Constrained and Dimensionality-Independent Path Openings

Cris L. Luengo Hendriks, *Member, IEEE*

***Abstract*—Path openings and closings are morphological operations with flexible line segments as structuring elements. These line segments have the ability to adapt to local image structures, and can be used to detect lines that are not perfectly straight. They also are a convenient and efficient alternative to straight line segments as structuring elements when the exact orientation of lines in the image is not known. These path operations are defined by an adjacency relation, which typically allows for lines that are approximately horizontal, vertical or diagonal. However, because this definition allows zig-zag lines, diagonal paths can be much shorter than the corresponding horizontal or vertical paths. This undoubtedly causes problems when attempting to use path operations for length measurements.**

This paper (*1*) introduces a dimensionality-independent implementation of the path opening and closing algorithm by Appleton and Talbot, (*2*) proposes a constraint on the path operations to improve their ability to perform length measurements, and (*3*) shows how to use path openings and closings in a granulometry to obtain the length distribution of elongated structures directly from a gray-value image, without a need for binarizing the image and identifying individual objects.

I. Introduction

Path openings and closings are morphological operations whose structuring elements are flexible line segments. These line segments have a general orientation, but due to their flexibility they can rotate and bend to adapt to local image structures. Path openings and closings were first proposed by Buckley and Talbot [1], and received a more thorough theoretical foundation by Heijmans, Buckley and Talbot [2], [3]. A path opening of length L is equivalent to the supremum of all openings with structuring elements composed of L connected pixels arranged according to a specific adjacency relation. In two dimensions (2D) there are four simple adjacency relations: one that produces approximately horizontal lines, one that produces approximately vertical lines, and two that produce approximately diagonal lines. The horizontal path is formed by adding pixels either horizontally or diagonally, but always to the right (North-East (NE), East (E) and South-East (SE) neighbors). The vertical path is formed using the North-West (NW), North (N) and NE neighbors, and the diagonal paths using either the N, NE and E or the E, SE and South (S) neighbors. Figure 1 shows the horizontal path connectivity diagram and some example horizontal paths.

Obviously, computing the opening with each of the possible connected path structuring elements of length L is prohibitive even for small values of L, since there are $O(3^L)$ such paths. Buckley and Talbot [1] proposed a recursive algorithm to compute path openings that is $O(NL)$ (with N the number of pixels in the image), and needs $2L-2$ temporary images. Later, Appleton and Talbot [4], [5] proposed a more efficient algorithm that seems to be $O(N\log(L))$, and only requires three temporary images. Both algorithms are explicitly defined for 2D images. Section II proposes a simplification to the Appleton and Talbot algorithm, which allows for a definition that is independent of image dimensionality.

A granulometry is a standard tool in mathematical morphology that builds a size distribution of objects in an image by applying an opening (or closing) of increasing size, and summing all pixel values after each step [6], [7]. In previous work [8], I have shown how a supremum of openings (or an infimum of closings) with line structuring elements at all orientations can be used in a granulometry to measure the length of objects in the image without segmenting the image first. Because the number of orientations needed increases linearly with the line length in 2D [9], this can be a time-consuming operation in a 2D image. But it becomes prohibitive in 3D, where the number of orientations needed depends quadratically on the length. It is therefore attractive to use path openings instead. In 2D there are only 4 orientations over which to compute the path opening, independent of path length. In 3D there are 13 possible orientations. Using Appleton and Talbot's algorithm, the operation's cost grows logarithmically with length, for any number of dimensions. Two-dimensional path openings have recently been used in a similar manner to detect roads in satellite images [10].

There is, however, one caveat when using path openings: as will be shown, diagonal paths can zig-zag (e.g. N, E, N, E, N, etc. instead of NE, NE, NE, NE, NE, etc.), resulting in a path that is physically much shorter than expected given the pixel count. Section III shows how to avoid this. The constraint introduced in that section also narrows the possible orientations for one path, making it more selective. This constraint is similar to that proposed by Buckley and Yang [11] for shortest path extraction, though implemented in a very different manner.

Subsection IV-D shows how the methods proposed in this paper can be applied to estimate the length of wood fibers in a 3D microtomographic image, without the need to identify individual fibers through complex segmentation routines. Other parts of Section IV highlight other possible uses of the path opening operation.

The source code for the path openings as proposed in this paper, together with the scripts used to generate all the results presented, are available on line at http://www.cb.uu.se/~cris/pathopenings.html. All tests were done in MATLAB (The

C. Luengo is with the Centre for Image Analysis, Swedish University of Agricultural Sciences, Uppsala, Sweden. e-mail: cris@cb.uu.se

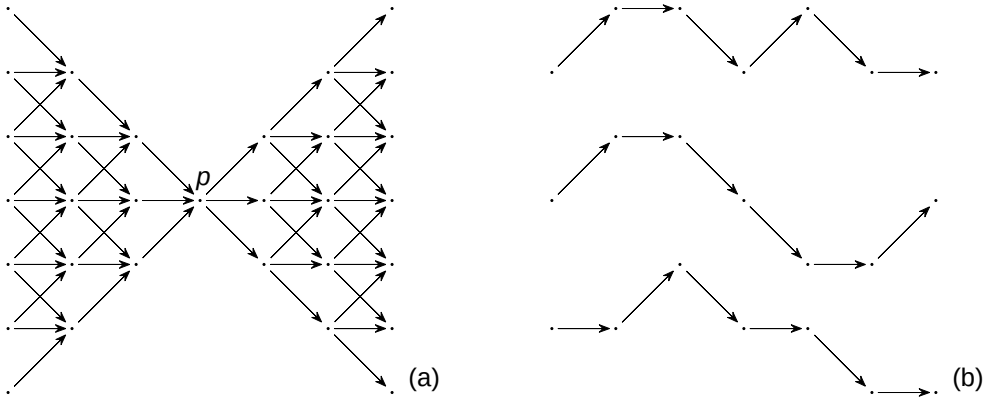

Fig. 1. *a*: Graph containing all possible 4-pixel horizontal paths that go through pixel p. To compute the 4-pixel path opening, one would compute the minimum value over each path (104 combinations) and use the maximum of these values. *b*: Three possible horizontal paths with 7 pixels.

MathWorks, Inc., Natick, MA) using the DIPimage toolbox (http://www.diplib.org/). Path opening algorithms were implemented in C.

II. d-Dimensional Path Openings

This section describes the modifications to the published algorithm to make it dimensionality independent. First I will give a short description of the algorithm as presented by Appleton and Talbot [4], [5]. This section ends with some implementation details.

