A Faster, Unbiased Path Opening by Upper Skeletonization and Weighted Adjacency Graphs

Teo Asplund and Cris L. Luengo Hendriks, *Senior Member, IEEE*

***Abstract*— The path opening is a filter that preserves bright regions in the image in which a path of a certain length L fits. A path is a (not necessarily straight) line defined by a specific adjacency relation. The most efficient implementation known scales as $\mathcal{O}(\min(L, d, Q)N)$ with the length of the path, L, the maximum possible path length, d, the number of graylevels, Q, and the image size, N. An approximation exists (parsimonious path opening) that has an execution time independent of path length. This is achieved by preselecting paths, and applying 1D openings along these paths. However, the preselected paths can miss important structures, as described by its authors. Here, we propose a different approximation, in which we preselect paths using a grayvalue skeleton. The skeleton follows all ridges in the image, meaning that no important line structures will be missed. An H-minima transform simplifies the image to reduce the number of branches in the skeleton. A graph-based version of the traditional path opening operates only on the pixels in the skeleton, yielding speedups up to one order of magnitude, depending on image size and filter parameters. The edges of the graph are weighted in order to minimize bias. Experiments show that the proposed algorithm scales linearly with image size, and that it is often slightly faster for longer paths than for shorter paths. The algorithm also yields the most accurate results—as compared with a number of path opening variants—when measuring length distributions.**

***Index Terms*— Path opening, granulometry, length distribution, image analysis, line segment, unbiased, mathematical morphology.**

I. Introduction

THE path opening is one of the many solutions to the line detection problem [3], which deals with enhancing long, thin structures in an image. This problem appears in a wide variety of applications, e.g. detecting blood vessels in eye fundus images [20], roads in remote sensing [27], tracks of dust devils on Mars [25], measuring wood fibers [13], etc. A straightforward approach is to take the supremum of openings using lines of different orientations as structuring elements. This approach, though slow, works well if the features of interest consist of straight line segments [23]. However, for curved, eccentric structures, the results are unsatisfactory.

Manuscript received March 11, 2016; revised August 8, 2016; accepted September 5, 2016. Date of publication September 15, 2016; date of current version October 5, 2016. The work of T. Asplund was supported by the Swedish Research Council under Grant 2014-5983. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Martin Kleinsteuber.

T. Asplund is with the Centre for Image Analysis, Uppsala University, SE-751 05 Uppsala, Sweden.

C. L. Luengo Hendriks is with Flagship Biosciences, Inc., Westminster, CO 80021, USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2016.2609805

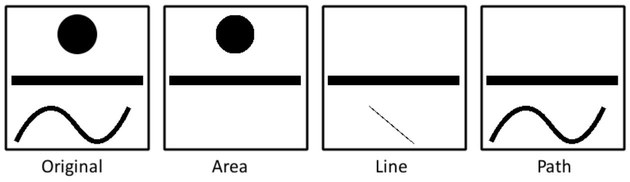

Fig. 1. The original image contains two long, thin structures that should be preserved by the opening, whereas the compact noise should be removed. The area opening cannot distinguish long, thin structures from compact ones of a similar number of pixels, whereas the line opening fails to preserve thin, sinuous structures. The path opening, however, successfully preserves the desired structures, while removing noise. A similar result may be achieved using elongation, however using this attribute is problematic if there are crossings. Moreover, it is expensive.

Another approach is to use so called attribute openings [2]. One example is the area opening, which removes connected components whose area (number of pixels) is less than some threshold. This will, in general, not give the desired result, since short compact features will be preserved as well as the long, thin structures of interest. Another attribute that has been proposed for this problem is elongation. However, elongation is not increasing [14], as required to form an opening, and thus leads to other problems. In particular, algorithms using non-increasing criteria are more complex and expensive. Additionally, using elongation as an attribute becomes problematic if there are crossings in the image. The path opening is a kind of opening that tackles the problem by removing bright features in which a path of length L does *not* fit. Figure 1 shows an (inverted) image where the path opening successfully solves the problem, while other openings fail.

The path opening was first presented by Buckley and Talbot in 2000 [3]. Since then, several efficient algorithms have been proposed to compute it. The algorithm by van de Gronde *et al.* [29] is the fastest algorithm that produces exact results, although its time complexity is the same as Appleton and Talbot's algorithm [1], [26]. It runs in $\mathcal{O}(\min(L, d, Q)N)$ time [29], where L is the chosen length threshold, d is the maximum path length (often approximately $\sqrt{N}$), Q is the number of graylevels, and N is the size of the image. Another recently proposed algorithm is the so called parsimonious path opening (PPO) by Morard *et al.* [15]. The PPO tries to approximate the path opening in a time that is independent of L, by preselecting relevant paths in the image and applying a fast 1D-opening on the preselected paths. Three strategies for preselection are proposed, each having pros and cons. However, no matter the strategy, structures in the image may occlude other structures during the path selection step [15].

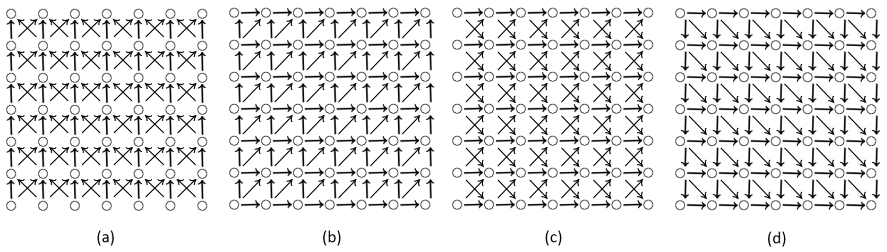

Fig. 2. The four adjacency graphs used for the path opening algorithm. Graph (a) shows the S-N adjacency graph, graph (b) shows the SW-NE graph, (c) is the W-E adjacency graph, and (d) is the NW-SE adjacency graph.

Both algorithms suffer from biased length measurements. The traditional path opening has a tendency to zig-zag in structures wider than one pixel, thus over-estimating the length. The PPO on the other hand, tends to underestimate lengths because paths are preselected in such a way that parts of the image may be occluded by bright structures. Both algorithms also use biased weights when estimating lengths.

The constrained path opening (CPO) [11] is a variant of the traditional path opening that addresses the problem of zig-zagging and measures lengths more accurately, however, it is slightly slower than the traditional path opening.

