MINIMAL PATH ALGORITHMS FOR THE ROBUST DETECTION OF LINEAR FEATURES IN GRAY IMAGES

LUC VINCENT

Xerox Scansoft, Inc.

3400 Hillview Avenue, Palo Alto CA 94306, USA

Abstract.
Minimal paths are proposed as a powerful way to extract faint linear structures in noisy gray-level images. A *global* algorithm based on gray-weighted distance transforms is first proposed and shown to sometimes be a compelling alternative to watersheds. A more general *local* minimal path method is also introduced, together with an algorithm for extracting minimal path cost at each pixel location. The proposed algorithms are very efficient, and examples of applications are used to demonstrate their robustness.

Key words: Algorithms, Generalized Distance Transforms, Linear Features, Minimal Paths, Mathematical Morphology, Segmentation, Watersheds.

1. Introduction

A range of image analysis applications are concerned with the extraction of linear features, such as edges, fibers, fault lines, scratches, etc. A common way to approach the problem is to use a morphological *top hat* [7], followed by skeletonization. Traditional edge detection operators can also be used, and in some cases watershed based algorithms provide a powerful alternative [9, 1]. However, in a number of difficult cases, image complexity, faintness of linear structures to be extracted, or high noise level combine to make such approaches fail.

In the present paper, we first examine the difficult problem of extracting correlogram tracks from sonar data [4]. Even watersheds are unable to extract these tracks under low signal-to-noise (SNR) conditions. Analyzing the reasons for this failure leads us to propose a robust "generalized dynamic programming" method based on *minimal paths.* The core of this method is a gray-weighted distance function, a very efficient recursive implementation of which is described in Section 2.

The global nature of this minimal path algorithm is the very reason why it is so powerful, but it is also a limitation: to extract a linear feature L from a grayscale image I, some knowledge about the location of the end-points of L is required, which is not always practical. To address this limitation, a novel *local minimal path* transform is introduced in Section 3, together with an efficient algorithm to compute it. It provides a new and very robust way to detect linear features in images. We demonstrate this by using this method to extract faint and thin glass fibers from noisy grayscale images.

2. Watersheds and Global Minimal Path Algorithms

Let us first examine a difficult image analysis problem: the detection of correlogram tracks in sonar data. Examples of such data are shown in Figs. 1a–b. In each case, the image contains a continuous track, running from the top to the bottom of the image. In image 1a, signal-to-noise ratio (SNR) is relatively high, whereas in image 1b, it is very low and the track is barely visible.

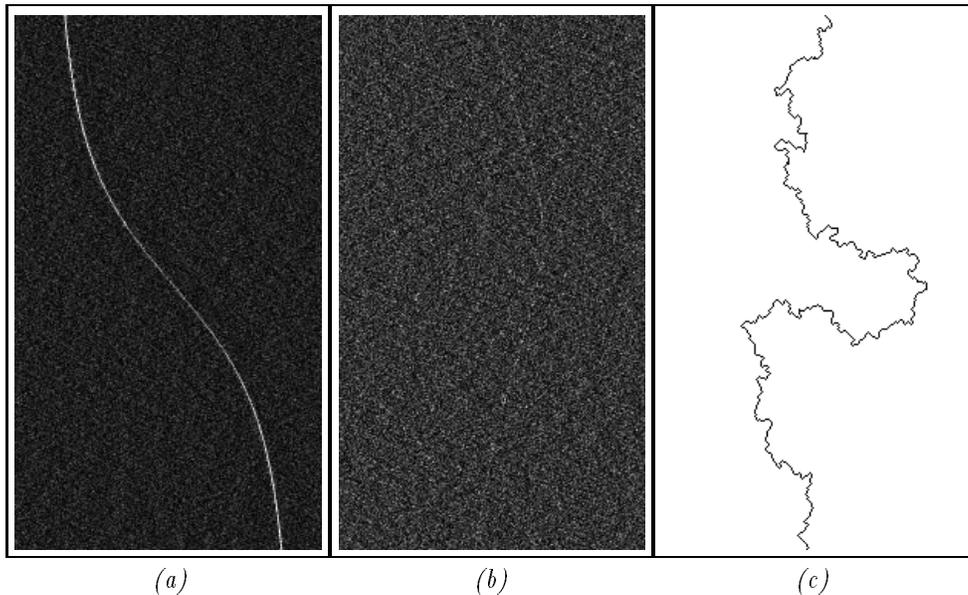(a)   (b)   (c)

