Available online at www.sciencedirect.com
ScienceDirect
Image and Vision Computing 25 (2007) 416–425
www.elsevier.com/locate/imavis

Efficient complete and incomplete path openings and closings

Hugues Talbot [a,*], Ben Appleton [b]

[a] IGM-A2SI-ESIEE, 2 bd Blaise-Pascal, F-93162 Noisy-le-Grand Cedex, France
[b] Google Inc., Mountain View, CA, USA

Received 27 September 2005; received in revised form 20 June 2006; accepted 31 July 2006

Abstract

Path openings and closings are algebraic morphological operators using families of thin and oriented structuring elements that are not necessarily perfectly straight. These operators can typically be used in filtering applications in lieu of operators based on the more standard families of straight line structuring elements. They yield results which are less constrained than filters based on straight line segments, yet more constrained than connected area or other attribute-based operators. Furthermore, path operators can be parametrised to behave more like either extreme.

Natural implementations of this idea using actual suprema or infima of morphological operators with paths as structuring elements would imply exponential complexity. Fortunately, a linear complexity algorithm exists in the literature. This algorithm has similar running times as the best known implementation of morphological operators using straight lines as structuring elements.

However, even this implementation is sometimes not fast enough, leading practitioners to favour some attribute-based operators instead, which in some applications is not the best solution.

In this paper, we propose an implementation of path-based morphological operators that is shown experimentally to exhibit a logarithmic complexity and comparable computing times with those of attribute-based operators. This implementation has the added benefit of allowing the computation of the related opening transform at no extra computational cost.

In order to give additional flexibility and noise-robustness to these operators, we also investigate the case when some pixels are left ignored from the path (i.e. "jumps" are allowed) and form incomplete paths.

Keywords: Algebraic morphological operators; Attributes; Complexity

1. Introduction

Many problems in image analysis involve oriented, thin, line-like objects, for example measuring thin fibres [20,18], hair detection [17,14], blood vessel detection [7], grid-line extraction on stamped metal pieces [19] and others.

In an application where some bright, thin and elongated structure needs to be segmented, one typical approach is to remove the features in the image which are neither thin nor elongated. If the structures are also bright on a dark background, the standard approach would be to use an infimum of openings using lines as structuring elements oriented in many directions [9]. The result is an isotropic operator if the line structuring element lengths are adjusted to be independent of orientation [11].

The implementation of such an operator with actual lines as structuring elements is inefficient. However, using recursive implementations of openings at arbitrary angles yields a linear time algorithm [15] with respect to the length of the structuring elements. Note that this algorithm is not translation-invariant. A translation-invariant version, which must be used if features are very thin, was proposed in [16]. This version is more expensive but still of linear complexity.

Area and attributes openings [1,12,21] are also often used for the analysis of thin structures. An area opening

* Corresponding author.
E-mail addresses: h.talbot@esiee.fr (H. Talbot), ben.appleton@gmail.com (B. Appleton).

0262-8856/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2006.07.021

of parameter λ is equivalent to the supremum of all the openings by connected structuring elements of area λ. Clearly, this includes all the straight line structuring elements of this length.

Practitioners often note that using only straight line structuring elements removes too much of the desired features, while using connected area or other known attribute operators does not allow them to distinguish between long and narrow features on the one hand, and short compact ones on the other.

While it is sometimes possible to combine these operators to obtain the desired effect of retaining thin narrow structures while filtering out compact noise, this cannot always be done.

Recently efficient morphological operators using paths as structuring elements were proposed in [3,5]. Paths are families of narrow, elongated, yet not necessarily perfectly straight structuring elements. These path operators constitute a useful alternative to operators using only straight lines and those using area or other attributes.

Fig. 1 summarizes the usefulness of path operators. On this toy example we wish to eliminate the compact round object and retain the line-like features. An area opening does not work in this case because the compact round noise is too big and one feature is eliminated before the noise as the parameter increases. Similarly the supremum of openings by lines suppresses features that are not perfectly straight. On the other hand the path opening delivers the expected result. Note that path openings do not afford control over the thickness of detected paths, both the thin wavy line and the thicker straight line are detected. These can be separated using further morphological operators such as standard top-hats.

In the remainder, we propose ordered, significantly faster algorithms for implementing path operators, with logarithmic complexity with respect to the length of the structuring elements. The version with an unbroken path shall be called "complete path operators" and the version where a few pixels can be missing shall be called "incomplete path operators".

This algorithm also allows the user to compute the path opening or closing transform, the operator which for each pixel associates the length of the longest path going through it – in the foreground for the opening, and in the background for the closing, at no extra cost. Opening and closing transforms are useful for computing granulometries [10], for interactively selecting opening and closing parameters and for filtering [22]. In the paper we extend this notion to grey-scale images.

In the final section, an algorithm is also proposed to compute *incomplete* path operators, where a few pixels can be missing at arbitrary locations along the path. This complexifies the algorithms significantly but adds a measure of noise robustness in the manner of rank-max openings [13].

2. Path-based morphological opening

The theory of path openings is explained in detail in [5] and in a shorter fashion in [4]. We only summarize the main points here.

2.1. Definitions

Let E be a discrete 2-D image domain, a subset of $\mathbb{Z}^2$. Then $\mathscr{B} = \mathscr{P}(E) = 2^E$ is the set of binary images and $\mathscr{G} = \mathscr{T}^E$ the space of grey-scale functions, where $\mathscr{T}$ is the set of grey-values. We assume E is endowed with an adjacency relation $x \mapsto y$ meaning that there is a directed edge going from x to y. Using the adjacency relation we can define the structuring function $\delta(x) = \{y \in E,\ x \mapsto y\}$. The L-tuple $\boldsymbol{a} = (a_1, a_2, \ldots, a_L)$ is called a *δ-path of length L* if $a_{k+1} \in \delta(a_k)$ for $k = 1, 2, \ldots, L-1$. Given a path $\boldsymbol{a}$ in E, we denote by $\sigma(\boldsymbol{a})$ the set of its elements, i.e: $\sigma(a_1, a_2, \ldots, a_L) = \{a_1, a_2, \ldots, a_L\}$. We denote the set of all δ-paths of length L by Π_L and the set of δ-paths of length L contained in a subset X of E is denoted by $\Pi_L(X)$.