This paper proposes a new way of preselecting paths that avoids the problem of occlusion while also inhibiting zig-zagging, thus providing more accurate measurements while still providing a significant increase in speed due to the decreased number of possible paths. The algorithm makes use of the upper skeleton [30] to find important paths. By skeletonizing an image, its important parts are preserved as bright ridges (since we are looking to preserve thin, bright structures), setting the rest of the pixels to zero. These ridges, then, are the preselected paths. A graph is constructed, taking the non-zero pixels as nodes, and a modified path opening is used to open this graph. The result can then be reconstructed, as in the PPO, using the original image as a mask. This procedure results in an approximation of the path opening of the original image. Importantly, the paths selected by the upper skeleton are not occluded by bright structures. Moreover, using appropriate weights, the lengths measured in our experiments by this algorithm are less biased than for any other known path opening variant.

II. Path Opening

In this section a brief description of the path opening first proposed by Buckley and Talbot [3] is presented. For more detail see also Heijmans *et al.* [9], and Talbot and Appleton [26].

A. Adjacency Graphs

The path opening is the result of combining a family of paths into an algebraic opening. These paths are defined by adjacency graphs. Four adjacency graphs are traditionally used (Fig. 2).

These adjacency graphs define adjacency relations between pixels in the image. Let $I \subset \mathbb{Z}^2$ be a set of pixel locations. A given adjacency graph G defines an adjacency relation on I. Two elements $x, y \in I$ are adjacent if (x, y) forms a (directed) edge in G.

For an adjacency graph $G = (V, E)$ and a set of image locations I, let

$$\nu_G(x) = \{y \in I : (x, y) \in E\},$$

be the set of all neighbors of x. Then, a tuple $a = (a_1, a_2, \ldots, a_L)$ is a path of length L if $a_{k+1} \in \nu_G(a_k)$, for all $k \in \{1, 2, \ldots, L-1\}$. Let $\Pi_L^G(X)$ denote the set of all paths of length L in $X \subset I$ using the adjacency relation specified by G.

B. Binary Path Openings

For a path $a = (a_1, \ldots, a_L)$, let $\sigma(a) = \{a_1, \ldots, a_L\}$. Define the operator $\gamma_L^G(X)$ as

$$\gamma_L^G(X) = \bigcup_{a \in \Pi_L^G(X)} \sigma(a),$$

that is, $\gamma_L^G(X)$ is the union of the elements of all paths of length L in X, according to the adjacency graph G. It is easy to check that γ_L^G is an opening.

C. Threshold Decomposition

To get a grayscale path opening operator, one may use thresholding to decompose a grayscale image into a series of binary images, apply the binary path opening operator on each of these and finally combine the resulting openings into a single image, which corresponds to the grayscale opening. Such an operation may at first glance seem very costly. Recent algorithms [1], [26], [29] have managed to speed up this process, although the cost is still considerable.

A recent paper by Morard *et al.* [15] proposes an approximate algorithm that substantially improves the speed at the cost of introducing bias. In the following section we describe a new approach.

III. Upper Skeleton Path Opening

The algorithm proposed in this paper, the upper skeleton path opening (USPO), takes an input image and returns an approximate path opened image.

The algorithm consists of six steps:

1) Apply the H-minima transform to the input image [22].
2) Generate the upper skeleton of the transformed image [19], [30].

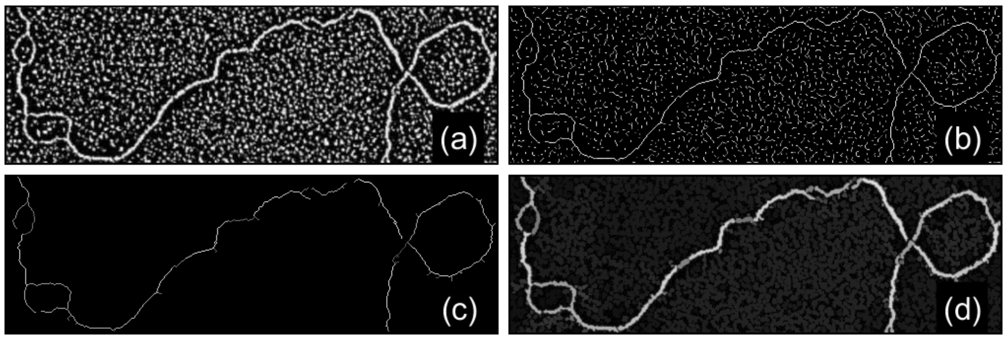

Fig. 3. An example application of the proposed opening. (a) is an 8-bit grayscale image of a DNA molecule on a textured background. (b) Is the result of upper skeletonization after an H-minima transform using $h = 80$. (c) is the opened image using $L = 25$. Finally, (d) is the result of reconstructing the image using (c) as marker and (a) as mask.

3) Generate a graph with edges weighted by length from the skeleton (Section III-A).
4) Perform a path opening on the graph (Sections III-B and III-C).
5) Generate an image from the opened graph (Section III-D).
6) Perform a reconstruction by dilation [31], yielding an approximation of the traditional path opening.

The H-minima transform reduces the number of branches in the skeleton by removing local minima shallower than some h. The h value for the H-minima transform should be as large as possible, while retaining all essential information. We have not found a way of automatically choosing it. By skeletonizing the image, the resulting graph (see Section III-A) becomes much smaller (i.e. fewer nodes and edges), which results in a faster algorithm. The ratio of input pixels to graph nodes depends on the image, but the number of nodes can often be an order of magnitude less than the number of pixels in the original image.

The upper skeletonization is related to the watershed algorithm. In fact, simply pruning the skeleton will result in a watershed. The upper skeleton may be found by thresholding the input image at each level, from lowest to highest intensity, and using binary operations at each level to remove pixels. In short, the upper skeleton is created by a conditional erosion that preserves the topology of the original image. The skeletonization can be implemented using a priority queue algorithm, and runs in $\mathcal{O}(N \log N)$ time for floating-point images. For integer-valued images, this can be improved to $\mathcal{O}(N)$.

Although pixels are removed from the image, the most relevant information is preserved in the skeleton, since the divides produced by the upper skeleton always run over the surface of the image.

Figure 3 shows an example application of this algorithm to a noisy image of a DNA molecule.