Fig. 1.   (a)–(b) Correlogram tracks with high SNR and low SNR respectively. (c) Attempt to use watersheds for extracting track in (b).

In either case, one can think of the track to be extracted as a "crest-line" running from the top of the image to the bottom of the image. The powerful watershed transformation is therefore an obvious candidate for extracting such tracks: following the segmentation methodology exposed in [9, 1], one can constrain the watershed by using as markers the right side and the left side of the image. This is equivalent to saying that we wish to extract a crest-line separating the right side from the left side, that is, running from the top of the image to the bottom of the image.

This approach works extremely well for Fig. 1a, where the track is so well marked that a number of alternative methods would work equally well. However, the watershed approach fails on Fig. 1b: the crest line it extracts (See Fig. 1c) looks fairly "random" and bares little similarity with the desired correlogram track

This apparent contradiction can be explained by the very nature of the watershed algorithm. Examining the algorithm described in [9], or any other derived algorithm, one can get convinced of the following two facts:

1. The watershed line between two markers A and B in an image I highly depends on the position of the *saddle points* between these two markers. In other words, considering all the paths that join A to B with the minimal elevation (i.e., gray-

level) gain in I, the highest pixels along those paths are the saddle points, and their location is one of the main factors determining the location of the watershed line between A and B.

2. The criteria used to build the watershed are all solely based on gray level, and length of watershed lines is irrelevant.

In particular, fact #1 above means that a "disconnected" crest-line between A and B will not necessarily be extracted by the watershed operator: it all depends on the gray values of pixels in between the disconnected pieces. In addition, even if the watershed line does end up going over some of these disconnected pieces, there is absolutely no guarantee that it will do it in a way that minimizes its overall length. These two points are illustrated by Fig. 2.
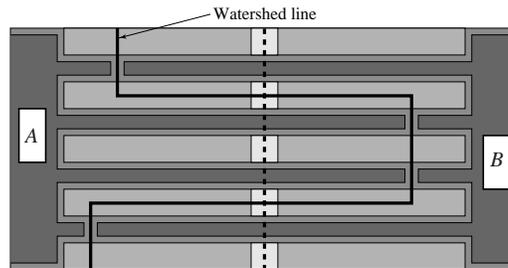

Fig. 2.   The watershed line between A and B is very tortuous and does not connect all the pieces of the vertical bright feature in the center of the image. The vertical dashed line represents what one might want to extract as optimal crest line between A and B.

Length constraints can be introduced by using *global minimal paths* algorithms, as already proposed in [4]. Suppose that we wish to *connect* two regions A and B of a grayscale image I using the *shortest path(s) going preferentially over dark pixels*. This can be achieved by considering the neighborhood graph (typically, 4- or 8-connected grid) G of pixels in image I. In this graph, each edge E between two neighboring pixels p and q is assigned value $V_I(E) = V_I(p,q) = I(p) + I(q)$, or any other increasing function of $I(p)$ and $I(q)$, such as $\max(I(p), I(q))$ or $\min(I(p), I(q))$ (See Fig. 3). Each path P in this graph is now associated a *cost* $C_I(P)$, equal to the sum of all its edge values[1]. This provides a new metric in image I, for which the distance d_I between two pixels p and q is given by: $d_I(p, q) = \min\{C_I(P), P \text{ path between } p \text{ and } q\}$.
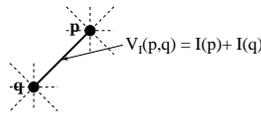

Fig. 3.   Valuation of edge E between pixels p and q.

