the second situation in which padding would normally be required. Here, the structuring element lies over both edges of the image and both $g(x + k/2)$ and $h(x - k/2)$ are undefined. In such case, Result(x) is given the value $g(nx - 1)$, which contains the running maximum in the case of erosion or the running minimum in the case of dilation (note that $h(0)$ could also be used, as it is equivalent to $g(nx - 1)$ in this

situation). The algorithm is summarized for the case of dilation:

$$g(x) = \begin{cases} f(p(x)) & \text{if } x = 0, k, \ldots, (m-1)k, mk, \text{ and } x < nx - 1, \\ \max[g(x-1), f(p(x))] & \text{otherwise.} \end{cases}$$

$$h(x) = \begin{cases} f(p(x)) & \text{if } x = nx - 1, mk - 1, (m-1)k - 1, \ldots, k - 1, \\ \max[h(x+1), f(p(x))] & \text{otherwise.} \end{cases}$$

$$r(p(x)) = \begin{cases} g(nx-1) & \text{if } nx \leq k/2, \\ gh(x) & \text{if } k/2 < nx \leq k, \\ g(x + k/2) & \text{if } x = 0, \ldots, k/2 - 1, \\ h2(x - k/2) & \text{if } x = nx - 1, \ldots, nx - k/2, \\ \max[g(x + k/2), h(x - k/2)] & \text{otherwise.} \end{cases}$$

where

$$gh(x) = \begin{cases} g(x + k/2) & \text{if } x = 0, \ldots, nx - k/2 - 1, \\ g(nx - 1) & \text{if } x = nx - k/2, \ldots, k/2, \\ h(x - k/2) & \text{if } x = k/2 + 1, \ldots, nx - 1. \end{cases}$$

$$h2(x) = \begin{cases} f(p(x)) & \text{if } x = nx - 1, \\ \max[h2(x+1), f(p(x))] & \text{otherwise.} \end{cases}$$

Figure 4 displays the processing time obtained for a dilation with a SE of increasing size with a slope of 27 degrees. Using the new recursive algorithm for lines at arbitrary angles, the time required is independent of the length k of the SE contrary to a $O(k)$ computational complexity for the classical algorithm. Moreover, the whole image plane is processed in place and there are no border effects (for the classical algorithm, not all pixels from the input image would be processed unless the original image had a border added to it). Note that the break even point between the classical and the new recursive implementation for lines at arbitrary angles is located at 5 pixels.

IV. Discussion

A. Actual length or number of pixels?

The extent of the SE is given in number of pixels rather than as an actual length. The actual length varies according to the slope of the SE. In some applications it may be useful to compensate for this effect. Therefore, let k be the actual length of the SE, the pixel width being defined as 1. For horizontal and vertical SE's, the actual length coincides with the extent in number of pixels. The number of pixels, k_α, of a SE of actual length k along a discrete line of slope α is then given