A. Skeleton to Graph

The algorithm relies on representing the image as a directed, acyclic graph. To generate the graph, the algorithm creates a node at every non-zero value of the image. After all the nodes are created, each node is connected to any adjacent node (specified by the adjacency graphs in Fig. 2). Note that these adjacency graphs yield directed, acyclic graphs. Each connection is marked according to which adjacency graph is used. Edges connecting diagonal neighbors are weighted by 1.340 and edges connecting horizontal/vertical neighbors are weighted by 0.948. These weights minimize the relative error (i.e. the difference between the measured length and the actual length divided by the actual length) for digital lines of arbitrary orientation [17]. Since paths are line-like, they will also minimize the error of paths reasonably well, assuming that the paths do not zig-zag inside thicker, line-like structures. Since the skeleton produces one-pixel thin preselected paths, the assumption holds. There are, however, other weights that could be used, depending on the error to be minimized (see for example the paper by Dorst and Smeulders [7]). In addition to connections, each node also stores its status ($active$), the grayvalue of the pixel that spawned it ($value$), as well as its corresponding position (pos). The up- and downstream lengths will be initialized in an important step of the algorithm, detailed in Section III-B, and $active$ will be initially TRUE for all nodes. The pseudocode presented in Alg. 1 shows how the graph is generated.

B. Initialize Lengths

After the graph has been generated, the initial length values need to be set. Let us consider the adjacency graph corresponding to paths in the S-N direction, since the procedure is essentially the same for each adjacency graph. The algorithm goes through the array of nodes supplied by the function CREATEGRAPH. Whenever a node that has no neighbors towards the north is discovered, start from that node and propagate to its southward neighbors. At each node, look at the northward neighbors and their northward lengths together with the weight of their corresponding edge. If the sum of the northward length and the edge weight is *greater* than the nodes current northward length, then update this length and propagate southward. For the initial node with no northward neighbors, set its northward length to the largest weight a northward edge *would* have, if it had all its northern neighbors, i.e. 1.340 (this simplifies implementation slightly, and gives good results). Do the same thing for any node that has no southward neighbors (except switching south with north in the above description). This, then, completes the initialization in the S-N direction. The initialization process is the same for

Algorithm 1 This Function is Used to Generate the Graph

```
1:  class NODE
2:      Defines $value$, $active$, $pos$
3:      method NEIGHBORS($dir$)
4:          return the neigboring nodes in the direction of $dir$, i.e. neighbors based on the corresponding adjacency graph.
5:      method GETLENGTH($upOrDown$)
6:          return the up- or downstream length. $upOrDown$ takes as value either UP or DOWN
7:      method SETLENGTH($upOrDown$, $len$)
8:          Sets the length in direction $upOrDown$ to $len$
9:      method TOTALLENGTH
10:         return GETLENGTH(UP) + GETLENGTH(DOWN)
11: function CREATEGRAPH($image$)
12:     Let $w$ be the width of $image$ and $h$ be the height.
13:     Let $nodes$ be an array of size equal to the number of non-zero pixels in $image$.
14:     for each $x \in \{0, 1, \ldots, w-1\}$ do
15:         for each $y \in \{0, 1, \ldots, h-1\}$ do
16:             if $image[x, y] > 0$ then
17:                 $n \leftarrow$ NODE.$new$
18:                 $n.active \leftarrow true$
19:                 $n.value \leftarrow image[x, y]$
20:                 $n.pos \leftarrow (x, y)$
21:                 Push $n$ onto $nodes$
22:     for each $n \in nodes$ do
23:         Check for neighboring nodes to $n$ in each direction
24:         Register neighbors together with their direction.
25:         Associate 1.340 with diagonal edges and 0.948 with straight edges.
26:     return $nodes$
```

any other adjacency graph. The pseudocode in Alg. 2 outlines an algorithm to perform this task. Now that the graph has been created and all values have been initialized, only one major hurdle remains, namely adapting the path opening algorithm presented by Luengo Hendriks [11] to a path opening on graphs.

C. Graph-Based Path Opening

To understand the path opening on graphs presented in this section, one should first consider the path opening algorithm [11] that calculates the opening directly on the image. Therefore, what immediately follows is a brief description of this approach.

First, create a list of indices to every pixel in the image, then sort this list according to the grayvalue of the pixels pointed to, from low to high. Mark each pixel as active (except border pixels). Also keep two temporary images λ^+ and λ^- that keep track of path lengths (cf. GETLENGTH/SETLENGTH in the node class). The two length images are initialized to L for each pixel, where L is the length of the path opening.

After this initialization, the pixels are updated one by one, from those with the lowest grayvalue to those with the highest, by looking forward and backward (according to the adjacency graph) and iteratively enqueuing neighbors, while updating their associated lengths. After the propagation for a given pixel, check all changed pixels to see if their total length (i.e. $\lambda^+ + \lambda^- - 1$) is less than L. If so they are not part of a path of length L at this graylevel and so they are set to the value of the threshold at which they *were* part of such a path, and marked as inactive. After each pixel has been processed, the algorithm is finished.

Now let us look at how a similar operation on a graph that has been initialized by INITIALIZELENGTHS (see Alg. 2) may be performed.

The algorithm looks through the nodes, checking TOTALLENGTH, and marking as inactive those nodes whose associated length is shorter than the path length. This means that, often, many nodes are inactive at the start of the algorithm (after the initialization). Then, just as before, the nodes are sorted according to their associated grayvalue, from lowest to highest.

Now, for each node, from lowest grayvalue to highest, propagate first forward (according to the adjacency graph), then backward simply following edges associated with the current adjacency graph. However, instead of counting pixels (as is done in the traditional algorithm) calculate lengths based on the edge weights that have been set during the graph creation outlined in Alg. 1.