A. Appleton and Talbot's Path Opening Algorithm

This is an ordered algorithm, which means it sorts all pixel values and, starting at the lowest gray value, processes each pixel exactly once. The algorithm requires three temporary images. One temporary, binary image b marks each pixel as *active* or *inactive*. Initially, all pixels are active, and become inactive as they are assigned their final value. The other two temporary images, λ^+ and λ^-, accumulate *upstream* and *downstream* lengths for each pixel. The upstream direction is given by the adjacency relation: for example when making horizontal paths, upstream is to the E, and the set { NE, E, SE } are the upstream neighbors. The downstream direction is the opposite direction (in this case to the West (W)). These images are both initialized to L, indicating that, for each pixel, at the initial gray level, it is possible to draw a path in either direction of length L. Since L is the target length, it is not necessary to accumulate any value larger than L. As the algorithm progresses, λ^+ and λ^- will decrease in value. For a pixel p, $\lambda^+(p) + \lambda^-(p) - 1$ is the length of the longest path through it.

The initialization of λ^+ and λ^- allows paths to extend indefinitely past the image boundary. In Appleton and Talbot's algorithm, pixels closer to the boundary obtain lower values. In this way paths are constrained to the image domain. There are other ways to treat the boundary condition [3], but this is not explored further in this paper. No boundary condition is correct if it is not known what was outside the image. Therefore it makes sense to use the boundary condition that requires the least effort to implement or the least time to compute. Furthermore, adding a dark border around the image also constrains the paths in the opening to the image domain, but in a much simpler manner (use a light border for the closing).

After initialization, all pixels with the lowest gray value are selected for update, and the temporary and output images are updated as detailed below. Then the next higher gray value is chosen, and all pixels with this value that are still active are selected for update, etc., until the highest gray value is reached. It is possible to write directly in the input image, it is not necessary to create a separate output buffer.

At each threshold level g, the selected pixels are processed twice, once in the upstream direction and once downstream. In each of these passes, upstream/downstream pixels are iteratively enqueued. Pixels on the queue need to be processed in the appropriate order. For example, when making horizontal lines, the upstream direction is E, meaning that in the upstream pass the enqueued pixels need to be processed from W to E. In the downstream pass, the enqueued pixels need to be processed from E to W.

In the upstream pass, for each of the selected pixels p the corresponding $\lambda^-(p)$ is set to 0 and the upstream neighbors are enqueued. Pixels in the queue are processed according to their position in the image, such that pixels further upstream are processed later. For each of the pixels q in the queue, the maximum λ^- of its downstream neighbors is found and increased by one. For horizontal paths this is:

$$\lambda = 1 + \max(\lambda^-(NW(q)), \lambda^-(W(q)), \lambda^-(SW(q))) \; .$$

If this new value λ is smaller than $\lambda^-(q)$, $\lambda^-(q)$ is assigned the value λ and its upstream neighbors are also enqueued. Because λ^- was initialized to L, the update procedure automatically stops after L steps.

In a second, downstream pass, λ^+ is updated in a similar way.

Next, For each of the pixels q updated in these two passes, the value $\lambda(q) = \lambda^+(q) + \lambda^-(q) - 1$ is computed. If $\lambda(q) < L$, the pixel q is not part of a path of length L. The corresponding pixel in the output image is set to the current gray value g. Because this pixel will also not be part of a longer path at higher gray values, $b(q)$ is set to inactive and $\lambda^+(q) = \lambda^-(q) = 0$.

Note how, after sorting the image pixels by gray value, there is no further comparisons of gray values. Therefore, to compute the closing instead of the opening, one need only change the initial sort order, starting at the largest gray value.

B. Simplifying the Algorithm

The most complex part of the algorithm as described above is the queue used to process the pixels: depending on the chosen orientation, pixels need to be sorted in different ways. This sorting can be accomplished by writing four versions of the algorithm, one for each possible orientation, but that does not generalize to higher dimensional images. At the expense of a slight increase in execution time, the above algorithm can be modified such that, instead of processing all pixels with identical gray value at the same time, the λ^+ and λ^- update procedure is performed for each input pixel independently. The pixel queue can now be implemented as a simple first-in, first-out (FIFO) queue. The recursive update algorithm only requires a list n of offsets to the upstream and downstream neighbors, such that, e.g. $p + n(1) = NW(p)$, $p + n(2) = W(q)$ and $p + n(3) = SW(q)$. By changing this neighbor list, a different path orientation can be selected. And by defining the neighbor list appropriately, the same algorithm can process images of any number of dimensions.

This implementation of the algorithm starts by creating a list of linear indices (or memory addresses) to every pixel in the image. It skips the border pixels to avoid tedious out-of-bounds tests when indexing neighbors. The list of indices is then sorted according to the gray value of the input image (low to high for the opening, high to low for the closing). The temporary images b, λ^+ and λ^- described above are created and initialized. b is set to inactive for all the border pixels to stop the iterative propagation routine when it reaches the image boundary. Two FIFO queues are created, Q_q and Q_c. Q_q is the queue used for propagating lengths. Q_c is a queue to which pixels are added for which either λ value changed, and avoids testing all pixels in the image for changes at the end of every length propagation pass. The binary image b is stored using one byte per pixel, which lets two additional flags per pixel to be stored in the same space: f_q and f_c. When a pixel is added to Q_q, f_q is set, indicating that the pixel need not be enqueued again; when the pixel is popped from the queue, f_q is reset. f_c has the same function for queue Q_c. These two flags are only used for efficiency, and could be done without at the cost of longer queues Q_q and Q_c. The full algorithm is outlined in Figure 2.

By not processing pixels on the boundary of the image, the algorithm never changes their value. What is more, the pixels on the image boundary do not influence the result of the operation. To circumvent this issue, it is possible to add a one-pixel border around the image.

C. Adjacency Relations in d Dimensions

In the previous algorithm description I did not specify how to create the offsets to the upstream and downstream neighbors, n^+ and n^-. This list is the only element of the algorithm that contains any notion of the image dimensionality. Let us assume that the image is stored in a linear array such that adding the integer value s_1 to the address of any pixel yields the address of its neighbor along the first dimension, adding s_2 yields the address of the neighbor in the second dimension, etc. The values s_i are called *strides*. In a 2D image, for example, the NE neighbor of p is $p+s_1-s_2$, and therefore the value s_1-s_2 is the offset to the NE neighbor. This indexing does not work on border pixels, which is why the border pixels are not processed in the algorithm as described above. It is relatively easy to add the necessary tests to be able to process border pixels, but these significantly increase the execution time of the algorithm.