The problem comes down to extracting the path(s) of minimal cost between regions A and B, according to distance d_I. This is almost a classic *dynamic pro-*

---

[1] $C_I(P)$ should not be mistaken for the *length* $l(P)$ of path P: the latter is equal to the number of edges in P, that is, its number of pixels minus 1.

gramming problem, except that the paths considered here do not necessarily have to be monotonous. One can easily prove that a pixel p belongs to such minimal paths if and only if $d_I(p, A) + d_I(p, B) = d_I(A, B)$. To extract such paths, we can therefore proceed as follows:

- Compute *gray-weighted* distance transform to A, that is, for each pixel p, compute $d_I(p, A)$;

- Compute gray-weighted distance transform to B;

- Sum these two distance functions and threshold the result in order to only keep the pixels p such that $d_I(p, A) + d_I(p, B) = \min_q\{d_I(q, A) + d(q, B)\}$.

Note that the general method of using "forward" and "backward" distance functions is not new, and was already used, e.g., to extract the longest path in a binary set [6], or to detect potential fracture lines in porous media [8].

From an algorithmic point of view, the problem is reduced to computing gray-weighted distance transforms. This can easily be done by modifying the classic two-pass sequential distance function algorithm [5] so that: (1) edge cost is taken into account; (2) raster and anti-raster scans of the image are iterated until stability. More specifically, denoting by $N^+(p)$ (resp., $N^-(p)$) the neighbors of pixel p scanned *before p* (resp. *after p*) in a raster scan of image I, the algorithm proceeds as follows:

Algorithm: Generalized distance function to set A in image I

- `Initialize result image` J: $J(p) = 0$ `if` $p \in A$ `and` $+\infty$ `otherwise`;
- `Iterate until stability:`
  - `Scan image in raster order; For each pixel` p`, do:`
    $$J(p) \leftarrow \min\{J(p), \min\{J(q) + V_I(p, q), q \in N^+(p)\}\};$$
  - `Scan image in anti-raster order; For each pixel` p`, do:`
    $$J(p) \leftarrow \min\{J(p), \min\{J(q) + V_I(p, q), q \in N^-(p)\}\};$$

This algorithm typically converges in two or three iterations and is therefore extremely efficient. Applying this method to our correlograms, we use as set A the first line of the image, and as set B, the last line. In addition, since the extracted minimal paths are preferentially located on dark pixels, we invert the original image before computing the two generalized distance functions needed. This method, which proved to be very robust, is illustrated in Fig. 4. For more details, see [4].

3. Local Minimal Paths

Unfortunately, in a number of applications, the linear feature extraction problem cannot be formulated as the extraction of a minimal path running between known regions of an image, therefore the global approach described in the previous section cannot be used. A number of authors have proposed techniques for extracting linear structures in difficult images. For example, in [2], directional morphological operations are used to enhance images without destroying linear features and directional dilations are used to combine disconnected lines. In [3], a metric is proposed for line segments and used to provide robust connection criteria.

Such methods still fail to provide satisfactory solutions in some complex situations. In addition, in cases where the problem can indeed be formulated in terms of global minimal path, then the algorithm described in Section 2 proves to be far
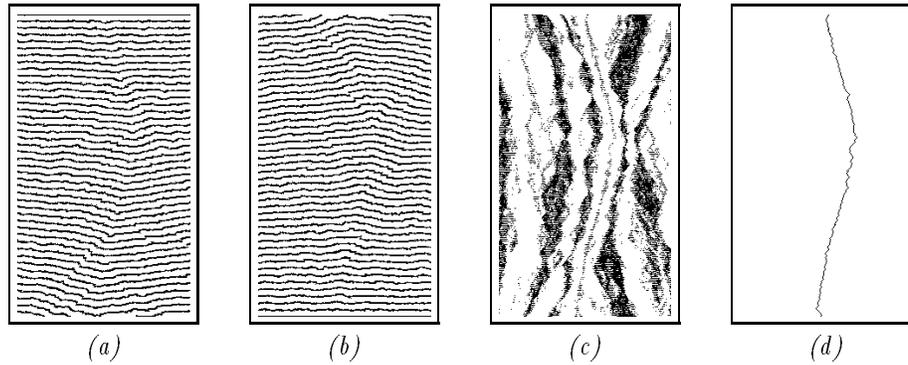
(a) (b) (c) (d)