2.2. Adjacency graphs

In the following, we shall define morphological operators using families of paths as structuring elements. These paths shall be defined on adjacency graphs. The layout of this graph is of course of critical importance. In theory it can be arbitrary. However, in practice it shall be most commonly repetitive and bound to the discrete grid. The main reason for this is that practitioners most commonly wish morphological operators to be translation-invariant. If the layout of the adjacency graph is translation-invariant for whole pixel translations, then any associated path operator also shall be.

Simple examples of adjacency graph are the familiar 8-connected – shown on Fig. 2(a), and 4-connected graphs associated with the square grid. However, paths in such a

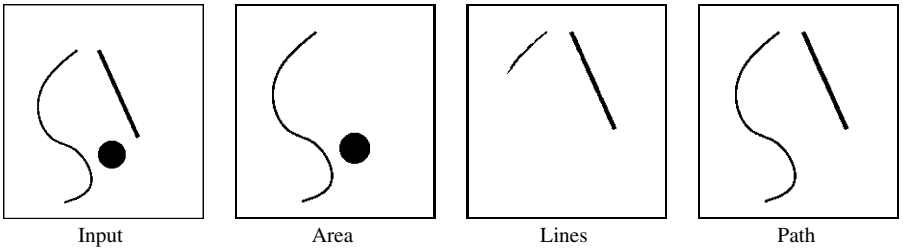

Fig. 1. Toy example: on the input we wish to retain the line-like features while eliminating compact noise. Only the path opening works in this case.

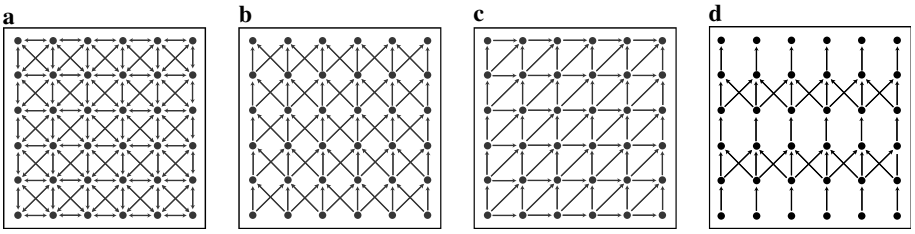

Fig. 2. Some examples of adjacency graphs: (a) 8-connected graph, (b) S–N 90° graph, (c) SW–NE 90° graph, (d) S–N 45° graph.

graph follow no particular pattern, and would not possess any particular orientation. Morphological operators using such paths would simply reduce to connected area operators. *A contrario*, using an adjacency graph consisting only of either vertical or horizontal edges would result in the familiar operators with respectively vertical or horizontal line segments.

In contrast, using as underlying graph that of Fig. 2(b), valid paths on such graphs are constrained at each of the path vertex to entirely fit in a 90° angle cone oriented south–north (SN), as shown in Fig. 3. These are clearly oriented but not necessarily perfectly straight. Similarly the graph of Fig. 2(c) constrains paths to fit in a 90° angle cone oriented SW–NE.

As path operators fit in their usefulness in between connected area operators and operators with segments as SE, practitioners might elect to vary the adjacency graph to be closer to one extreme or the other. For example the graph of Fig. 2(d) constraints paths to fit in a narrower S–N cone, of aperture close to 45°. This will result in path operators closer in their effects to operators with line segments as SE. Note that operators using this graph alone will not be translation-invariant, because the graph itself is not. Nevertheless the supremum (resp. infimum) of path openings (resp. closing) using the same graph shifted one pixel row down shall be translation-invariant.

It should be noted that in the following adjacency graphs shall be oriented out of necessity, as we will not allow paths to back down on themselves, but that the choice of global orientation is arbitrary. In other words, pointing all the arrows down instead of up in the graph of Fig. 2(b) would result in the same operators.

2.3. Path openings

We define the operator $\alpha_L(X)$ as the union of all δ-paths of length L contained in X:

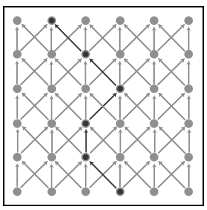

Fig. 3. Valid path with the the adjacency graph of Fig. 2(b).

$$\alpha_L(X) = \bigcup \{\sigma(\boldsymbol{a}), \boldsymbol{a} \in \Pi_L(X)\} \tag{1}$$

It is easy to establish that α_L has all the properties of an opening. Fig. 4 offers an illustration using the adjacency graph of Fig. 2(b). For this graph, if $X = E = \mathbb{Z}^2$, there are 3^{L-1} distinct paths of length L starting from any point. For bounded image this number may vary due to edge effects, but remains true far enough from the edges. The path opening α_L is in fact the union of the morphological opening using these paths as structuring elements, which would suggest an inefficient way to compute the operator. Fortunately, [5] proposes a useful recursive decomposition which allows the operator α_L to be computed in linear time with respect to L. Clearly path openings will depend on the choice of adjacency, as discussed in the previous subsection.

In the remainder, for simplicity, we refer almost exclusively to the S–N 90° adjacency of Fig. 2(b) and its 90-degree rotations.

2.4. Threshold decomposition

Let X be a binary image, i.e. an element $\mathscr{B}$. We use the Boolean indicator function b defined on $\mathbb{Z}^2$ as follows:

$$\begin{aligned} b(x) &= \textbf{true}, \text{ if } x \in X \\ &= \textbf{false}, \text{ if not.} \end{aligned}$$

Given a grey-scale image $g \in \mathrm{G}$, a threshold operator T_t: $\mathscr{G} \to \mathscr{B}$ with threshold t, and a binary opening $\gamma_\mathrm{B} : \mathscr{B} \to \mathscr{B}$,there exists a grey-scale opening γ_G: $\mathscr{G} \to \mathscr{G}$ such that for all thresholds t we have $T_t \circ \gamma_\mathrm{G} = \gamma_\mathrm{B} \circ T_t$, where $\circ$ is the composition operator.