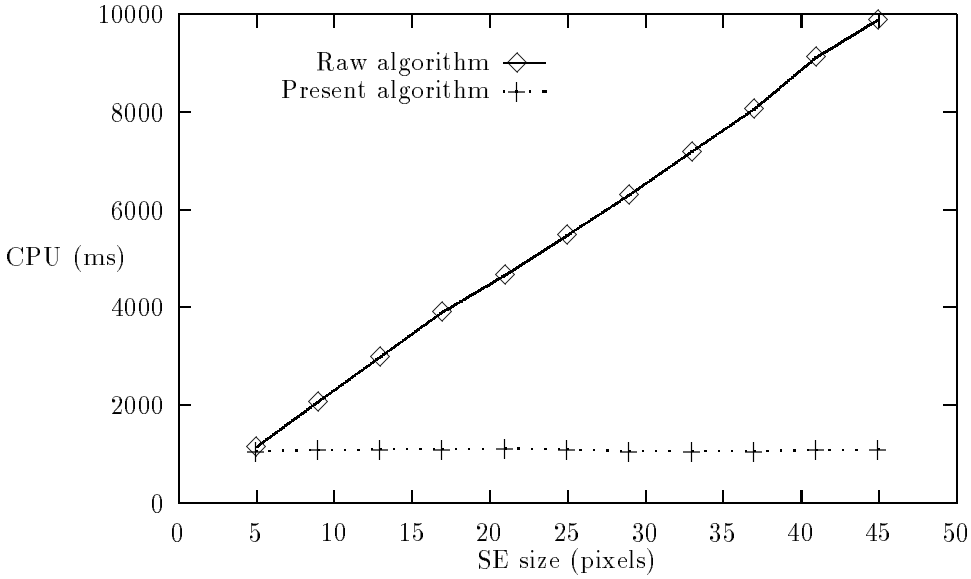


Fig. 4 : Processing time (ms) for a dilation with a SE at 27 degrees ($dx = 2$ and $dy = 1$ versus extent k of the SE in pixels on a 1024×1024 image of a simulated Brownian function using a SPARC-10 *(crosses for the new algorithm and diamonds for the classical algorithm requiring $k - 1$ comparisons per pixel).*

by:

$$k_\alpha = \text{round}(k * 0.5 * \max(|\cos\alpha|, |\sin\alpha|) + 0.5) * 2 + 1, \tag{1}$$

where $|x|$ returns the absolute value of x and $round(x)$ returns the integer part of x.

B. On the choice of the SE length and angle

The proposed algorithm performs erosions/dilations in place and along discrete lines at arbitrary angles. However, it is important to note that the shape of the SE may vary slightly from one pixel to another. Indeed, except for horizontal, vertical, and diagonal directions, a discrete line in the square grid contains 4– as well as 8–connected pixels. Hence, when the SE of length k moves along such a discrete line, its shape varies accordingly to its position along the line. In practice, this is not a major drawback provided that the angle specified is actually defined in the neighbourhood corresponding to the extent of the SE. More precisely, only $2k - 2$ directions are allowed for an odd extent of k pixels; for example, only the four principal directions of the square grid may be used when considering an extent of 3 pixels.

In practice, linear image features are filtered independently of their orientation. This is achieved by performing either a union of openings or an intersection of closings over all possible directions. To compensate for the anisotropy of the square grid, the best approach is to consider all orientations

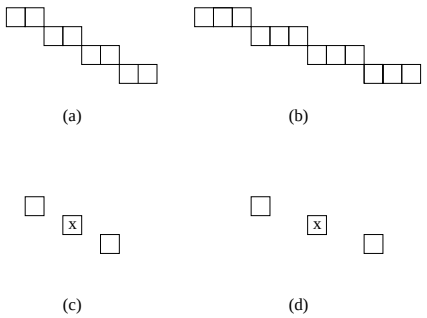


Fig. 5 : Two discrete lines (a) and (b). The corresponding elementary periodic lines have a periodicity λ equal to 2 for line (a) and 3 for line (b). The corresponding translation-invariant SE's are shown in (c) and (d) respectively.

defined from half the circumference of the Bresenham disc [3] drawn with a diameter equivalent to the length of the SE (the disc defines each dx and dy parameters and the corresponding extent is $\max[\text{abs}(dx), \text{abs}(dy)] * 2 + 1$).

C. Translation-invariant implementation

Since the SE is not exactly the same for all image pixels, the new algorithm for erosions/dilations along lines at arbitrary angles corresponds to erosions/dilations with a structuring *function* [12], and it is not translation-invariant. Note however that the openings and closings are still true openings and closings in the sense that they are idempotent and increasing.

To generate erosions and dilations that are translation-invariant, we use a periodic line rather than a Bresenham line. We define a periodic line as a line constructed from points that all occur a constant distance apart along the considered discrete line. Discrete lines with a periodicity of 2 and 3 are shown in Fig. 5. The corresponding elementary SE is defined as the set of three disconnected pixels (the central pixel and the two pixels occurring a constant distance apart). Longer SE's are constructed by adding 2 pixels at a time (each occurring the same constant distance apart). When the periodic SE moves along the corresponding periodic line, its shape remains invariant with the position along the line; it is therefore translation-invariant (see Figs. 5c and 5d).