Fig. 4. (a) Gray-weighted distance function to top of image, (b) Gray-weighted distance function to bottom of image, (c) Binarized sum of distance functions, (d) Minimal path extracted between the top and the bottom of the image.

superior. This leads us to propose a *local,* or more exactly, *regional* version of this minimal path algorithm. The principle is as follows:

> *Assign to each pixel p of an image I the minimal cost $C_I(P)$ for all the paths P of a given length l originating in p and whose possible orientation and straightness are within a given range.*

Specifically: (1) the cost $C_I(P)$ is taken to be the sum of the pixel values along path P; (2) let $\theta_1, \theta_2, \ldots, \theta_n$ be a set of possible orientations for the desired paths, and let $\delta\theta$ be the maximal acceptable deviation: the paths P considered must be such that there exists an i such that the dominant orientation of P is within $[\theta_i - \delta\theta, \theta_i + \delta\theta]$.

This orientation constraint is critical: it prevents a candidate minimal path P from looping back on itself or going around pixel p in a circle! It also provides a means to specify the local degree of straightness of desired linear features. Finally, this constraint is key to making the local minimal path algorithm introduced in the present section computationally tractable. In the discrete world, this constraint is implemented as follows: draw a square of size $(2l + 1) \times (2l + 1)$ pixels centered at pixel p; then draw two discrete lines starting from p, at orientation $\theta_i - \delta\theta$ and $\theta_i + \delta\theta$ respectively. These two lines delineate a section of the square, as shown in Fig. 5. The straightness constraint above is then taken to mean that path P is entirely included in this region, which we refer to as "search cone".
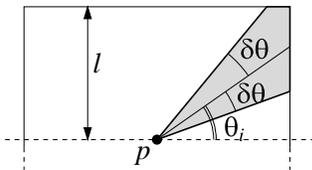

Fig. 5. Search cone: region defining the "orientation tolerance" for paths in direction θ_i.

Implementing this *local minimal path* transformation by considering each pixel p and exhaustively analyzing all the acceptable paths starting at p would be extraor-

dinarily slow. We now introduce efficient recursive algorithms for computing this transformation. Let us first consider the following simple case:

- $\delta\theta = \pi/4$
- $i = 4$, with $\theta_1 = 0, \theta_2 = \pi/2, \theta_3 = \pi, \theta_4 = 3\pi/2$.

The search cones for minimal paths are now four simple triangles, as illustrated by Fig. 6. These structures being the l-fold Minkowski addition of the unit-size 4-pixel triangles with themselves, a simple recursive implementation can be proposed for the local minimal path algorithm. This implementation merely consists of a sequence of *global image shifts, global pixelwise minima, global image additions* as follows:

Algorithm: Local minimal path transform, simple case

- For $\theta = 0$, $\pi/2$, π and $3\pi/2$, do:
    - $I_\theta \leftarrow I$ (image copy); Then repeat l times:
        - For $\alpha = \theta - \pi/4, \theta, \theta + \pi/4$:
            - $J_\alpha \leftarrow I_\theta$ shifted by 1 pixel in direction $-\alpha$;
    - $I_\theta \leftarrow I_\theta + \min_\alpha\{J_\alpha\}$;
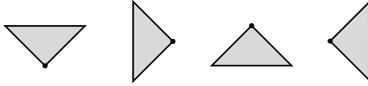- Result: $J \leftarrow \min_\theta\{I_\theta\}$

Fig. 6. Triangular "search cones" used when $\delta\theta = \pi/4$, $\theta_1 = 0$, $\theta_2 = \pi/2$, $\theta_2 = \pi$, $\theta_4 = 3\pi/2$.