Figure 3 outlines an algorithm to loop over every neighbor of a pixel, no matter what the dimensionality d is, and compute its offset based on the strides s. It includes a test "w is close in direction to v", to determine if a neighbor w needs to be added to the list of upstream neighbors n^+, given the path's main direction v. Here, v and w are defined as vectors with d elements, each one taken from the set $\{-1, 0, 1\}$, and point at a neighbor pixel. The test "w is close in direction to v" is defined as follows: (*1*) $w_i \neq v_i$ for at least one i; (*2*) $w_i = v_i \neq 0$ for at least one i; and (*3*) $|w_i - v_i| \leq 1$ for all values of i. In 2D this produces the same neighbor relations as in the original path opening definition [1]–[5]. In higher dimensions it produces an equivalent graph with paths that can bend in any direction.

D. Enumerating All Possible Orientations

To obtain a path opening that is not constrained to a specific orientation, one would take the supremum of the path openings in each possible orientation. In 2D there are 4 possible orientations: $0°$, $90°$, $45°$ and $-45°$. In d dimensions the number of orientations is given by $N_d = (3^d - 1)/2$ (this is half the number of neighbors of a pixel). Enumerating these orientations is very simple, using a loop similar to that shown in Figure 3. This time we do a simple test on w to verify that it is on the positive half-sphere: the first non-zero element of w must be positive.

III. Constraining Path Openings

The one issue with path openings using the adjacency relations as defined previously concerns the length of diagonal paths ($45°$ and $-45°$ paths). An opening will preserve a two-pixel wide line of certain length, but not a one-pixel wide line of the same length, because it can zig-zag in the wider line but not in the thinner one. This effect is demonstrated in Figure 4 and Subsection IV-C.

To reduce a path's ability to zig-zag, it is possible to change the adjacency graph. For example, Heijmans et al. mention the possibility of an adjacency graph where even and odd steps are different [3]. Their example graph [3, Figure 2d] forms vertical paths from S to N; even rows are as described here, containing steps either NW, N or NE; odd rows contain only steps in the main direction, N. This results in lines with at least half the steps to the N, meaning its angle is more constrained and zig-zagging is reduced. The negative side effect is that these openings are not invariant to a translation of 1 pixel

create list of offsets n^+ to upstream neighbors and n^- to downstream neighbors
create a list i of indices to every pixel in the image I (except border pixels)
sort i according to value of $I(i)$
create temporary images b, λ^+ and λ^-
initialize: $b \leftarrow$ true, $\lambda^+ \leftarrow L$, $\lambda^- \leftarrow L$
initialize: $b(p_b) \leftarrow$ false (for all border pixels p_b)
for every element p in i for which $b(p) =$ true
    propagate (p, λ^-, n^+, n^-)
    propagate (p, λ^+, n^-, n^+)
    for every element q in Q_c **:**
        if $\lambda^+(q) + \lambda^-(q) - 1 < L$ **:**
            $I(q) \leftarrow I(p)$
            $b(q) \leftarrow$ false, $\lambda^+(q) \leftarrow 0$, $\lambda^-(q) \leftarrow 0$

function propagate (p, λ, n_f, n_b) **:**
    $\lambda(p) \leftarrow 0$
    enqueue in Q_q all neighbors $p_f = p + n_f$ for which $b(p_f) =$ true
    for every element q in Q_q **:**
        $\ell \leftarrow \bigvee_i \lambda\,(q + n_b(i)) + 1$
        if $\ell < \lambda(q)$ **:**
            $\lambda(q) \leftarrow \ell$
            enqueue in Q_q all neighbors $q_f = q + n_f$ for which $b(q_f) =$ true
            enqueue q in Q_c

Fig. 2. Dimensionality-independent version of the path opening algorithm. See text for definition of variables.

create an empty list n^+
create coordinate array w, with d elements, initialized to -1
loop indefinitely :
    if $w = v$ **or** w is close in direction to v **:**
        $p = 0$
        for every i in $(1, d)$ **:** $p \leftarrow p + v_i s_i$
        add the offset p to the list n^+
    for every i in $(1, d)$ **:** *(find the coordinates for another neighbor)*
        $w_i \leftarrow w_i + 1$
        if $w_i \leq 1$ **: break** *(we have found a new neighbor to process)*
        $w_i \leftarrow -1$
    if $w_i = -1\ \forall i$ **: break** *(we have processed all neighbors)*
$n^- \leftarrow -n^+$

Fig. 3. Dimensionality-independent looping over all neighbors to create a list of offsets to neighboring pixels. See text for definition of variables.

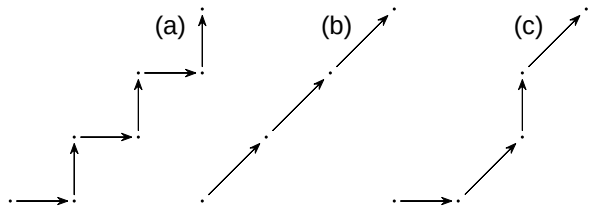

Fig. 4. The $45°$ path can have very different number of pixels depending on the width it is allowed to have. *a*: When given the space, the path will zig-zag, resulting in a physically shorter path than expected given the pixel count (7 pixels). *b*: The same physical length can be covered with fewer pixels (4 pixels). *c*: The proposed constraint does not completely avoid the zig-zagging of the path, but reduces its consequences to acceptable levels (5 pixels).

in the vertical direction, though translation invariance can be recovered by combining the output of two openings.

I propose to constrain paths in a similar manner, but within a single operation. When building the path with main direction v, a step in a direction other than v must always be followed by a step v. This is a less strict constraint than using the graph described above because the restriction to the step is not given by the location in the image, but rather by the previous step taken. Figure 4c gives an example. With this constraint the steps in a direction other than v can happen anywhere on the path, but it is not possible for two of these steps to happen consecutively.

To implement this idea two additional temporary images are required: instead of one upstream and one downstream length image, λ^+ and λ^-, we need two of each, one for the *normal* length, and one for the *constrained* length: λ^+, λ_c^+, λ^- and λ_c^-. The constrained lengths are the lengths propagated from the pixel in the main direction v, the normal length can be propagated from any of the possible directions. When propagating lengths, the constrained length can be propagated to any direction, whereas the normal length can only be propagated in the main direction. This implies that the constrained length must be used to update the next pixel's normal length, and the normal length must be used to