Concerning the implementation of the algorithm, the current buffer is again partitioned into blocks, but this time each block has a size of λ times the number of pixels in the structuring element. Figure 6 shows an example where the periodicity $\lambda = 2$ and the structuring element k has three points, hence the size of each block is $2 \times 3 = 6$. The algorithm works in exactly the same way; a recursive computation of maxima using two buffers g and h. The only difference is that each maximum is taken between pixels that are a distance λ apart and not between adjacent

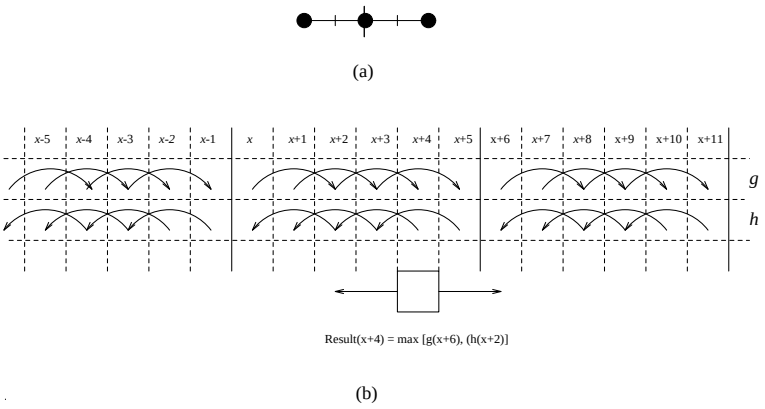


Fig. 6 : Implementation of Morphological Dilation using a modified van Herk algorithm. (a) A structuring element with periodicity $\lambda = 2$. (b) Schematic of the algorithm.

pixels (for which effectively $\lambda = 1$). Hence, the recursive procedure is called *period* times for each periodic line; but the number of pixels to process at each call equals the number of pixels of the line divided by the periodicity. Note also that the number of pixels to process may decrease by one after the first call since the last pixel of the periodic line may fall off the image.

As before, the result is given by the maximum of the two values found at the extremes of the structuring element. In our example, the extremes of the structuring element are at ± 2, and so $\text{Result}(x) = \max[g(x+2), h(x-2)]$.

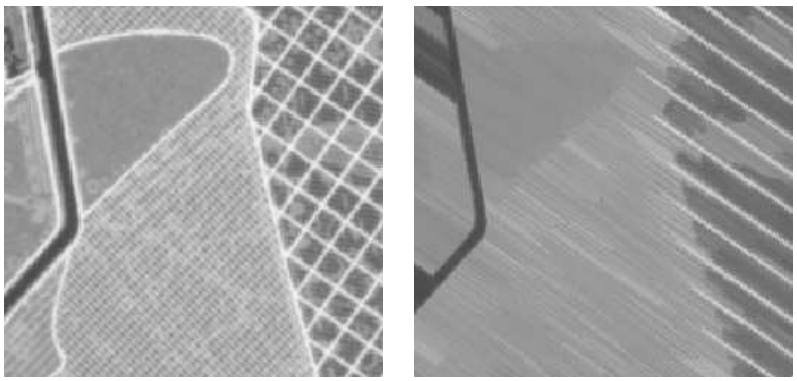
An example is shown in Fig. 6b for the coordinate $x + 4$.

V. Applications

Our algorithm can be used as a primitive for all morphological transformations based on linear erosions/dilations. For example, an opening $f \circ K = (f \ominus K) \oplus K$ can be generated as a cascade of an erosion and a dilation and a closing $f \bullet K = (f \oplus K) \ominus K$ as a cascade of a dilation and an erosion. Figure 7 shows an example. In Fig. 7b is an opening of the image in Fig. 7a by a Bresenham line generated using using a SE of length 21 pixels at angle 35 degrees. In this section, we further illustrate our algorithm's performance and practical application for combinations of linear openings/closings and for radial decomposition of discs.

A. Union/intersection of linear openings/closings

The union (resp. intersection) of a series of linear openings (resp. closings) is used for extracting long–thin features within an image, independently of their orientation. The principle is the following: A SE of a given length is first defined. It should correspond to the length of the shortest bright linear features one would like to preserve. A series of openings is then performed by considering all



(a) (b)

Fig. 7 : An example of an opening by a line at an arbitrary angle. (a) Original image. (b) Opening by a line of length 21 pixels at 35 degrees.

possible rotations of the SE in the discrete grid. The openings are then combined by considering for each pixel the maximum value of these openings at each pixel position. This transformation is still an opening but only in the algebraic sense [13]. Let γ_i be the ith linear opening, the opening γ by union of linear openings can be written as $\gamma = \max_i \gamma_i$. By duality, one obtains the following relationship for closings ϕ by intersection of linear closings ϕ_i: $\phi(x) = \min_i \phi_i(x)$.