This grey-scale opening $\gamma_\mathrm{G}(\mathrm{g})$ may be constructed explicitly by 'stacking' the results of the binary opening applied

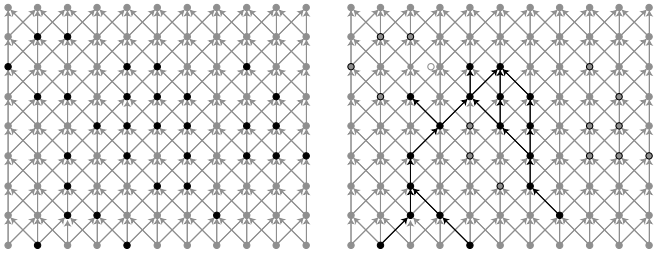

Fig. 4. A set $X \subseteq E$ (black points on the left) and its opening $\alpha_6(X)$ (black points on the right). Unfilled points on the right have been discarded. The underlying adjacency graph is in light grey.

to each threshold of the original image. This stacking assigns to each pixel p the highest threshold t for which the binary opening $\gamma_B \circ T_t(g)$ is **true.**

2.5. Grey-level operator and practical considerations

The binary operator defined in earlier sections extends to the grey-level domain by threshold decomposition. The recursive decomposition in [5] also extends to the grey-level domain in the same way.

In order for path openings and closings to be useful in a context where features of interest are arbitrarily oriented, one must compose paths openings and closing in the usual way through supremums and infimums (resp.), much in the same way as when using straight segments as structuring elements [11]. However, in order to achieve acceptable levels of isotropy with path operators, far fewer compositions are necessary.

By definition of a path opening with the adjacency graph of Fig. 2(b), only features containing valid paths according to that graph – i.e. oriented South–North within a 90° cone, shall be preserved by the opening. By taking the supremum with the same adjacency graph rotated 90°, clockwise for example, one would also preserve paths oriented East–West in the opening. Recall from Section 2.2 that the direction of the arrows in the graph does not matter as long as it is consistent. To further improve isotropy, one can also use the adjacency graphs of Fig. 2(c) and its 90° rotation and their corresponding openings.

As described earlier, one can also use adjacency graphs such as that of Fig. 2(d), which is more constraining, as paths must fit in a narrower cone. In this case, to achieve acceptable levels of isotropy one must compose operators with rotated versions of that graph.

3. Ordered algorithm

The grey-scale path opening algorithm presented in this paper is based on a few simple observations. First, the principle of threshold decomposition allows the construction of grey-scale morphological operators from binary morphological operators. Second, in the case of grey-scale path openings it is possible to efficiently compute the set of binary path openings for all thresholds in sequence.

3.1. Updating binary path lengths

The second observation is that the binary images produced in this construction tend to change little between sequential thresholds. In the case of path openings we will show how to efficiently update the result of the binary opening $\gamma_B \circ T_t(g)$ from the result of the binary opening at the previous threshold $\gamma_B \circ T_{t-1}(g)$.

For brevity we here describe only the case of the 90-degree S–N adjacency of Fig. 2(b). In this case, for a binary image b we store at each pixel p two values: the length $\lambda^-[p]$ of the longest path travelling upward from pixel p (not including p itself), and the length $\lambda^+[p]$ of the longest path travelling downward from pixel p. Then the length of the longest path passing through pixel p (where $b[p] = \textbf{true}$) is $\lambda[p] = \lambda^-[p] + \lambda^+[p] + 1$. If $b[p] = \textbf{false}$ then we set $\lambda[p] = 0$. The recursive computation of λ^- and λ^+ is described in details in [5]. Here, in short, we may state that in the stated case, for $b[p] = \textbf{true}$,

$$\lambda^-[p] = 1 + \max \lambda^-[(p^1 - 1, p^2 + 1)], \lambda^-[(p^1, p^2 + 1)], \lambda^-[(p^1 + 1, p^2 + 1)] \quad (2)$$

and

$$\lambda^+[p] = 1 + \max \lambda^+[(p^1 - 1, p^2 - 1)], \lambda^+[(p^1, p^2 - 1)], \lambda^+[(p^1 + 1, p^2 - 1)] \quad (3)$$

For $b[p] = \textbf{false}$, $\lambda^-[p]$ and $\lambda^+[p]$ are both 0. In both these equations p^1 denotes the first coordinate of p, and p^2 the second, i.e. the coordinates of p are (p^l, p^2).Note that we are actually computing the *opening transform* for a given threshold, i.e. the operator that associates to each pixel the length of the longest path going through that pixel [10]. The path opening is obtained by thresholding this opening transform image at a given parameter length L.

Now, in order to update the binary opening $\gamma_B \circ T_t(g)$ given the result from the previous threshold $\gamma_B \circ T_{t-1}(g)$, we must compute the new binary opening transform λ and hence λ^- and λ^+. Rather than recomputing these from the image $b = T_t(g)$,we may compute the changes to λ^- and λ^+ due solely to the pixels which made the transition from **true** to **false** between $T_{t-1}(g)$ and $T_t(g)$. This is performed in the following steps:

Algorithm 3.1 (*Updating* λ^- and λ^+).

- Initialisation:
  - Set all pixels with $g[p] = t$ to active and enqueue
- For each row from top to bottom:
  - For all active pixels p in this row:
    - Recompute $\lambda^-[p]$ according to Eq. (2)
    - If $\lambda^-[p]$ changed, set as active and enqueue the dependent pixels $(p^1 - 1, p^2 + 1)$, $(p^1, p^2 + 1)$, $(p^1 + 1, p^2 + 1)$
- For each row from bottom to top:
  - For all active pixels p in this row:
    - Recompute $\lambda^+[p]$ according to Eq. (3)
    - If $\lambda^+[p]$ changed, set as active and enqueue the dependent pixels $(p^1 - 1, p^2 - 1)$, $(p^1, p^2 - 1)$, $(p^1 + l, p^2 - 1)$

The queueing system for *active* pixels consists of a first-in-first-out (FIFO) queue for each row as well as a queue of rows which contain active pixels. This queueing system is necessary to comply with the dependencies in Eqs. (2) and (3) and also avoids inefficiently scanning the entire image.

3.2. Recursive ordered path opening

Here, we use the preceding updating method to compute the grey-scale path opening. L denotes the desired path length. Note that, as we are interested in the specific path length L, path lengths λ^-, λ^+ greater than $L-1$ may be treated as equal to $L-1$ in Algorithm 3.1 – that is, $\lambda^-[p]$ and $\lambda^+[p]$ are not considered 'changed' if their value increase above $L-1$. This limits the propagation of changes to the binary opening transform and hence improves the efficiency of the grey-scale path opening.