In this particular case, for a given length l, the algorithm requires $12l$ shift operations, $8l$ pointwise minima operations, and $4l$ image additions. All these steps are very simple and run extremely quickly on any computer. The resulting algorithm is therefore very efficient. It can also be adapted to work with overlapping search cones, or thinner search cones, which often improves the quality of the results.

However, the narrowest search cones that can be used with this method are 45 degree angle triangles. In many cases, this is still not narrow enough. With search cones narrower than 45 degrees, a fully recursive algorithm cannot be used any more. This means that for a search cone of k pixels, the number of image shifts, pixelwise minima, and pixelwise additions needed will be $O(k)$: these simple global image operations being extremely fast, this is still acceptable. However, to minimize the total number of operations needed, we now introduce a *partly recursive* algorithm.

Let us describe this method using the search cone of Fig. 7. In order to extract the minimal path between the pointy side of the cone (left) and the right side, a breadth-first (First-In-First-Out) scan is used. During this scan, each pixel p is assigned: (1) the cost of the minimal path between it and the right side of the triangle; (2) a configuration label config(p). This configuration uniquely characterizes the number of "ancestors" of p in the search cone, as well as the configuration of these ancestors. For example, looking at Fig. 7, all the pixels with configuration "4" have exactly two ancestors in the search cone, each of them with label "2". Each time a new configuration is uncovered, the value of the corresponding pixel is computed from the value of its "ancestors", that is, its (one or two) neigbors that were previously scanned. Since this step is the same for all the pixels in the image, it is

also parallelized by means of global image shift, pixelwise minimum, and pixelwise addition operations. However, when the configuration of the current pixel p has been seen previously, it means that the shift, minimum, and addition operations needed to compute the value of this pixel have already been performed! We therefore skip to the next pixel. In the example of Fig. 7, this technique enables us to uncover 10 pixels whose configuration is equivalent to that of a previously scanned pixel. We therefore cut down the total number of global image shifts, minima and additions by about 30% in this particular case.
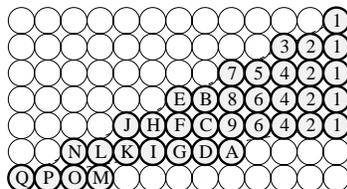

Fig. 7.   Successive configurations $1, 2, \ldots, 9, A, B, \ldots, P, Q$ (26 of them) encountered when recursively scanning the pie slice shaped region starting from its right side.

Experimentation was conducted using 512×512 images on a 233 MHz Pentium PC. When using length parameter $l = 15$ pixels, $\delta\theta = 10$ degrees, and $\theta = 0$, 15, 30,..., 345 degrees, several hundred image shifts, pixelwise minima and additions were needed, but the minimal path image was computed in the order of only a minute. This time could still easily be cut by a factor of 5 to 10 by optimizing the speed of the global image operations the algorithm is currently based on.

To illustrate this method, we use the image in Fig. 8a, which represents a thin glass fiber in a rather noisy environment. To extract this fiber, one could carefully design a filter to attenuate background noise, then use a "top hat" followed by skeletonization and pruning... The local minimal path method provides a much more streamlined approach: by applying the algorithm directly on Fig. 8, we combine all the pixels in the fiber and greatly attenuate background noise, as shown in Fig. 8b.

As can be seen on this image, the side effect of this algorithm is that the fiber now appears thicker: we can extract it as one object by simple thresholding, as shown in Fig. 8a, but, extra work is needed to thin it down to unit-pixel thickness. To do this robustly, we compute the *longest path* inside the previously extracted object: this can be done using gray-weighted distance transforms geodesically inside the previously extracted object. The excellent result thus obtained is show in Fig. 8d.