create list of offsets n^+ to upstream neighbors and n^- to downstream neighbors
create a list i of indices to every pixel in the image I (except border pixels)
sort i according to value of $I(i)$
create temporary images b, λ^+, λ_c^+, λ^- and λ_c^-
initialize: $b \leftarrow$ true, $\lambda^+ \leftarrow L$, $\lambda_c^+ \leftarrow L$, $\lambda^- \leftarrow L$, $\lambda_c^- \leftarrow L$
initialize: $b(p_b) \leftarrow$ false (for all border pixels p_b)
for every element p in i for which $b(p) =$ true
    propagate (p, λ^-, λ_c^-, n_c^+, n^+, n_c^-, n^-)
    propagate (p, λ^+, λ_c^-, n_c^-, n^-, n_c^+, n^+)
    for every element q in Q_c **:**
        if $\lambda^+(q) + \lambda_c^-(q) - 1 < L$ **or** $\lambda_c^+(q) + \lambda^-(q) - 1 < L$ **:**
            $I(q) \leftarrow I(p)$
            $b(q) \leftarrow$ false, $\lambda^+(q) \leftarrow 0$, $\lambda_c^+(q) \leftarrow 0$, $\lambda^-(q) \leftarrow 0$, $\lambda_c^-(q) \leftarrow 0$

function propagate (p, λ, λ_c, $n_{f,c}$, n_f, $n_{b,c}$, n_b) **:**
    $\lambda(p) \leftarrow 0$, $\lambda_c(p) \leftarrow 0$
    enqueue in Q_q all neighbors $p_f = p + n_f$ for which $b(p_f) =$ true
    for every element q in Q_q **:**
        $\ell \leftarrow \lambda\left(q + n_{b,c}\right) + 1$
        if $\ell < \lambda_c(q)$ **:**
            $\lambda_c(q) \leftarrow \ell$
            enqueue in Q_q all neighbors $q_f = q + n_f$ for which $b(q_f) =$ true
            enqueue q in Q_c
        $\ell \leftarrow \ell \vee \left\{\bigvee_i \lambda_c\left(q + n_b(i)\right) + 1\right\}$
        if $\ell < \lambda(q)$ **:**
            $\lambda(q) \leftarrow \ell$
            enqueue in Q_q neighbor $q_f = q + n_{f,c}$ if $b(q_f) =$ true
            enqueue q in Q_c

Fig. 5. Algorithm for the constrained path opening, compare to Figure 2. See text for definition of variables.

update the next pixel's constrained length. Because constrained and normal lengths alternate, it is necessary to add normal upstream and constrained downstream lengths (and vice versa) when computing the total length of a path through a point p. This length is thus given by

$$\lambda(q) = \left\{\lambda^+(p) + \lambda_c^-(p) - 1\right\} \vee \left\{\lambda_c^+(p) + \lambda^-(p) - 1\right\} \ .$$

Furthermore, it can be shown that $\lambda^+ \geq \lambda_c^+$ and $\lambda^- \geq \lambda_c^-$. The modified algorithm is given in Figure 5.

IV. Results

This section contains some experimental results that quantify and compare the performance of the path openings as described in this paper, and illustrate some possible uses for this operation.

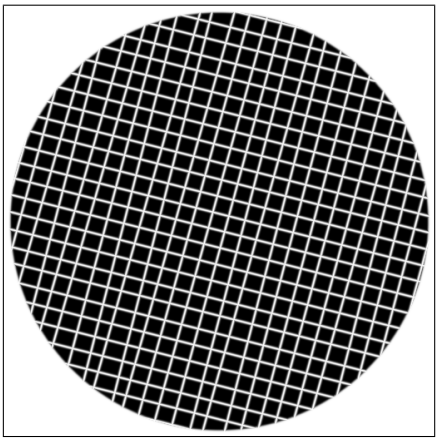

Fig. 6. Example input image used for the results in Figure 7.

A. Time versus Accuracy

Path openings have many different possible applications. The most obvious one is to filter the image, preserving line-like features while removing other features. The path opening has to be applied once for each of the directions described in Subsection II-D, which is 4 in 2D. The other method to accomplish this, using openings with straight line segments, becomes more accurate with increasing number of orientations used [9]. Non-morphological methods such as the structure tensor [12] or directional second derivatives (including steered filters [12]) are often employed to detect linear features in images. There is, however, a significant difference between *detecting* and *preserving*: the morphological filters will keep the desired features unchanged while removing the non-desired ones, whereas the non-morphological filters will yield a strong response at the desired features, not preserving their intensities nor shapes. Therefore I limited the following evaluation to the morphological methods.

To answer the question of how the path opening compares in quality to openings with straight line segments at the same execution time, and how the path opening compares in execution time to openings with straight line segments at equal quality, the following experiment was carried out. Fifty synthetic images were generated as follows, see Figure 6. A grid of perpendicular lines cover the image. Each line was given a small random rotation (between 0° and 1.9°). The

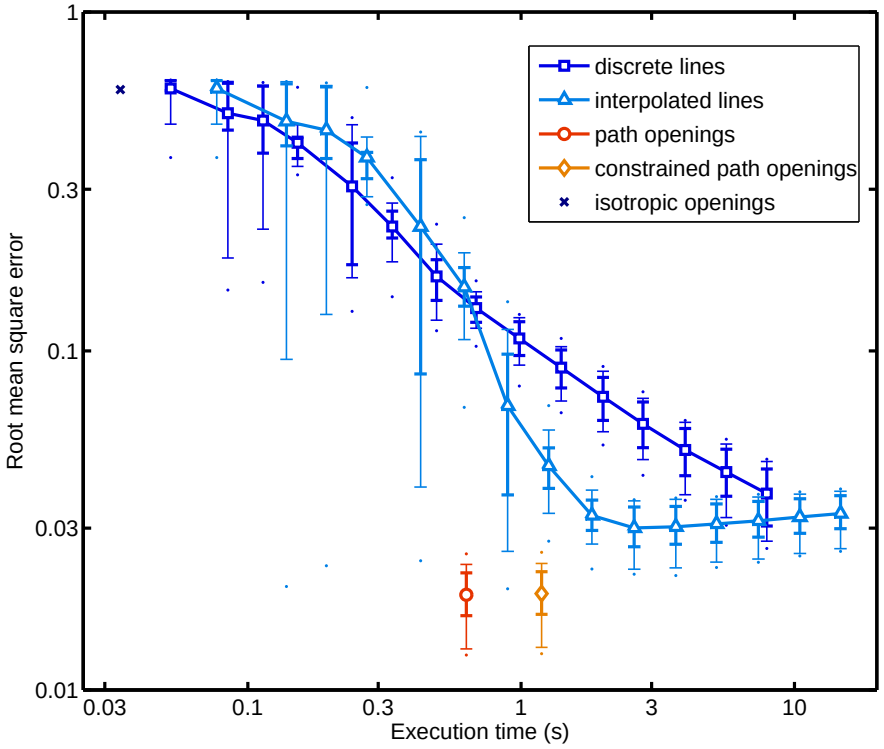

Fig. 7. Accuracy versus computation time of the various options to filter an image preserving linear structures. Values plotted are the mean over 50 repetitions. The thick bars indicate the 25^{th} and 75^{th} percentiles, the thin bars the 10^{th} and 90^{th} percentiles, and the dots to minimum and maximum errors. Isotropic openings are included to show the upper limit on the error (cross at the top left corner of the graph). For openings with discrete and interpolated lines, the plot shows the result using $\text{round}(2^i)$ orientations, with i increasing linearly from 2 to 9 in steps of 0.5.

whole grid was rotated randomly, but avoiding angles close to $0°$, $45°$ and $90°$. Lines were given a Gaussian profile ($\sigma = 1.2$ pixels). Note that the lines were directly sampled in their final position in the image, there was no resampling of the image to obtain the rotations. No noise was added. A region around the boundary of the image is set to a high gray value to avoid edge effects. The images were 512 by 512 pixels in size.