Algorithm 3.2 (*Recursive ordered path opening*).

- Initialisation:
  - Sort the pixels by their intensities
  - Set $b[p] = \textbf{true}$ for all pixels p
  - Compute λ^+, λ^- from b
- For each threshold t in $\mathscr{T}$:
  - Using Algorithm 3.1, update λ^-, λ^+ for the new threshold
  - For all active pixels p whose path length $\lambda[p]$ became shorter than L as the result of the update, set $\gamma_G(g)[p] = t$

Sorting the pixels by their intensities is a necessary preprocessing step in order to efficiently locate the pixels whose threshold changes in the step $t-1 \to t$. For integer data a linear-time sorting algorithm such as the Radix sort is recommended [8]. Alternatively a suitable priority queue data structure [2] can be used.

A simple heuristic has been found to further improve the efficiency of this algorithm in practice. When the maximal path length of a pixel p drops below L, it cannot contribute to a path of length L or greater at any further threshold. Therefore we may remove this pixel from further consideration by setting $b[p] = \textbf{false}$. We refer to this as the *length heuristic* in the remainder of this paper. We believe that the average running time of this algorithm is $O(N \log L)$ on images containing N pixels. However, the formal derivation of this average running time would require the selection of an appropriate stochastic image model and is not pursued in this paper.

Note that Algorithm 3.2 works for arbitrary integer data, not necessarily unsigned (positive) data types. In particular the zero threshold holds no special significance. In practice images use discrete data types, so we have a finite number of thresholds to process. We start at the threshold immediately below the global minimum of the image. Starting at the global minimum threshold is also valid but saves little time in practice.

4. Grey-scale opening transform

The algorithm presented in Section 3.2 may be extended in a simple manner to compute the grey-scale path opening transform. Recall from Section 3.1 that the binary opening transform, in this instance, is the operator which at each pixel associates the length of the longest path going through that pixel. Here the grey-scale path opening transform associates a whole vector to each pixel. That vector contains the lengths of the longest path going through it for each successive threshold.

The binary opening transform is typically used to efficiently compute granulometries. Indeed the histogram of the binary opening transform image is exactly the granulometry curve of the original image. Unfortunately, there is no such simple relationship in the grey-scale case. However, useful operators based on the grey-scale opening transform using simpler structuring elements have been proposed in the literature, for example in [23,22].

In the course of Algorithm 3.2, the path opening transforms for all binary thresholds were computed in sequence. Instead of discarding these intermediate results we may store them in compressed form allowing them to be queried at a later time. At each threshold, those *active* pixels whose maximal path length $\lambda[p]$ has decreased store a point $(t, \lambda[p])$ in a linked list. This linked list is monotonically increasing in t and monotonically decreasing in $\lambda[p]$ Once computed, we may query this structure with any desired path length to extract the associated grey-scale path opening.

Algorithm 4.1 (*Recursive ordered path transform*).

- Initialisation: As per Algorithm 2.2
- For each threshold t in $\mathscr{T}$:
  - Using Algorithm 3.1, update λ^-, λ^+ for the new threshold.
  - For all active pixels p whose path length $\lambda[p]$ decreased, append the node $(t, \lambda[p])$ to the linked list at pixel p.

This algorithm requires the same order of computation as Algorithm 3.2, that is $O(N \log L)$. The number of linked list nodes generated in Algorithm 4.1 must be less than the number of operations in Algorithm 3.2, and therefore the average memory required by Algorithm 4.1 is $O(N \log L)$. This may also suggest that path openings are inherently informative, as we may store all path openings in $O(\log L)$ bytes per pixel rather than $O(L)$ as may be initially expected.

5. Incomplete paths

We also consider the case of incomplete paths, both binary and grey-scale, as defined in [5]. Paths openings with increasing L are also increasingly sensitive to noise, as long paths are more likely to contain noisy pixels. In order to decrease the sensitivity to noise, it is useful to allow a few pixels to be ignored along the structuring paths.

This is precisely what the rank-max opening achieves [13]. It has proved to be useful [6] and workable especially when using line structuring elements [16]. However, the rank-max opening implementation relies on being able to compute an arbitrary rank filter using the chosen structur-

ing element as a window. It is a challenging task to implement this efficiently when using paths for windows.

Heijmans et al. [5] discuss and provide an incomplete path operator algorithm whose complexity is in $O(NLK)$, where N is the number of pixels in the image, L the length of the paths and K the tolerance, i.e. the number of pixels that are allowed to be ignored along the paths.

In effect, the result is an opening (or a closing) using a family of paths of length L which can have as many as K missing pixels at arbitrary locations along the way, including the extremities. It is clear that this family of SE is even larger that the family used for regular (or complete) path opening, and that the Heijmans et al. implementation is a tremendous improvement over a trivial implementation which would use the supremum of morphological openings over this family.

However, we can implement the incomplete opening in the same way as in the previous section, in an ordered fashion, at the cost of significantly increased bookkeeping. Since for each missing pixel in the path, each path might be split into two subpaths, we need to track changes in each of these as we go up the grey levels. The bookkeeping, and the memory requirements costs are therefore proportional to K.

For example, for a tolerance of a single missing pixel, we assume the path is made of two sub-paths, one of which can be of 0-length, and a gap pixel. In Algorithm 3.2, instead of a single path to update, we have an array of two. As we to up in the threshold we update both.

5.1. Incomplete path opening algorithm

Here we need to give new versions of the updating equations of Section 3.1. Eq. (2) becomes

$$\begin{aligned}\text{if } b[p] \text{ is } \textbf{false}: \quad \lambda^-[p,k] &= 1 + \max\begin{pmatrix}\lambda^-[(p^1-1,p^2+1),k-1],\\ \lambda^-[(p^1,p^2+1),k-1],\\ \lambda^-[(p^1+1,p^2+1),k-1]\end{pmatrix} \quad (4)\\ \text{else}: \quad \lambda^-[p,k] &= 1 + \max\begin{pmatrix}\lambda^-[(p^1-1,p^2+1),k],\\ \lambda^-[(p^1,p^2+1),k],\\ \lambda^-[(p^1+1,p^2+1),k]\end{pmatrix}\end{aligned}$$

i.e. if $b[p]$ is **false**, then we penalise the path accordingly and inherit the path length from the path with $k-1$ gaps. Apart from this change the updating algorithm is identical to Algorithm 3.1, and we inherit the path length from the path with k gaps. Note that when $k=0$, we interpret $\lambda[p(\cdots),-1]$ as zero length. In this way we can see that it collapses back to the old algorithm for zero gaps.