After the forward and backward propagation of each node, check the associated length for each changed node (i.e. those nodes whose lengths have been updated). If too short, then just

Algorithm 2 The Procedure INITIALIZELENGTHS is Used to Set the Initial Lengths of the Nodes in the Graph

```
1: procedure INITIALIZEFROMNODE($node$, $forward$, $backward$, $streamDir$)
2:     Let $Q_i$, and $Q_n$ be empty queues
3:     Enqueue $node$ in $Q_i$
4:     while $Q_i$ is not empty do
5:         $q_i \leftarrow Q_i$.POP
6:         $N_b \leftarrow q_i$.NEIGHBORS$(backward)$
7:         if $N_b$ is not empty then
8:             $\lambda_{max} \leftarrow \max_{q_n \in N_b}(q_n.\text{GETLENGTH}(streamDir) + \text{EDGEWEIGHT}(q_i, q_n))$
9:         else
10:            $\lambda_{max} \leftarrow 1.340$
11:        if GETLENGTH$(streamDir) < \lambda_{max}$ then
12:            SETLENGTH$(streamDir, \lambda_{max})$
13:            Enqueue in $Q_n$ all elements of $q_i$.NEIGHBORS$(forward)$
14:        if $Q_i$ is empty then
15:            Enqueue in $Q_i$ all elements of $Q_n$
16:            Let $Q_n$ be an empty queue
17: function EDGEWEIGHT($node_1$, $node_2$)
18:     return the weight of the edge between $node_1$ and $node_2$
19: procedure INITIALIZELENGTHS($nodes$, $up$, $down$)
20:     for each $n \in nodes$ do
21:         if $n$.NEIGHBORS$(up)$ is empty then
22:             INITIALIZEFROMNODE($n$, $down$, $up$, UP)
23:         if $n$.NEIGHBORS$(down)$ is empty then
24:             INITIALIZEFROMNODE($n$, $up$, $down$, DOWN)
```

as before, mark the node as inactive and set its corresponding grayvalue to the appropriate threshold value. The algorithm is outlined in pseudocode in Alg. 3. It should be noted that the algorithm does not guarantee processing the pixels at each graylevel in topological order. For input with a large number of quantization levels, the number of pixels at the same level will generally not be sufficient to make the processing order significantly affect timing, however, for images with fewer graylevels (even 8-bit images), the algorithm might be sped up if this processing order could be guaranteed.

D. Graph to Image

After the graph has been opened, an image can be constructed by creating an image of the same dimensions as the input image, with zeros everywhere, and then, for each node *n* in the graph, setting the pixel in the black image, at the position given by $n.pos$, to the value of $n.value$.

Finally, reconstruct the image by dilation if the thickness of lines needs to be preserved.

IV. RESULTS

We implemented the upper skeleton path opening described above in C++, with an interface to MATLAB. We have used the H-minima transform from MATLAB's Image Processing Toolbox, and the upper skeleton implemented in the DIPimage toolbox. This implementation is compared to the implementation of the stack-based path opening by van de Gronde *et al.* [29] (using the adjacency relations illustrated in Fig. 2), the constrained path opening described by Luengo Hendriks [11], as well as the parsimonious path opening presented by Morard *et al.* [15]. We also compare it to a slightly modified version of the PPO, where we use the same weights [17] mentioned in Section III-A for the edges.

A. Applying the USPO

In this section the result of applying PO, CPO, PPO, and USPO is shown. Figure 4 shows the effect of applying the four different algorithms on the image in Fig. 3.

We will now examine two applications of the path opening and compare the different variants.

1) Straight Line Detection: In Figure 5(a) a photograph of a circuit board is shown. We shall see that the USPO can be used to detect the conductive traces. A tophat transformation using a disk structuring element of radius 6 is applied. Figure 5(b) shows the result of applying the PPO of length 50 with $\beta = 1$, skipping the final reconstruction step, to the transformed image. Figure 5(c) is the result of the USPO of length 50 using $h = 10$, also without reconstruction. The magnified areas show a patch of the board where the conductive traces are packed closely together. Note that this example is not meant to suggest that the traces extracted by applying the USPO in this manner are useful in any practical sense, since many conductive traces are missed. Instead, the purpose is to illustrate a weakness of the PPO.

2) Application to Fundus Images: Sigurðsson *et al.* [20] present an algorithm for detecting blood vessels in retinal fundus images. The algorithm uses the traditional path opening to filter out false positives by removing paths shorter than a

Algorithm 3 Directed Path Opening in S-N Direction on a Graph

```
1: $N \leftarrow$ CREATEGRAPH$(I)$, where $I$ is the image to be processed.
2: Run INITIALIZELENGTHS($N$, NORTH, SOUTH), to set the inital length values.
3: Sort active nodes in $N$, such that they appear in order of lowest value to highest value.
4: Call DIRECTEDPATHOPENING($N$, SOUTH, NORTH, $L$), where $L$ is the length.
5:
6: procedure DIRECTEDPATHOPENING($graph$, $down$, $up$, $length$)
7:     Let $C$ be an empty FIFO-queue
8:     for each $n \in graph$ do
9:         if $n$.TOTALLENGTH $< length$ then
10:            $n.active \leftarrow$ FALSE
11:            $n.value \leftarrow 0$
12:    for each $n \in graph$, such that $n.active$ is TRUE do
13:        PROPAGATE($n$, $down$, $up$, UP, $C$)
14:        PROPAGATE($n$, $up$, $down$, DOWN, $C$)
15:        for each $c \in C$ do
16:            if $c$.TOTALLENGTH $< length$ then
17:                $c.value \leftarrow n.value$
18:                $c.active \leftarrow$ FALSE
19:        $n.active \leftarrow$ FALSE
20: procedure PROPAGATE($node$, $forward$, $backward$, $streamDir$, $C$)
21:    Let $Q$ be an empty FIFO-queue
22:    $node$.SETLENGTH$(streamDir, 0)$
23:    $N_f \leftarrow node$.NEIGHBORS$(forward)$
24:    Enqueue in $Q$ all elements $f \in N_f$ for which $f.active$ is TRUE
25:    while $Q$ is not empty do
26:        $q \leftarrow Q$.POP
27:        $N_b \leftarrow q$.NEIGHBORS$(backward)$
28:        $\lambda_{max} \leftarrow \max_{q_n \in N_b}(q_n$.GETLENGTH$(streamDir)$ + EDGEWEIGHT$(q, q_n))$
29:        if $\lambda_{max} < q$.GETLENGTH$(streamDir)$ then
30:            $q$.SETLENGTH$(streamDir, \lambda_{max})$
31:            $N_{ff} \leftarrow q$.NEIGHBORS$(forward)$
32:            Enqueue in $Q$ all elements $f \in N_{ff}$ for which $f.active$ is TRUE.
33:            Enqueue $q$ in $C$
```