4. Conclusion, Future Work

The use of paths of minimal cost for the extraction of linear structures in images was explored. A number of efficient algorithms were proposed to extract such minimal paths as well as to label each pixel p with the cost of the minimal path(s) originating in p. These algorithms are either *global* or *regional* in nature: to compute the output value of any given pixel, a large neighborhood—sometimes the entire image—is taken into account. As a result, these proposed methods are particularly robust, making it possible to extract even the faintest linear structures.
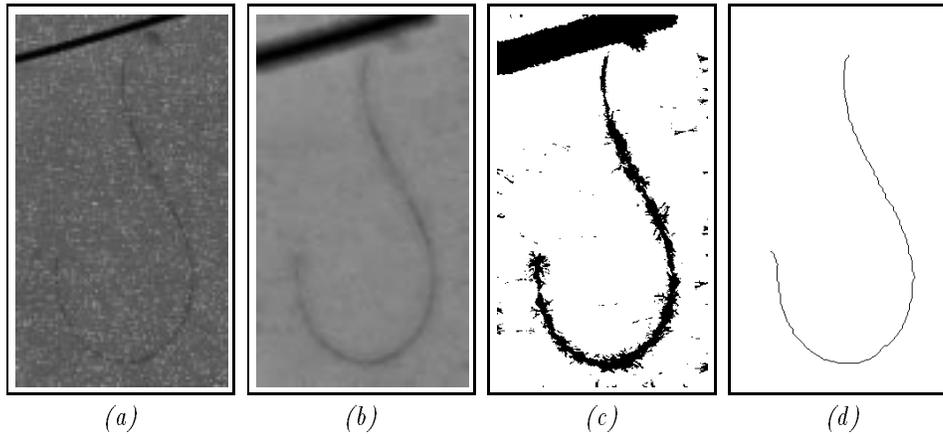
Fig. 8.   (a) Original image of thin glass fibers; (b) Result of local minimal path algorithm, using
paths of length 15 and overlapping search cones of 15 degrees; (c) Thresholded minimal path image;
(d) Extracted fiber.

One of the most novel methods introduced in the paper is the local minimal algo-
rithm described in Section 3. While the proposed technique is a vast improvement
over a "brute force" approach, the resulting algorithm remains CPU intensive. Work
is currently underway to further improve this algorithm, e.g., through the careful se-
lection of the "search cones" it uses. In addition, relationships between watersheds
and minimal paths are being further analyzed. For example, relaxation methods
are being explored as a way to constrain minimal path extraction using markers,
as is commonly done with watershed segmentation. This work will be reported in
subsequent publications.

References

1.   S. Beucher and F. Meyer. The morphological approach to segmentation: the watershed trans-
     formation. In E. R. Dougherty, editor, *Mathematical Morphology in Image Processing*, pages
     433–481. Marcel-Dekker, Sept. 1992.
2.   B. Kurdy and D. Jeulin. Directional mathematical morphology operations. In *5th European
     Congress For Stereology*, pages 473–480, Freiburg im Breisgau FRG, Sept. 1989. Acta Stereo-
     logica. Vol. 8/2.
3.   P. F. Nacken. A metric for line segments. *IEEE Trans. Pattern Anal. Machine Intell.*, 15:1312–
     1318, 1993.
4.   B. Rosen and L. Vincent. Morphological image processing techniques applied to detection of
     correlogram tracks. *U.S. Navy Journal of Underwater Acoustics*, 44(2):571–586, Apr. 1994.
5.   A. Rosenfeld and J. Pfaltz. Distance functions on digital pictures. *Pattern Recognition*, 1:33–61,
     1968.
6.   M. Schmitt. *Des Algorithmes Morphologiques à l'Intelligence Artificielle*. PhD thesis, Ecole
     des Mines, Paris, Feb. 1989.
7.   J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.
8.   L. Vincent and D. Jeulin. Minimal paths and crack propagation simulations. In *5th Euro-
     pean Congress For Stereology*, pages 487–494, Freiburg im Breisgau FRG, Sept. 1989. Acta
     Stereologica. Vol. 8/2.
9.   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.