Eq. (2) is modified likewise for $\lambda^-[p,k]$.

Algorithm 5.1 (*Incomplete path opening*)**.**

- Initialisation:
  - Sort the pixels by their intensities
  - Set $b[p] = \textbf{true}$ for all pixels p
  - Set $\lambda^+[p,k]$, $\lambda^-[p,k]$ to maximum for every pixel.
- For each threshold t in $\mathscr{T}$:
  - For each k in increasing order, apply Algorithm 3.1 to update $\lambda^-[p,k]$, $\lambda^+[p,k]$ for the new threshold.
  - For all active pixels p whose path length $\lambda[p] = \max_k\{\lambda^-[p,k] + \lambda^+[p,K-k-\bar{b}[p]]\}$ became shorter than L as the result of the update, set $\gamma_G(g)[p] = t$

In the computation of $\lambda[p]$, K is the maximum number of gaps, k the current number of gaps in the upward path, $\bar{b}[p]$ is the negation of $b[p]$, and **true** $= 1$, **false** $= 0$ where necessary in the interpretation of $b[p]$. This is considering all possible arrangements of the K gaps over the up/down paths, partitioned as follows:

- The upward path (including the pixel p) has k gaps in it
- The downward path (including the pixel p) has $K-k$ gaps in it
- $b[p]$ itself may indicate an extra gap, hence the complexity of the λ^+ expression

6. Results

Here we present results regarding the regular (or complete) path opening and the incomplete version.

6.1. Complete paths

An example of use is shown in Fig. 5. We wish to detect the small thin fibres in this electron micrograph present at the bottom of this image. The large fibres are detected by a different method [18] which is of no interest here, and removed from the image, so as not to perturb the detection of the small fibres. The thin fibres are present on a noisy background which requires some filtering. A supremum of openings by lines is too crude here (result not shown). An area opening does not eliminate enough of the noise, but a complete path opening works as expected.

6.2. Opening transform

We give an example of the use of the opening transform. In Fig. 6, we have computed the binary opening transform of the noisy top-hat image of Fig. 5.

In this figure, (a) is the result of the transform. Each path is associated with its length, thus longer paths appear brighter, (b) is the granulometry curve, obtained from the cumulated histogram of this image, which can be used for image classification or finding an appropriate length threshold automatically. For example, thresholding at the first plateau allows the user to eliminate all the short paths noise and yields the same result as the final image in Fig. 5.

6.3. Incomplete paths

Here we give two examples for the use of incomplete paths.

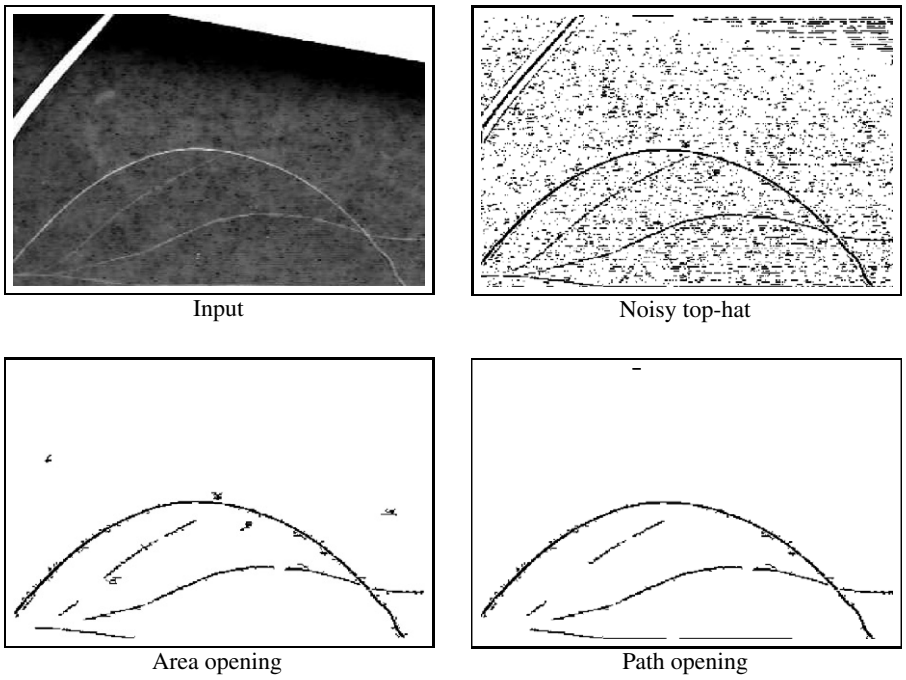

Fig. 5. Electron micrograph of glass fibres: to detect the small thin fibres in the bottom of the image, a white top-hat is useful but noisy. When this top-hat image is filtered by an area opening some compact noise remains while a path opening yields a better result.

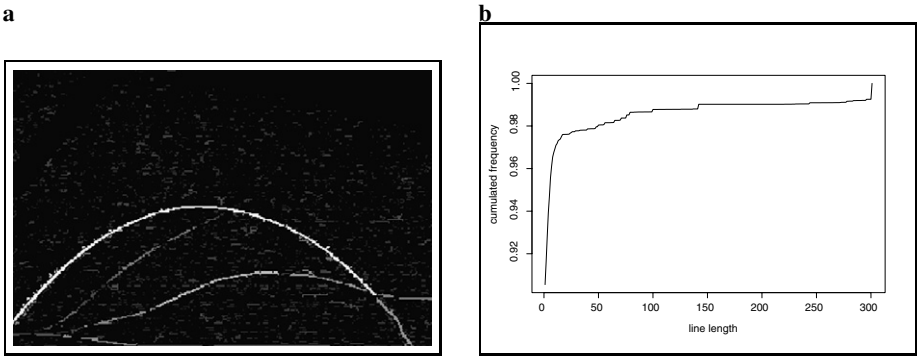

Fig. 6. (a) The opening transform of the noisy top-hat image of Fig. 5. (b) The granulometry curve, obtained from the histogram of (a).