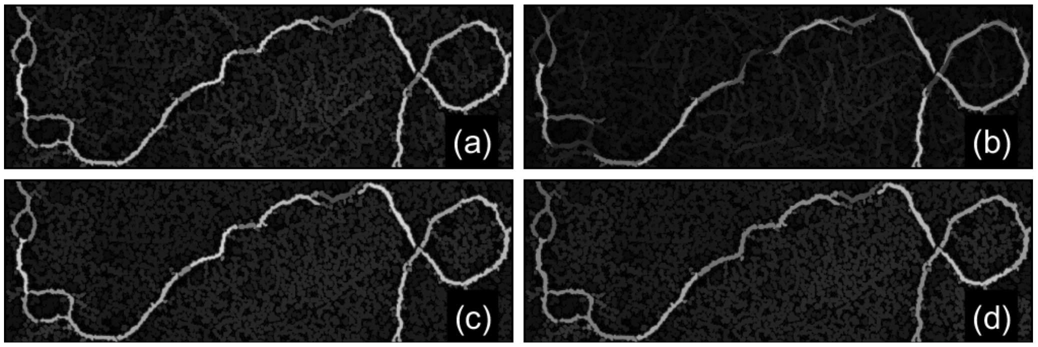

Fig. 4. Comparison between the traditional path opening (a), the constrained path opening (b), the parsimonious path opening (c), with $\beta = 1$, and the upper skeleton path opening (d), with $h = 0$. The length used is $L = 50$. The CPO (b) sticks out because of its restrictions on paths which, in this case, lead to failing to preserve some segments of the molecule.

threshold length of 34. To compare the performance of the USPO with the traditional path opening (PO), as well as the PPO and the CPO, a slightly modified version of this algorithm is implemented using each of these four types of path opening. The algorithm is tested on the DRIVE [24] and the STARE [10] datasets. The performance is evaluated using the method proposed by Gegúndez-Arias *et al.* [8] (with $\alpha = \beta = 2$), which looks at the connectivity (C), area (A), and length (L) of the segmentation compared with the ground truth. As in the paper [20], the parameter K_1 is fixed as 1.1. The parameter K_2 is chosen by finding the optimal value (with respect to the C∗A∗L-score) for a number of training images

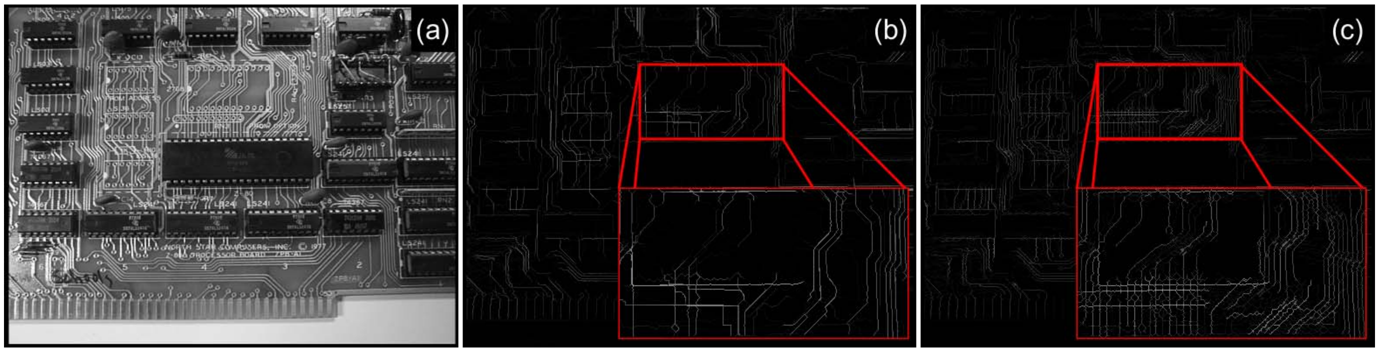

Fig. 5. Comparison between PPO and USPO on a photograph of a circuit board. A tophat transform is applied to the image in (a). (b) Shows the result of applying the PPO on the transformed image. (c) Shows the result of applying the USPO on the transformed image. The red box shows a magnified view of the indicated part of the image. Notice that the PPO misses several of the tightly packed lines while the USPO does not have this problem. The reconstruction step is skipped in both (b) and (c).

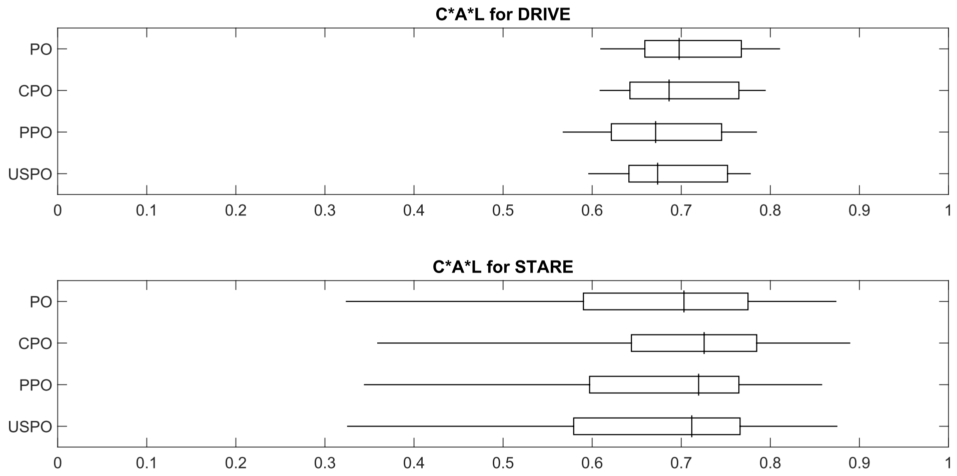

Fig. 6. Box plots showing the results when measuring the performance of the vessel segmentation method using different path openings on the DRIVE dataset, and the STARE dataset. The whiskers represent the C*A*L-scores for the best and worst test results, the box indicates the first and third quartile over the test images, and the vertical line shows the median test result.

and taking the mean of these values of K_2 as the parameter value used when testing. These parameters adjust how likely it is that a pixel is classified as a vessel/non-vessel.

Aside from the different path openings used, the implementation differs from that of Sigurðsson *et al.* [20] on two additional points: The green channel is examined instead of the luminance of the YUV color space, and a mask that delineates the FOV is automatically found by thresholding the image.