Both the path openings ($L = 50$ pixels) and the openings with straight line segments (length 50 pixels) were applied to each image, and compared to the input image. The error measure used was the root mean square error, computed only on and close to the lines of the input image. The average error for each filter is plotted against computation time in Figure 7. Openings with straight line segments were computed using between 4 and 512 steps, and using two methods to compute the opening: discrete lines and interpolated lines [9]. The graph includes results for both the standard path opening and the constrained path opening. Constraining the path approximately doubles execution time, but does not affect the error measure in this experiment. The graph also includes the result for the isotropic opening (area equivalent to that of the line segments). The isotropic openings, not preserving linear structures, gives an upper bound for the error.

In this experiment, path openings produce results much better than those that can be produced with openings with straight line segments. The interpolated lines method obtains a minimum at 91 orientations (the method does not improve much with more orientations, but the interpolation errors keep accumulating, thereby slightly increasing the error measure). At 91 orientations, the error is 50% larger than for the path openings, and the computation time four times as long as that of the path openings. The difference in execution time increases for increasing length, and decreases for decreasing length.

B. Detecting Lines Without Knowing Their Exact Orientation

In this example application, I apply the path opening filter to detect linear elements. Figure 8a shows a photograph of a printed circuit board. A simple top-hat filter [6] filters the image so that only thin elements remain, independent of their length or orientation. The constrained path opening is then applied in all four directions (Figure 8b-d), leaving only elongated thin elements that are approximately horizontal, vertical and diagonal. Note how the exact orientation of a line segment is irrelevant. Most notably, this makes the algorithm insensitive to distortions such as a small rotation, projective distortion and lens distortions. In contrast, using straight structuring elements requires exactly matching the orientation of the structuring element with the orientation of the lines to be detected. In Figure 8e-f these straight line operators failed to correctly filter the image because none of the lines in the image are exactly horizontal or vertical.

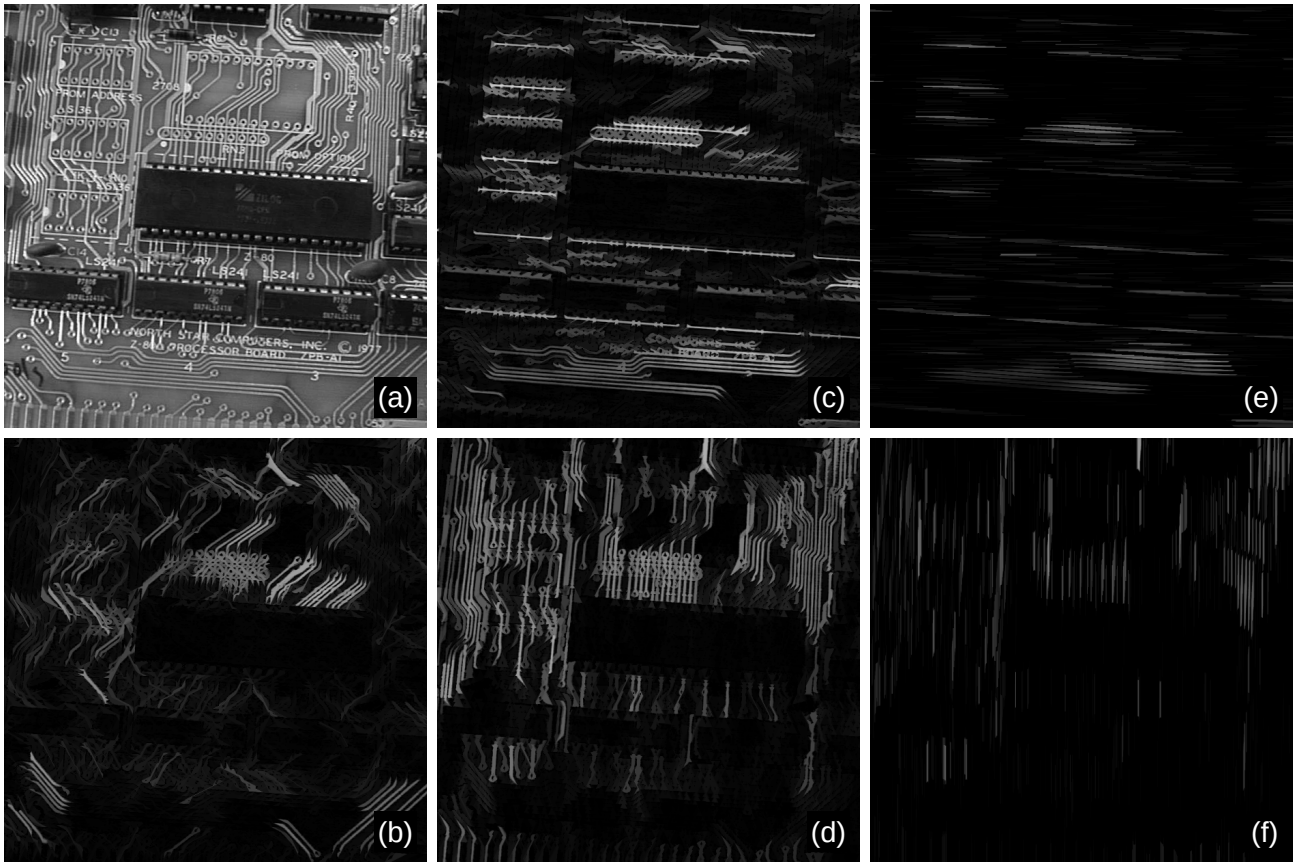

Fig. 8. Line detection. *a*: Input image, photograph of a printed circuit board. *b*: Image filtered for diagonal lines using the supremum of the path openings at 45° and -45°. *c*: Image filtered for horizontal lines using a path opening at 0°. *d*: Image filtered for vertical lines using a path opening at 90°. *e*: Image filtered for horizontal lines using an opening with a straight line segment at 0°. *f*: Image filtered for vertical lines using an opening with a straight line segment at 90°.