6.3.1. Illustrative example

Again we use an artificial but perhaps more interesting example as an illustration. In Fig. 7(a) we have an example image of lines and curves, corrupted by heavy noise in (b). A supremum of openings using straight lines as structuring elements of total length 21 pixels, covering all discrete orientations recovers eliminates all information (not shown). Similarly, as shown in Fig. 7(d), a complete path opening with the same length eliminates most of the information. If instead of a regular opening, a rank-max opening with tolerance 2 pixels is used, most of the noise is eliminated, and the two straight lines remain, as shown in Fig. 7(c). However, some random noise pixels happen to be well aligned, and so some artifacts are generated at the bottom of the image. These would be very difficult to eliminate by further filtering.

Using an incomplete path opening of the same length as before, but with tolerance 2 pixels, we recover the data shown in Fig. 7(e). Most of the noise is eliminated and both lines and the curve are still in the data. Further filtering (e.g. dilation followed by thinning) can be used to recover the lines and curve even better if needs be.

6.3.2. Application to music OCR

As a realistic illustration, we use an experimental music OCR system currently in development at ESIEE. The idea behind music OCR is to be able to convert sheet music to exploitable digital data, in the same way standard OCR is used for text.

Fig. 8(a) shows a small portion of a musical sheet. All processings are performed in grey-levels. The objective is to segment all the components of the music as image elements for later recognition. For example we can readily extract beams in Fig. 8(b), and note and alteration stems in Fig. 8(c). Other elements which do not intersect others are also relatively easy to segment, such as the lyrics or

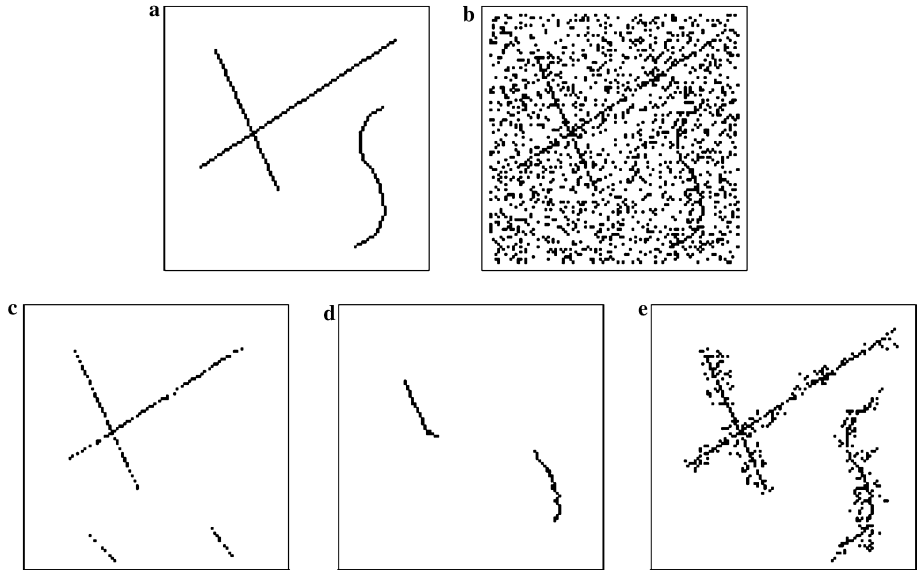

Fig. 7. Artificial example of (a) straight lines and a thin curve, which are (b) corrupted by heavy salt and pepper noise (15% of each). Foreground is black. A supremum of rank-max openings (c) with line structuring elements recovers the straight line at the expense of some artifacts. A complete path opening (d) recovers very little, while an incomplete path opening with a tolerance of 2 pixels (e) recovers both lines and the curve.

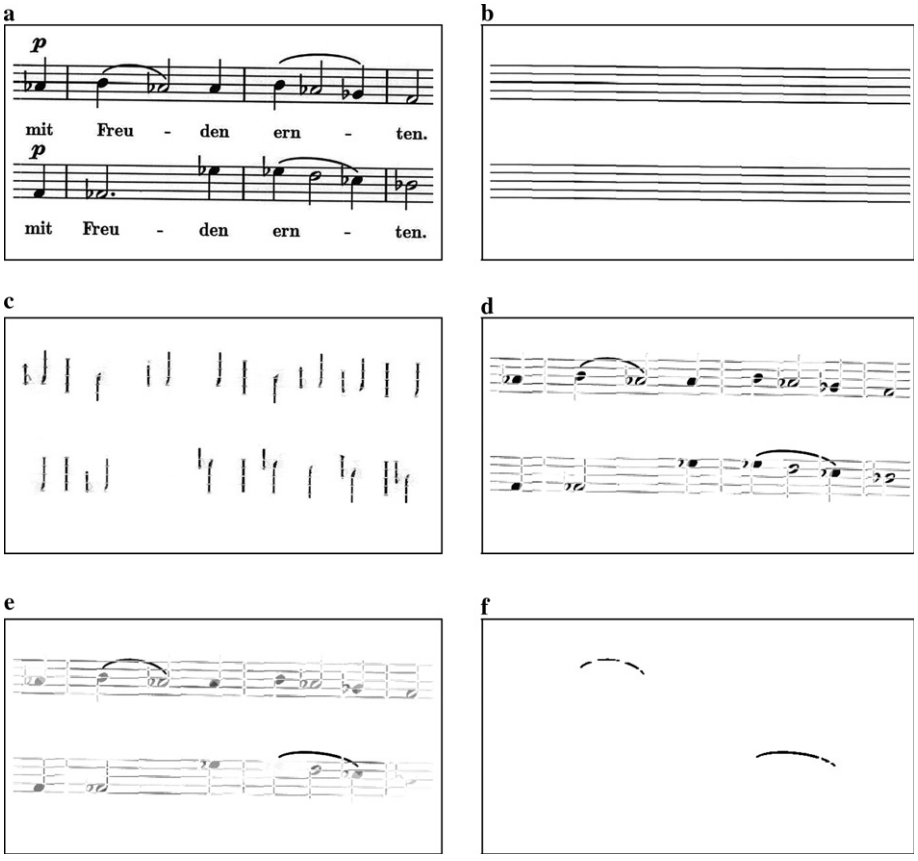

Fig. 8. A few steps in a music OCR system–slur segmentation, (a) original image; (b) beams; (c) stems; (d) residues; (e) incomplete path closing on residues, length = 80, tolerance = 5; (f) final slurs by thresholding. Note that all processing is done on the grey-level information.

various elements above and below the main beams (not shown).