For the DRIVE dataset the last 20 images are used for training and the other 20 images are used for testing. The box plot in Fig. 6 shows the max/min values of C$*$A$*$L, as well as the 25th and 75th percentiles for the test images. The traditional path opening performs best, however, all the path opening methods tested give quite similar results.

The STARE dataset contains a lot fewer labeled images (20 in total), therefore a 10-fold cross-validation is performed. The training used the segmentation by the first observer (label-ah) as ground truth. Figure 6 shows the results when comparing against the first observer.

Figure 7 shows an image from the STARE dataset together with its manual segmentation and the automatic segmentation when using USPO, PPO, PO, and CPO. Visually the results are very similar. On this image, the constrained path opening gets the highest C$*$A$*$L-score and the parsimonious path opening gets the lowest score. The differences are slight, however a few problematic areas are highlighted in the different segmentation results.

B. Size Distribution

In order to measure the accuracy of the upper skeleton path opening, eight images were created. These images contain 50 line segments of length 40 with a Gaussian profile, where $\sigma = 1$ pixel, and a random subpixel shift. All lines within an image have the same slope. The angles chosen range from 0° to 45° at equal intervals. Figure 8(a) shows three examples of such images. An experiment that measures the length granulometries for these images using openings of length 30 to 66 increasing by steps of 1 was performed for the traditional path opening, the constrained path opening, the parsimonious path opening, the parsimonious path opening with tweaked weights (we will refer to this variation, where horizontal/vertical steps are weighted as 0.948 and diagonal steps as 1.340, as the *unbiased PPO*), and the upper skeleton path opening.

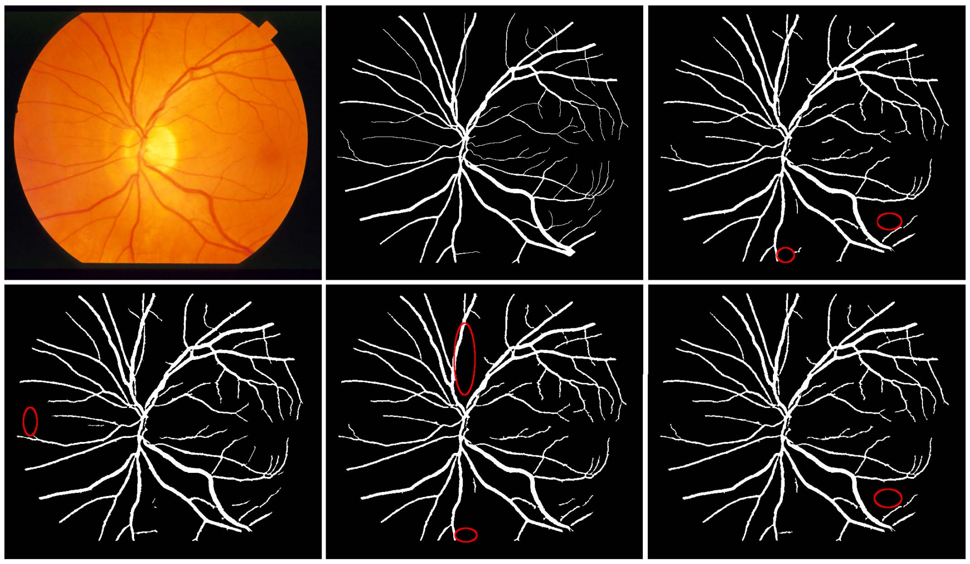

Fig. 7. Top row, left to right: Original image from the STARE dataset. Ground truth. Vessels detected using PO during filtering step. Bottom row, left to right: Vessels dected using CPO, PPO, and USPO. For this image, the CPO has the highest C∗A∗L-score (0.8891) out of all the opening variants, followed by the USPO (0.8747), the PO (0.8736), and finally the PPO (0.8576). Some differences are circled in red.

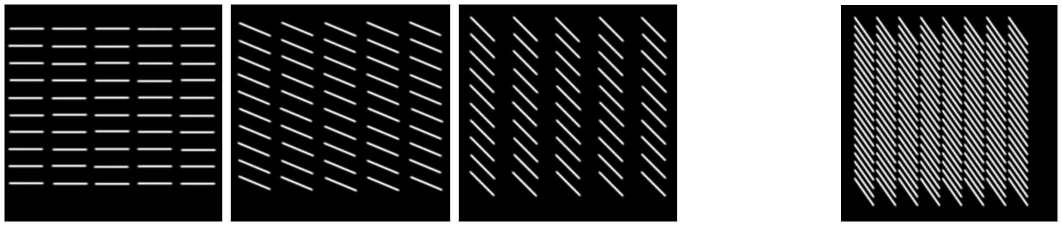

Fig. 8. Examples of test images used. The first set of test images contain 50 line segments of length 40 at angles between 0° and 45° with a Gaussian profile ($\sigma = 1$) and a random subpixel shift. (a) Shows three examples of such test images. (b) Shows the test image used to show the problem of occlusion. It contains tightly packed lines of length 40 at an angle of 55° with a Gaussian profile ($\sigma = 1$).

Additionally, the supremum of hundreds of line openings of length 30 to 66 at angles between 0° and 45° was also used as comparison. Figure 9 shows the results. Ideally, the graph should be 0 for all lengths shorter than 40 and 1 otherwise. However, because of the nature of the test images, a perfect result is not to be expected.

To quantify the accuracy of a method, we compute three measures on the set of eight curves $\{f_\theta\}$:

1) $\mu = \frac{1}{8}\sum_{\theta=1}^{8}\int (x-40)^2 g_\theta(x)\, dx$, where

$$g_\theta(x) = \begin{cases} f_\theta(x), & \text{if } x < 40 \\ 1 - f_\theta(x), & \text{otherwise} \end{cases}$$

2) $b = \frac{1}{8}\sum_{\theta=1}^{8} x_\theta^t - 40$, where x_θ^t is the first x s.t. $f_\theta(x_\theta^t) > 0.5$

3) σ_b is the standard deviation of the b_θ, where $b_\theta = x_\theta^t - 40$.

The first measure is the second order moment around 40 for g_θ, which, in the ideal case is 0. This is a measure of how closely, on average, the curves follow the ideal case. Errors are weighted depending on the distance from the correct length. The integral is estimated using the trapezoid rule. The second measure shows the mean bias of the method by looking at the average distance between 40 and the point at which the curve crosses 0.5. Finally, the third measure quantifies dependency on orientation. A high standard deviation of the b_θ, means that the eight curves are spread apart, while a low deviation means that the spread is small (at least near $x = 40$).