C. Improved Length Estimation

This section studies the applicability of the path opening to construct a granulometry. For accurately measuring length using a granulometry, the opening (or closing) filter must remove all line segments shorter than a specified length ℓ, and not affect the ones longer than ℓ. The first problem that one notices when studying the path openings is that diagonal paths are longer than horizontal paths with the same parameter L, since this parameter specifies the number of pixels in the path, not its length. It is conceivable to modify the path opening algorithm to more accurately measure path lengths, using existing perimeter estimation algorithms [13], [14]. Here I will ignore this problem, since, as shown in Section III, the zig-zag of the diagonal path opening introduces a much larger bias. This means that, in practice, diagonal paths are much shorter than horizontal paths with the same parameter L. The constrained path opening reduces this problem but does not eliminate it. As shown in the following experiment, the zig-zag in the constrained path opening approximately balances the incorrect length measure used in the algorithm, producing a reasonable length estimate. There is no justification for the balancing of these two effects; it just happens that counting the pixels of a line that slightly zig-zags as does the constrained path yields a reasonable estimate of the physical length of the path.

Eight images as in Figure 9 were generated. Each image contains 50 lines of 40 pixels length at the same orientation. Eight orientations were chosen at equal intervals between 0° and 45°. Each line has a Gaussian profile ($\sigma = 1$) and a random sub-pixel shift, as in reference [9].

The cumulative length distributions for these eight images was computed with a granulometry, following reference [7], using one of the following three operations at each length scale ℓ: (*1*) the supremum of 4 unconstrained path openings with $L = \ell$, (*2*) the supremum of 4 constrained path openings with $L = \ell$, and (*3*) the supremum of $2\lceil \pi \ell \rceil$ openings with straight line segments (by interpolation) of length ℓ [9]. The length scale axis is sampled in increments of four pixels. The results are shown in Figure 10. Ideally, the cumulative distribution is zero for lengths ℓ smaller than the length of the lines in the image (40 pixels), and one for larger ℓ. Because of the Gaussian profile of the lines and inaccuracies in the operations, small deviations are expected; a perfect result is not possible. For the horizontal lines, all three methods produce a correct output, as expected. However, as the angle increases the unconstrained path opening yields an increasingly severe overestimation of the lengths of the lines (Figure 10a), due to the zig-zagging discussed in Section III. By constraining the paths as proposed in this paper, the length of diagonal lines is measured much more accurately (Figure 10b). Only the

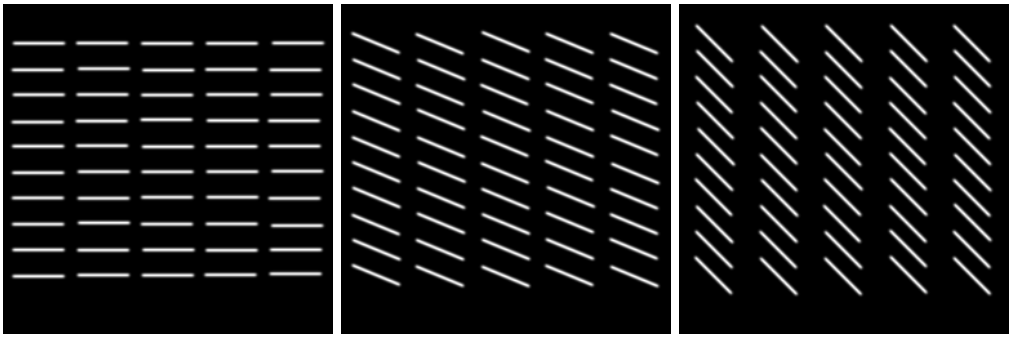

Fig. 9. Three of the images used as input for the granulometries in Figure 10.

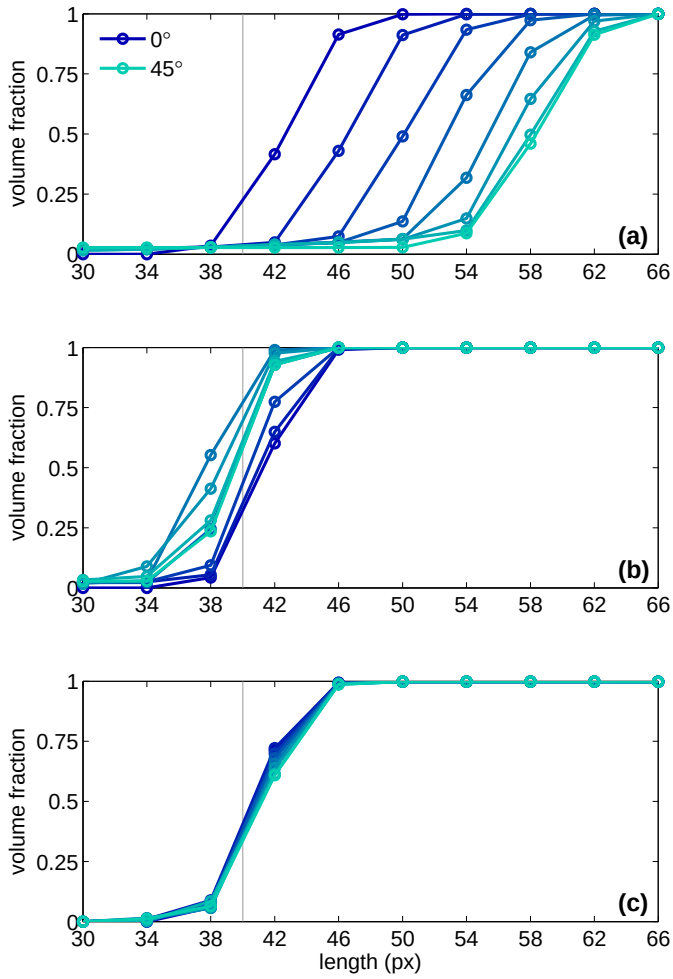

Fig. 10. Length granulometries for the two images in Figure 9. *a*: Granulometry using unconstrained path openings. *b*: Granulometry using the constrained path openings proposed here. *c*: Granulometry using the supremum over openings with straight line segments at many angles.

more computationally expensive and non-robust third method is rotationally invariant (Figure 10c). Very similar results were obtained with wider line segments in the input images (not shown).

D. Length Distribution of Wood Fibers in 3D Images

Next I applied the granulometry with path openings on a μCT image to demonstrate its usefulness for estimating length distributions in 3D images. Thermomechanically pulped (TMP) wood fibers were mixed with polylactide (PLA) as matrix, at 10% weight fraction of fibers, and injection molded. A small cube of approximately 1 mm^3 was cut from the sample and imaged at the TOMCAT beamline at the Swiss Light Source (Paul Scherrer Institut, Villigen, Switzerland). The resulting volumetric image was cropped to 724^3 pixels and further downsampled to 362^3 pixels. Figure 11a shows the middle slice from this volume, a little over 500 μm across.