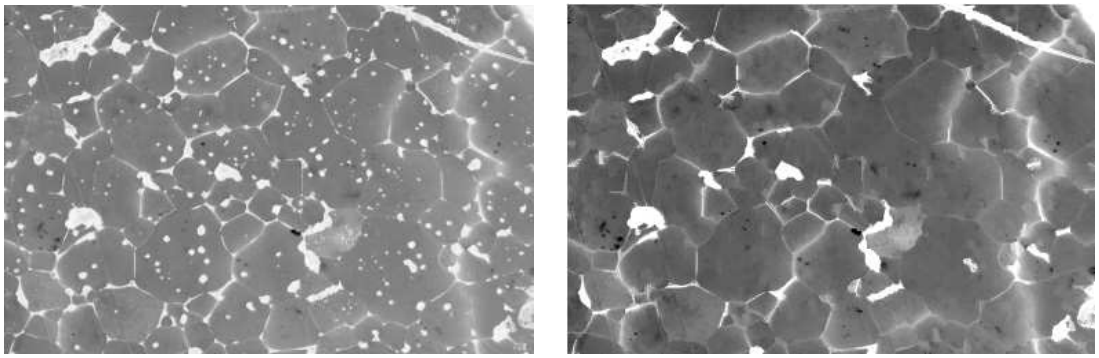
In Fig. 8 is an example of a union of linear openings using our algorithm for the corresponding linear erosions and dilations. The task here is to remove as many pores as possible from the magnesia image (Fig. 8a), while preserving the information concerning the grain boundaries (i.e., linear and bright image features). The resultant image, Fig. 8b, is obtained by considering for each pixel the maximum of 16 linear openings with a SE of extent 11 pixels.

B. Radial decomposition of discs

A method known as radial decomposition has been proposed by Adams [1] and enables dilations or erosions by discs to be approximated by a series of dilations or erosions defined on line segments. Let $L_{k_\alpha,\alpha}$ represent a linear SE of length k_α and orientation α, for $\alpha \in [0, \pi[$. Also let D_r be a disc SE of radius r. D_r is approximated by a $2n$ equal sided polygon using the dilation of n equally angularly spaced diameters of length k_α:

$$D_r \approx \bigoplus_{i=0}^{n-1} L_{k_{\alpha_i},\alpha_i}, \tag{2}$$

where $\alpha_i = i\pi/n$, $n \in \{2, 3, \ldots, \infty\}$ and k_{α_i} is obtained from Eq. 1 when k is replaced by the real number $r\pi/n$. Equation 2 produces a square when $n = 2$, a hexagon when $n = 3$, an octagon when $n = 4$, and so forth. Theoretically, the number n of linear SE's should be as large as possible to get



(a) (b)

Fig. 8 : An example of opening as the maximum of a series of linear openings. (a) An image of deadburned magnesia. The dark regions represent grains, while the light regions represent pores. (b) Opening as the maximum of 16 linear openings with an extent of 11 pixels.

the best approximation of the disc. However, the SE's $L_{k_{\alpha_i},\alpha_i}$ cannot all be exactly represented in a digital space. Therefore, Adams [1] proposed a table giving the optimal value of n for each disc of radius r.

Figure 9a shows a series of three Euclidean digital discs of radii 3, 11 and 21 pixels respectively. Figure 9b shows equivalent discs generated by radial decomposition using $n = 6$ periodic lines. Shown in Fig. 9c are the discs generated by radial decomposition using $n = 4$ Bresenham lines. As the angles used here generate lines that are either horizontal, vertical or at angles ± 45 degrees, the discs are convex. In contrast, the discs generated using $n = 6$ Bresenham lines are not convex, as is apparent in Fig. 9d: in these instances they contain both 4– and 8–connected pixels and hence there is neighbourhood variance along them. However, when the Bresenham line contains only 4–connected or only 8–connected pixels, as seen for the octagons (Fig. 9c), there is no neighbourhood variance along the Bresenham line and a regular $2n$–gon is produced. In the case of periodic lines, discs generated using odd n may result in a 'checker-board' structure and hence the disc will not be convex. However, using even n ensures that both vertical and horizontal line segments are used in the radial decomposition and the resulting discs will be convex.