However, ties and slurs on the main beams have proved difficult to segment, as they are not straight elements and intersect with others. One possible method to segment them is to remove the stems from the notes, as in Fig. 8(d). In this image remain only compact elements and the ties/slurs. However, because of the use of the difference operator, the

slurs do not form connected elements anymore, and of course they are still elements of a curve rather than a straight line.

Using a incomplete path closing we can eliminate all compact "noise" (the notes in that case) and keep the slurs mostly intact, as shown in Fig. 8(e). Thresholding and the use of a closing by a 2×2 square SE yields the slur segmentation, as shown in Fig. 8(f). Note that we do find the full extension of the slurs in spite of the gaps.

6.4. Timings

6.4.1. Complete openings/closings

Table 1 shows the running times of the complete proposed algorithm compared with various alternatives. We observe that the proposed ordered path opening implementation has a running time approximately logarithmic (plus a constant) with respect to L, while both the recursive path opening and the supremum of openings by lines have approximately linear running times. Note that the individual openings by lines in the latter algorithm are all running in constant time irrespective of L, but for larger L more orientations need to be explored. Note also that the presented algorithm for the supremum of opening by lines is a non-translation-invariant implementation. A translation-invariant version would be significantly slower still. The area opening algorithm seems to converge to a constant-time algorithm with low constant. The area parameter was simply L, although a constant times L could have been chosen, e.g. $3 \times L$ without significantly affecting the result.

Memory demands for these algorithms are all low except the recursive path opening implementation which requires an amount of memory proportional to LN, with N the number of pixels in the image.

We observe that the area opening is the fastest algorithm by far, but that the presented path opening algorithm comes second, a factor of between 4 and 5 times slower than the area opening, however still significantly faster than the other two algorithms for most useful values of L.

6.4.2. Incomplete openings/closings

Table 2 shows the running times of the proposed incomplete, ordered path opening compared with various relevant algorithms. First the complete path opening presented earlier, then a supremum of rank-max openings with lines as structuring elements, and finally the same incomplete path openings but implemented with the algorithm of Heijmans et al.

Table 1
Comparison for complete path openings

L	Ordered PO	Recursive PO	Supremum lines	Area
1	0.56	0.08	0.14	0.13
5	0.69	0.54	0.65	0.17
10	0.73	1.16	1.38	0.17
50	0.90	14.24	3.29	0.21
100	0.93	30.74	11.43	0.22

From left to right the columns are the ordered complete path opening presented in this paper, the recursive path opening of Heijmans et al., the supremum of openings by lines and the area opening. Timings are in seconds, image was $560 \times 510 \times 8$-bit. Processor was a single Pentium IV 1.7 GHz.

Table 2
Comparison for complete/incomplete path openings

Algorithm Tolerance	COPO $K = 0$	SRMO $K = 3$	IOPO $K = 1$	IOPO $K = 3$	IRPO $K = 1$	IRPO $K = 3$
$L = 1$	0.56	0.20	5.9	11.3	21.3	39
$L = 5$	0.69	0.92	11.1	24.2	59	120
$L = 10$	0.73	1.80	12.0	27.6	123	283
$L = 50$	0.90	4.8	14.6	34.5	576	1560
$L = 100$	0.93	19.1	15.3	37.1	1200	3100

From left to right the columns are COPO, the complete ordered path opening; SRMO, the supremum of rank-max opening with lines, tolerance = 3; IOPO, the incomplete path opening with tolerance 1 and 3 pixels from this article and IRPO the incomplete recursive path opening with tolerance 1 and 3 pixels from Heijmans et al. Timings are in seconds, image was $560 \times 510 \times 8$-bit. Processor was a single Pentium IV 1.7 GHz.

We observe that the incomplete ordered path opening (IOPO) is much slower than the complete version, The running time of the incomplete opening is approximately $10{*}k$ that of the complete version, and so we verify experimentally the expected complexity of the algorithm.

We also observe that the IOPO is generally slower than using a supremum of rank-max openings (SRMO) with lines, except for small K and large L, which is to be expected since the complexity of the SRMO with lines is $O(LN)$. SRMOs running time are also independent of K (for $K > 0$). We should note that the IOPO algorithm does something significantly more complicated and flexible than the SRMO.

As well, the proposed ordered algorithm is much faster than the previously available recursive algorithm of Heijmans et al., by a factor of 4–80, depending on L.

6.5. Discussion

Results obtained by path openings depend greatly on the adjacency graph. It is for example possible to define narrower or wider cones for paths as discussed in [5]. Using narrower and more numerous cones one can define paths that are increasingly similar to lines, and hence obtain results similar to those obtained by regular openings by unions of line structuring elements. Using fewer and wider cones one can obtain results more similar to area openings.

In the results above we used 90° angle cones which are easy to implement and seem to strike a good balance between area openings and line-based openings.

7. Conclusion and future work

We have presented new, ordered implementations of the complete and incomplete path opening and closing operators. These operators are identical to the supremum (resp. infimum) of openings by a family of structuring elements described as oriented paths, possibly with a limited number of missing pixels along the way. The family of paths is of

exponential size with respect to both their length L and K, the number of allowed missing pixels. However, a recursive implementation with only linear complexity with respect to L and K is available in the literature.

We have proposed a new implementation with an experimental logarithmic complexity with respect to L and linear in K, which is observed to be much faster than the recursive implementation except for very small L. It is also faster than the usual operator by unions of lines structuring elements used for the study of thin structures, at least for small K.

For the complete operator, area operators are still faster than the proposed implementation, by a nearly constant factor of 4–5. However, the proposed algorithm is fast enough for many applications and can be used in cases where using an area or attribute operator is not appropriate, e.g. in the presence of sufficiently large compact noise.