The granulometry requires a uniform gray value over the objects and the background if the goal is to obtain a size (or, as in this case, length) distribution [7]. Binarizing the image accomplishes this, but makes a hard decision as to which pixels are fiber and which ones are matrix. Instead, I choose to use clipping to make the foreground and background gray values uniform without modifying the gray values of the pixels at the edges of the objects. Using gray values in this way improves the results of the granulometry [15]. Before clipping the image to the range $[62, 123]$, a bilateral filter (separable implementation [16], spatial σ of 1 pixel, tonal σ of 10) was applied to reduce noise. The result is shown in Figure 11b. This image is the input to the granulometry using constrained path openings, as used in Subsection IV-C. Figure 11c-f shows the middle slice of the filtering result at various length scales, demonstrating the increasing removal of fibers from the volume. Note that, because the fibers are not aligned with the cutting plane, their lengths are not evident from the slice shown.

Figure 12 shows the cumulative distribution (line plot, right vertical axis) for the image in Figure 11b, sampled twice per octave. The bar graph in that figure is the derivative of the cumulative distribution, and thus an estimate of the volume-weighted distribution of fiber lengths in the sample. Note however, that because of the boundary condition chosen (fibers touching the image boundary are considered infinitely long and do not contribute to the distribution), the distribution underestimates the weight of the longer fibers. An unbiased measurement requires a complex boundary condition, as is known from stereology [17].

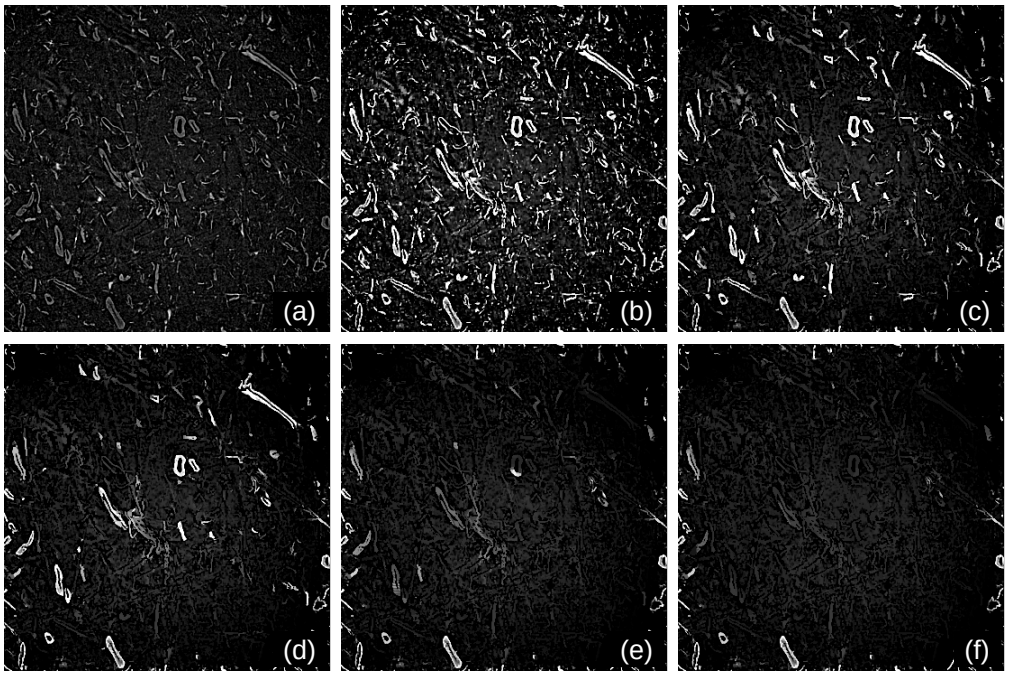

Fig. 11. 3D μCT image of wood fibers embedded in plastic. *a*: Slice of the input image. *b*: Slice of the preprocessed image, to which the granulometry is applied. *c*: Slice from the result of the path opening, at $L = 32$ pixels ($\sim$45 μm); *d*: at $L = 64$ pixels ($\sim$90 μm); *e*: at $L = 128$ pixels ($\sim$179 μm); and *f*: at $L = 256$ pixels ($\sim$358 μm).

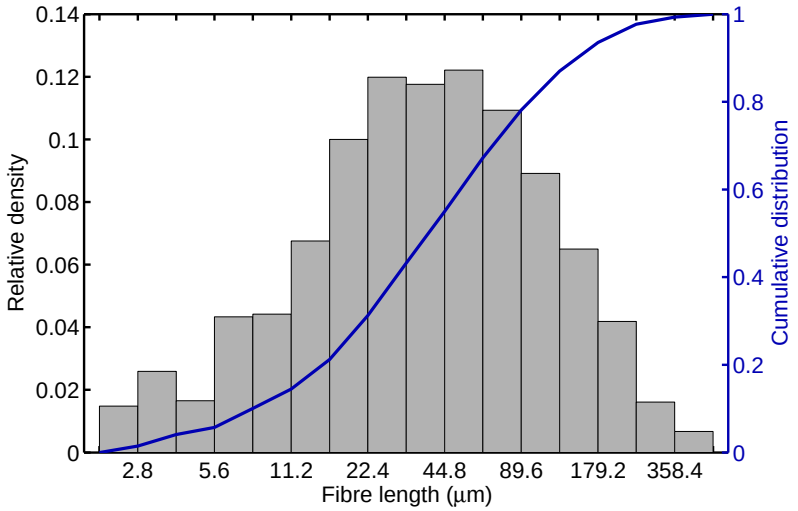

Fig. 12. Cumulative distribution (line plot and right vertical axis) of fiber lengths in the volume image shown in figure 11, and its derivative (bar graph and left vertical axis). The bar graph corresponds approximately to a log-normal distribution.

V. Conclusions and Discussion

The algorithm presented here increases the computational cost of the algorithm as presented by Appleton and Talbot. However, this increase is limited because the processing done for one pixel potentially fixes the output value of many pixels, thereby reducing the work needed for subsequent pixels at the same gray level. In return for the small increase in cost, the algorithm becomes simpler to implement, is applicable to images of any number of dimensions, and is applicable to floating-point images (the original version of the algorithm assumes a limited set of gray values in the input image).