As mentioned above, openings and closings constructed using cascades of erosions and dilations do not always form granulometries. If the opening γ by a digital disc of radius r is a granulometry, then $\gamma_{r_1}\gamma_{r_2} = \gamma_{r_2}$ when $r_1 < r_2$ and the residue $\gamma_{r_2} - \gamma_{r_1}\gamma_{r_2} = \emptyset$. Figure 10 shows the residues when the discs in Fig. 9 are opened by discs of smaller radii. Shown in Fig. 10a are the residues when the Euclidean disc of radius 21 is opened by itself, a Euclidean disc of radius 11, and a Euclidean disc of

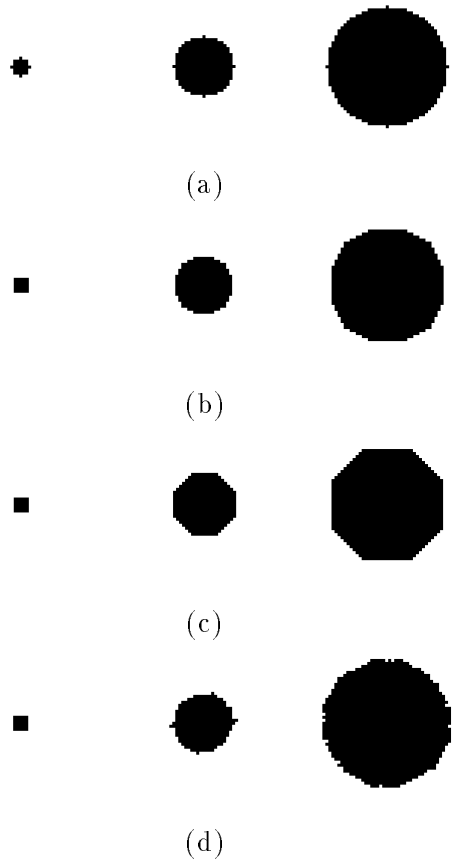


Fig. 9 : Radial decomposition of discs. (a) Euclidean digital discs of radii 3, 11, and 21 pixels respectively. (b) Discs generated by radial decomposition using periodic lines. (c) Discs generated by radial decomposition using Bresenham lines at 0, 45, 90 and 135 degrees. (d) Discs generated by radial decomposition using Bresenham lines.

radius 3 respectively. Notice that there is a residue in the second case, indicating that the opening using a Euclidean disc is not a granulometry. In Fig. 10b is shown the residues when the disc generated by radial decomposition using periodic lines are opened by consecutively smaller discs. In this case, there is no residue in the results, indicating that the discs generated using a cascade of periodic lines can be used for granulometric openings. The same result occurs for the octagonal discs generated using a cascade of four Bresenham lines. The residues for the discs generated using a cascade of six Bresenham lines is shown in Fig. 10d.

Shown in Fig. 11b is the image in Fig. 11a closed by a disc of radius 21 pixels, generated by a cascade of periodic line segments.