It is also of interest that, as discussed in Section I and illustrated in Fig. 5, the PPO has a problem with occlusion. To see how this affects the result, an image containing tightly packed lines of length 40 at 55° (see Fig. 8(b) was opened at different length scales using the unbiased PPO and the USPO. The results are shown in Fig. 10.

C. Timings

We ran some experiments to measure the speed of the new path opening. These times are compared to the constrained path opening algorithm presented by Luengo Hendriks [11], the stack-based path opening implementation by van de Gronde *et al.* [29] (for one and eight threads, when applied to the original image or the preprocessed image used by the USPO, abbreviated as "Stack" and "US + Stack" respectively), as well as the PPO [15]. Figure 11(a) shows timings for an image of 2300×2300 pixels and path length varying between 5 and 640. The USPO uses $h = 40$ and the PPO uses $\beta = 1$. The parameter β affects the preselected paths. Larger values result in an opening that is more robust to

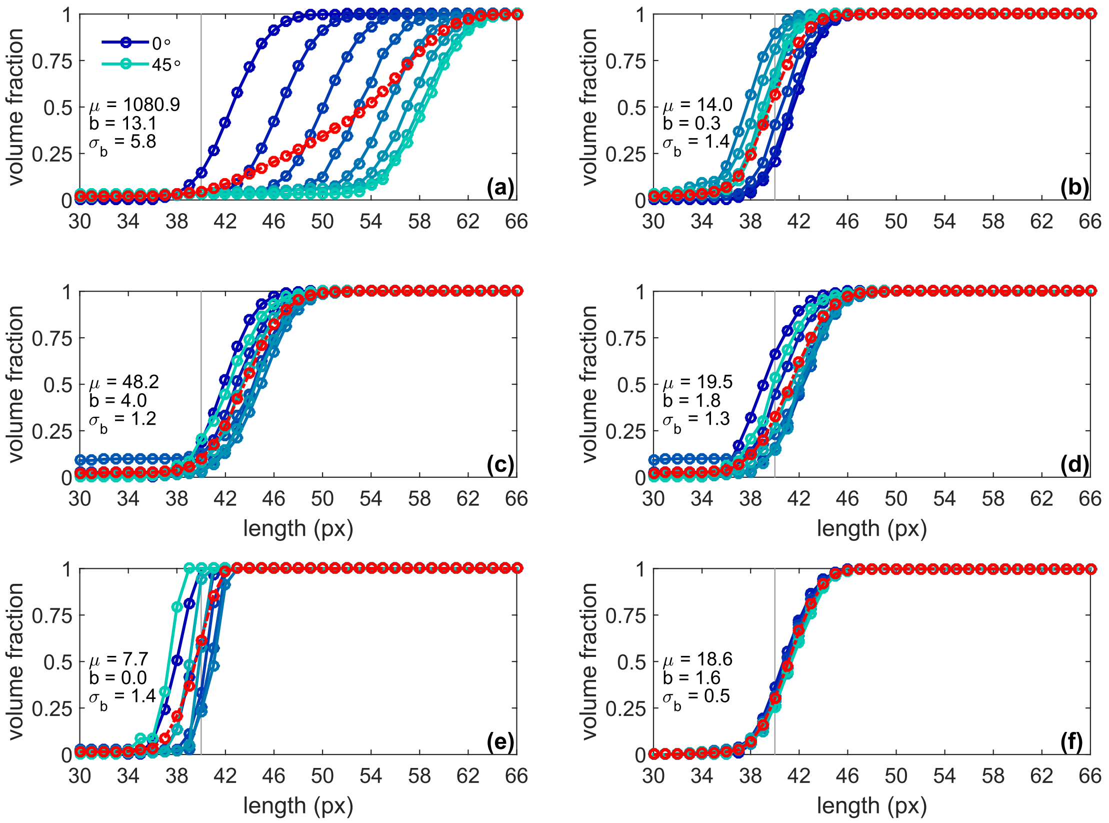

Fig. 9. The traditional path opening (a) overestimates the length of lines a lot when the path is able to zig-zag in the thick lines. The constrained path opening (b) addresses this problem. (c) PPO with $\beta = 1$, and weights 1 and $\sqrt{2}$, as presented by Morard *et al.* [15]. (d) Unbiased PPO with $\beta = 1$. The USPO (e) with $h = 0$ seems to give the best results apart from (f), the very expensive supremum over openings with straight line segments at many angles (hundreds), with a low bias, b, a low spread, σ_b, and a low second order moment, μ, around 40. In addition to being expensive, the supremum of line openings is also not robust (i.e. they fail to preserve paths). The red line shows the mean curve. Fig. 8(a) shows some examples of the images used for testing.

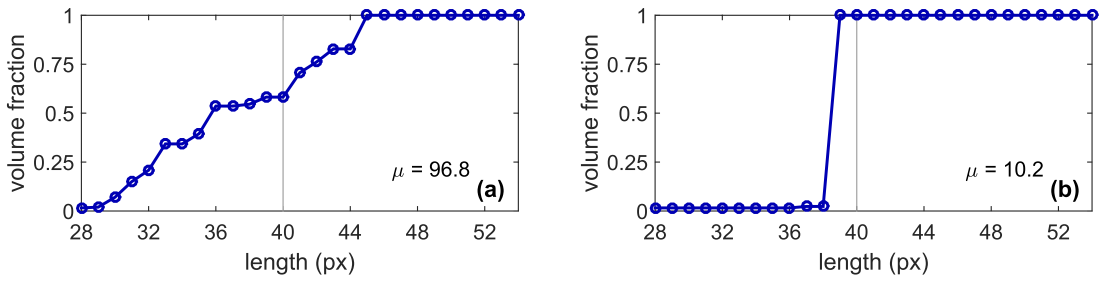

Fig. 10. Measuring length distributions of an image with many tightly packed lines of length 40 (see Fig. 8(b). The parsimonious path opening (a) has a problem of occlusion—bright structures can occlude parts of the image—and shows a clear bias towards shorter paths. The USPO (b) is not affected by the close spacing of the lines.

noise, but one that also leaves greater blind regions and takes more time to compute.