This opening algorithm processes gray values from lowest to highest. In contrast, an efficient algorithm to compute the area opening processes gray values from highest to lowest (the down-hill algorithm) [18], [19]. Because the area opening is similar to the path opening, it seemed at first that it should be possible to write a similar algorithm for the path opening as well. This turned out to not be possible because the path opening does not consider the whole connected component as a single unit: different parts of a connected component can obtain different gray values after the path opening operation. This is incompatible with the down-hill algorithm.

As mentioned earlier, the pixels at the border of the image are not processed in order to keep the algorithm simple.

Furthermore, due to the simple initialization, paths can extend indefinitely past the image boundary. This means that any path that goes to the edge is assumed to be infinite in length. Both these limitations can be overcome by adding a border around the image. To limit paths to the image domain, a dark border must be added for the opening, or a light one for the closing.

The second modification proposed in this paper is the constraining of the paths. By alternating between two connectivity graphs, diagonal lines can not zig-zag as much. This significantly improves the length accuracy of the method, and thereby also the rotation invariance. This constraint increases execution time and memory usage, but does not make the algorithm much more difficult to implement. Additionally, the constrained paths are more selective in orientation than the unconstrained paths: the unconstrained path opening is not able to separate the diagonal lines from either the horizontal or vertical lines. Whether this point is positive of negative depends on the application. A different way of improving length accuracy is by using a more correct length measure. However, using any length measure other than counting pixels would require a much more complex and costly algorithm.

In this paper it is proposed to use the supremum over all path openings instead of the supremum over many openings with straight line segments at different orientations. Path openings produce better results, even if the lines in the input image are straight, because they do not need to exactly match the orientation of the lines. Path openings are also faster for longer lines, due to the very large number of orientations required when using long lines (though for very short lines they might be less efficient). The higher the image dimensionality, the more time is saved by using path openings. An additional advantage is that path openings are insensitive to a small bending of the lines in the input. Such bending can occur e.g. due to lens aberrations, or can be inherent to the data, such as in the wood fiber example of Subsection IV-D. In this example, fibers can be bent or partly broken, yet still need to be measured correctly. Using straight line segments, a bent fiber would be broken into shorter sections in which straight lines do fit.

ACKNOWLEDGMENT

The author thanks Hugues Talbot for kindly making his path opening code available. The μCT image of wood fibers shown in Figure 11 was acquired on the TOMCAT beam line at the Swiss Light Source, Paul Scherrer Institut, Villigen, Switzerland; thanks to all the people in the WoodFibre3D project for this image. The WoodFibre3D project is funded by the European Union through the WoodWisdom-Net programme.

At the risk of being cliché, I also want to thank the reviewers, who read the manuscript very thoroughly and whose comments and suggestions improved this paper greatly.

REFERENCES

[1] M. Buckley and H. Talbot, "Flexible linear openings and closings," in *Mathematical Morphology and its Applications to Image and Signal Processing*, ser. Computational Imaging and Vision, J. Goutsias, L. Vincent, and D. S. Bloomberg, Eds., vol. 18. New York: Springer, 2000, pp. 109–118.

[2] H. Heijmans, M. Buckley, and H. Talbot, "Path-based morphological openings," in *2004 International Conference on Image Processing (ICIP)*, vol. 5. IEEE, 2004, pp. 3085–3088.

[3] ——, "Path openings and closings," *Journal of Mathematical Imaging and Vision*, vol. 22, no. 2-3, pp. 107–119, 2005.

[4] B. Appleton and H. Talbot, "Efficient path openings and closings," in *Mathematical Morphology: 40 Years On (Proceedings of the 7th International Symposium on Mathematical Morphology)*, ser. Computational Imaging and Vision, C. Ronse, L. Najman, and E. Decencière, Eds., vol. 30, 2005, pp. 33–42.

[5] H. Talbot and B. Appleton, "Efficient complete and incomplete path openings and closings," *Image and Vision Computing*, vol. 25, no. 4, pp. 416–425, 2007.

[6] P. Soille, *Morphological Image Analysis: Principles and Applications*, 2nd ed. Berlin: Springer, 2003.

[7] C. L. Luengo Hendriks, G. M. P. van Kempen, and L. J. van Vliet, "Improving the accuracy of isotropic granulometries," *Pattern Recognition Letters*, vol. 28, no. 7, pp. 865–872, 2007.

[8] C. L. Luengo Hendriks and L. J. van Vliet, "A rotation-invariant morphology for shape analysis of anisotropic objects and structures," in *Proceedings of the Fourth International Workshop on Visual Form*, ser. Lecture Notes in Computer Science, C. Arcelli, L. P. Cordella, and G. Sanniti di Baja, Eds., vol. 2059. Berlin: Springer-Verlag, 2001, pp. 378–387.

[9] ——, "Using line segments as structuring elements for sampling-invariant measurements," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1826–1831, 2005.

[10] S. Valero, J. Chanussot, J. A. Benediktsson, H. Talbot, and B. Waske, "Directional mathematical morphology for the detection of the road network in very high resolution remote sensing images," in *Proceedings of ICIP*. IEEE, 2009, pp. 3725–3728.

[11] M. Buckley and J. Yang, "Regularised shortest-path extraction," *Pattern Recognition Letters*, vol. 18, no. 7, pp. 621–629, 1997.

[12] B. Jähne, *Digital Image Processing*, 5th ed. Berlin: Springer, 2002.

[13] L. Dorst and A. W. M. Smeulders, "Length estimators for digitized contours," *Computer Vision, Graphics and Image Processing*, vol. 40, no. 3, pp. 311–333, 1987.

[14] D. Coeurjolly and R. Klette, "A comparative evaluation of length estimators of digital curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 252–258, 2004.

[15] C. L. Luengo Hendriks, "Structure characterization using mathematical morphology," Ph.D. dissertation, Delft University of Technology, Delft, The Netherlands, 2004. [Online]. Available: http://www.qi.tnw.tudelft.nl/Publications/phd_theses.html

[16] T. Q. Pham and L. J. van Vliet, "Separable bilateral filtering for fast video preprocessing," in *IEEE International Conference on Multimedia and Expo (Amsterdam, July 6-8)*. Los Alamitos, CA, USA: IEEE Computer Society, 2005.

[17] P. R. Mouton, *Principles and Practices of Unbiased Stereology: an Introduction for Bioscientists*. Baltimore: Johns Hopkins University Press, 2002.

[18] L. Vincent, "Grayscale area openings and closings, their efficient implementation and applications," in *Mathematical Morphology and Its Applications to Signal Processing*, J. Serra and P. Salembier, Eds., EURASIP. Barcelona, Spain: UPC Publications, 1993, pp. 22–27.

[19] K. Robinson and P. F. Whelan, "Efficient morphological reconstruction: a downhill filter," *Pattern Recognition Letters*, vol. 25, no. 15, pp. 1759–1767, 2004.