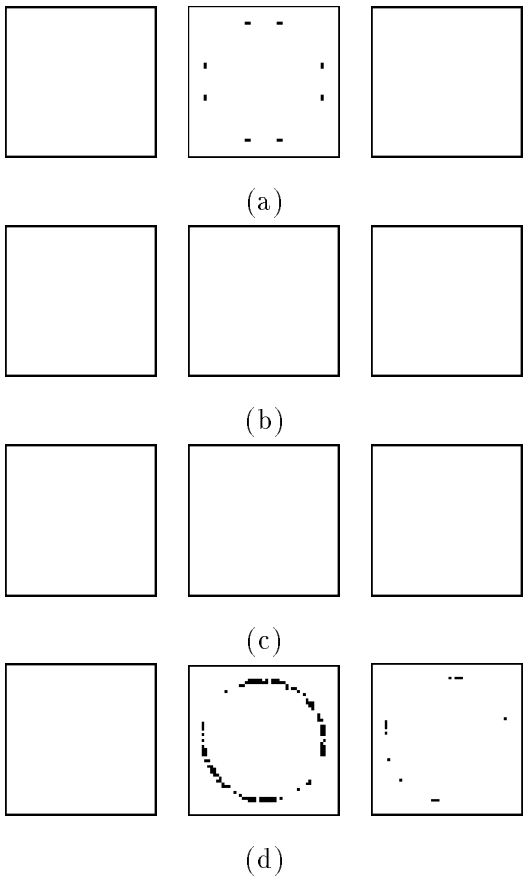


Fig. 10 : Residues generated by opening discs by consecutively smaller radii. (a) Residues for the Euclidean digital disc. (b) Residues for discs generated using radial decomposition by periodic lines. (c) Residues for octagons generated using radial decomposition by Bresenham lines. (d) Residues for discs generated using radial decomposition by Bresenham lines.

VI. Conclusion

In this paper, the algorithm presented by van Herk has been generalized to erosions and dilations along discrete lines at arbitrary angles for both Bresenham and periodic lines. We have also addressed the padding problem, that the image or buffer length does not have to be a discrete multiple of the length of the SE. In addition, applications to linear openings/closings and for radial decompositions of discs have been presented. These generalizations have extended considerably the range of SE's that are able to be implemented using the van Herk approach. The advantage is that it allows erosions/dilations by discs to be approximated at a computational cost of no more than $3n$ min/max operations per input pixel. In contrast, for a linear SE implementation, the cost would be no more than kn and for a disc SE element the computational cost would be proportional

(a) (b)

Fig. 11 : Example of closing by a disc. (a) Original image. (b) Closed image by disc of radius 31 pixels.

to r^2.

References

[1] R. Adams. Radial decomposition of discs and spheres. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, 55(5):325–332, September 1993.

[2] J. Bresenham. Algorithm for computer control of digital plotter. *IBM System Journal*, 4:25–30, 1965.

[3] J. Bresenham. A linear algorithm for incremental display of circular arcs. *Communications of the ACM*, 20(2):100–106, 1977.

[4] B. Chaudhuri. An efficient algorithm for running window pel gray level ranking in 2-D images. *Pattern Recognition Letters*, 11(2):77–80, 1990.

[5] S. Crabtree, L.-P. Yuan, and R. Ehrlich. A fast and accurate erosion-dilation method suitable for microcomputers. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, 53(3):283–290, May 1991.

[6] M. Van Droogenbroeck. On the implementation of morphological operations. In J. Serra and P. Soille, editors, *Mathematical morphology and its applications to image processing*, pages 241–248. Kluwer Academic Publishers, Dordrecht, 1994.

[7] R. Haralick, S. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):532–550, July 1987.

[8] B. Laÿ. Recursive algorithms in mathematical morphology. In *ACM Transactions on Graphics*, volume 6, pages 691–696. ICS Caen, 1987.

[9] J. Pecht. Speeding up successive Minkowski operations. *Pattern Recognition Letters*, 3(2):113–117, 1985.

[10] I. Ragnemalm. Fast erosion and dilation by contour processing and thresholding of distance maps. *Pattern Recognition Letters*, 13:161–166, 1992.

[11] J. Serra. *Image analysis and mathematical morphology*. Academic Press, London, 1982.

[12] J. Serra. Examples of structuring functions and their uses. In J. Serra, editor, *Image analysis and mathematical morphology. Volume 2: theoretical advances*, chapter 4, pages 71–99. Academic Press, 1988.

[13] J. Serra and L. Vincent. An overview of morphological filtering. *Circuits Systems Signal Process*, 11(1):47–108, 1992.

[14] R. van den Boomgaard and R. van Balen. Methods for fast morphological image transforms using bitmapped binary images. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, 54(3):252–258, May 1992.

[15] M. van Herk. A fast algorithm for local minimum and maximum filters on rectangular and octogonal kernels. *Pattern Recognition Letters*, 13:517–521, 1992.

[16] L. Vincent. Morphological transformations of binary images with arbitrary structuring elements. *Signal Processing*, 22(1):3–23, January 1991.

[17] I. Young, R. Pevereni, P. Verbeek, and P. Otterloo. A new implementation for the binary and Minkowski operators. *Computer Graphics and Image Processing*, 17:189–210, 1981.