For the incomplete operator, comparison with the area operator is not appropriate. The only comparable, but less flexible operator is the rank-max opening using lines as SE. A union of such operators along a large number of orientations yields conceptually similar, but in practice significantly different results. For small K and large L, our proposed implementation yields comparable running times, but in other cases the rank-max based operator is faster. However, the proposed implementation is always much faster than the recursive algorithm available in the literature, and the gap increases with L.

The path operators are intuitive methods useful for the analysis of thin, elongated but not necessarily perfectly straight structures. The present implementation is fast enough for application use, at least in the complete path version. The incomplete version provides a powerful operator useful in the presence of noise or for intricate problems involving thin curves, but the present version is probably still too slow for day-to-day use.

Future work will include studying ways to use the rank operator in a manner similar to the rank-max operator to render incomplete path operators independent on K.

Acknowledgements

The authors thank the anonymous reviewers for greatly improving this paper. **Source code.** We make the source code of our implementation available at the following URL: http://www.esiee.fr/~talboth/Research/Morpho/IOPO/.

References

[1] E. Breen, R. Jones, Attribute openings, thinnings and granulometries, Computer Vision, Graphics, and Image Processing: Image Understanding 64 (3) (1996) 377–389.

[2] E.J. Breen, D. Monro, An evaluation of priority queues for mathematical morphology, in: J. Serra, P. Soille (Eds.), Mathematical Morphology and its Applications to Image Processing, Kluwer, 1994, pp. 249–256.

[3] M. Buckley, H. Talbot, Flexible linear openings and closings, in: L. Vincent, D. Bloomberg (Eds.), Mathematical Morphology and its Application to Image Analysis, June 2000, Palo Alto, ISMM, Kluwer, pp. 109–118.

[4] H. Heijmans, M. Buckley, H. Talbot, Path-based morphological openings, in: Proceedings of ICIP'04, October 2004, Singapore, IEEE.

[5] H. Heijmans, M. Buckley, H. Talbot, Path openings and closings, Journal of Mathematical Imaging and Vision 22 (2005) 107–119.

[6] H.J.A.M. Heijmans, Morphological filters for dummies, in: Petros Maragos, W. Ronald, Schafer, Muhammad A. Butt (Eds.), Mathematical Morphology and its Applications to Image and Signal Processing, May 1996, Atlanta, GA, Proceedings for ISMM'96, Kluwer Academy, pp. 127–137.

[7] F. Jaumier, Application de la morphologie mathématique à la détection de formes sur les clichés angiofluorographiques de la rétine. Master's thesis, ENPC, July 1988.

[8] D. Knuth, The Art of Computer Programming, vol. 3: Sorting and Searching, Addison-Wesley, 1973.

[9] M.B. Kurdy and D. Jeulin, Directional mathematical morphology operations, in: Proceedings of the Fifth European Congress for Stereology, vol. 8/2, Freiburg im Breisgau, Germany, 1989. Acta Stereologica, pp. 473–480.

[10] B. Laÿ, Recursive algorithms in mathematical morphology, in: Seventh International Congress For Stereology. Acta Stereologica 6 (III) 691–696, Sept 1987.

[11] C.L. Luengo Hendriks, L.J. van Vliet, A rotation-invariant morphology for shape analysis of anisotropic objects and structures, in: C. Arcelli, L.P. Cordella, G. Sanniti di Baja (Eds.), Visual Form 2001: Fourth International Workshop on Visual Form, 2001, Capri, Italy, Springer, LNCS, p. 378.

[12] A. Meijster, H.F. Wilkinson, A comparison of algorithms for connected set openings and closings, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (4) (2002) 484–494.

[13] C. Ronse, Extraction of narrow peaks and ridges in images by combination of local low rank and max filters: implementation and applications to clinical angiography, Working Document WD47, Philips Research Laboratories, Brussels, Belgium, 1988.

[14] V.N. Skladnev, A. Gutenev, S. Menzies, L.M. Bischof, H.G.F Talbot, E.J. Breen, M.J. Buckley, Diagnostic feature extraction in dermatological examination, Technical Report, International Publication Number WO 02/094098 A1, Polartechnics Limited, 2001.

[15] P. Soille, E. Breen, R. Jones, Recursive implementation of erosions and dilations along discrete lines at arbitrary angles, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (5) (1996) 562–567.

[16] P. Soille, H. Talbot, Directional morphological filtering, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (11) (2001) 1313–1329.

[17] T. Lee, V. Ng, R. Gallagher, A. Coldman, D. McLean, Dullrazor: a software approach to hair removal from images, Computers in Biology and Medicine 27 (1997) 533–543.

[18] H. Talbot, T. Lee, D. Jeulin, D. Hanton, L.W. Hobbs, Image analysis of insulation mineral fibres, Journal of Microscopy 200 (3) (2000) 251–268.

[19] A. Tuzikov, P. Soille, D. Jeulin, H. Bruneel, M. Vermeulen, Extraction of grid lines on stamped metal pieces using mathematical morphology, in: Proceedings of Eleventh IAPR International Conference on Pattern Recognition, Conference A: Computer Vision and Applications, September 1992, vol. 1, The Hague, pp. 425–428.

[20] G. van Antwerpen, P.W. Verbeek, F.C.A Groen, Automatic counting of asbestos fibres, in: I.T. Young et al. (Eds.), Signal Processing III: Theories and Applications, 1986, pp. 891–896.

[21] L. Vincent, Grayscale area openings and closings, their efficient implementation and applications, in: Proceedings of the Conference on Mathematical Morphology and its Applications to Signal Processing, May 1993, Barcelona, Spain, pp. 22–27.

[22] L. Vincent, Local grayscale granulometries based on opening trees, in: P. Maragos, R.W. Schafer, M.A. Butt (Eds.), Mathematical Morphology and its Applications to Signal and Image Processing, May 1996, Atlanta, GA, ISMM, Kluwer, pp. 273–280.

[23] R.C. Vogt, A spatially variant, locally adaptive, background normalization operator, in: J. Serra, P. Soille (Eds.), Mathematical Morphology and its Applications to Image Processing, Sept. 1994, Fontainebleau, ISMM, Kluwer, pp. 45–52.