Figure 11(b) shows timings for varying image sizes (of similar images), with paths of length 40 pixels, and $h = 40$ for the USPO and $\beta = 1$ for the PPO.

Finally, Fig. 12 shows timings for the upper skeleton path opening algorithm when applying the H-minima transform with values of h ranging from 5 to 160 on an image of size 1040×780 pixels and a path length of 10 pixels.

V. Discussion and Conclusions

In this paper a new algorithm, the upper skeleton path opening, that approximates the path opening has been presented.

The algorithm creates a sparse graph, by applying an H-minima transform followed by an upper skeletonization, upon which a graph-based path-opening algorithm is run. The opened graph can then be used to reconstruct an approximate path opening (Fig. 4 shows an example). The H-minima transform produces a much neater skeleton by removing local minima shallower than some h, thus improving computation time (see Fig. 12). The h-value should be chosen as large as possible, while still retaining all structures of interest in the transformed image. The H-minima transform and skeletonization will generally prune the graph of the image down greatly, which is the source of the improvement in speed. The algorithm may be run on images without the preprocessing step, however, this will usually be slower than other path

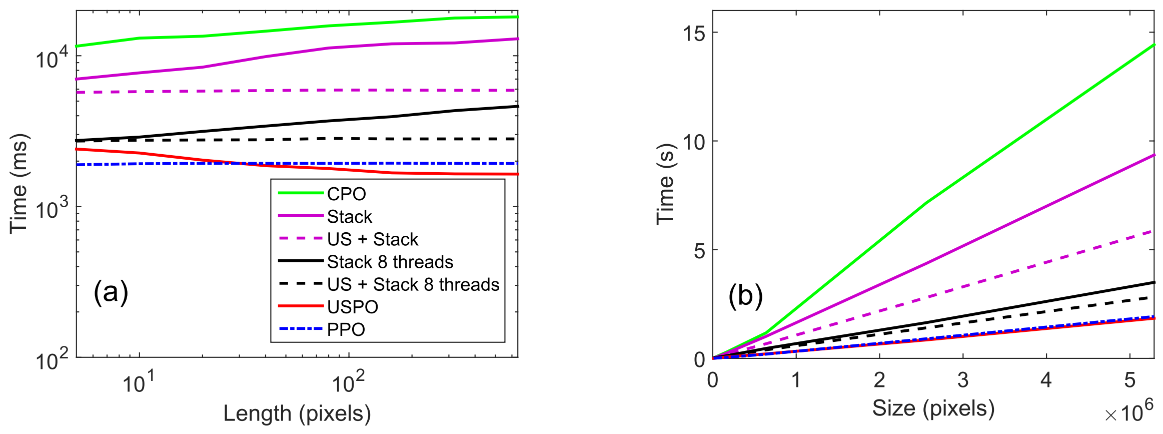

Fig. 11. (a) Time vs. path length for an 8-bit image of 2300×2300 pixels, with $h = 40$, and $\beta = 1$. (b) Time vs. image size, for similar 8-bit images, with paths of length 40 pixels and $h = 40$, $\beta = 1$. The constrained path opening is shown in green, the traditional path opening in purple, the upper skeleton path opening in red, and the parsimonious opening in dashed blue. The timings do not include the optional reconstruction step for PPO, USPO, or US + Stack.

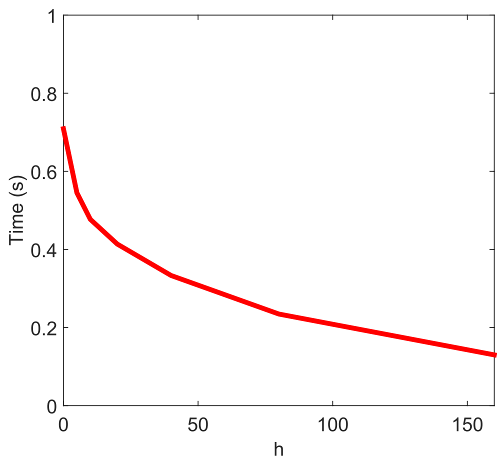

Fig. 12. Timing of the upper skeleton path opening, without reconstruction, for different values of h in the H-minima transform on an 8-bit image of size 1040×780 pixels using a path length of 10 pixels.

opening algorithms (such as the stack-based path opening), since the proposed algorithm explicitly generates a graph and therefore has to deal with extra overhead.

The algorithm is similar, in some ways, to the parsimonious path opening presented by Morard *et al.* [15]. The skeleton is essentially used to preselect paths analogous to the paths selected by the parsimonious algorithm.

In Section IV-A comparisons between different path opening variants are made by applying them to two images: A DNA molecule on a textured background (Fig. 4), and a circuitboard (Fig. 5). The performance of the various openings is also compared when used as a filtering step in an algorithm [20] that detects blood vessels in eye fundus images. The proposed opening is shown to perform on par with the parsimonious path opening when detecting blood vessels. The USPO is also shown to preserve tightly packed long thin structures which are sometimes missed by the PPO (see the conductive traces of the circuitboard in Fig. 5 and the measured length distribution in Fig. 10).

In Section IV-B, additional experiments on synthetic images show that the algorithm produces accurate results when measuring length distributions. The tests indicate that the USPO compares favorably to all path opening variants tested, with the lowest mean bias, b, a low orientation dependency, σ_b, and the smallest μ, indicating that the average curve is close to the ideal step function. However, it should be noted that for large values of h, the bias increases. We also suggest a small modification to the PPO and show that this modification reduces bias (Fig. 9(d)). The algorithm presented in this paper also compares favorably to this unbiased PPO.

Section IV-C measures the speed of the algorithm. The experiments suggest that, for large images, this algorithm is close to an order of magnitude faster than the constrained path opening [11] as well as the stack-based algorithm proposed by van de Gronde *et al.* [29] running on one thread. Additionally the USPO can be *faster* with increasing path length, and may behave linearly in time with respect to image size. The experiments also indicate that the USPO running on a single thread is considerably faster than the stack-based algorithm using eight threads. Performing the skeletonization preprocessing steps (i.e. H-minima transform followed by skeletonization) improves the speed of the stack-based algorithm, however, the USPO compares favorably with regard to time in this case as well. Moreover, the USPO algorithm is comparable in speed to the PPO (depending on chosen parameter values and the image), although the USPO implementation could be further optimized e.g. by combining the skeletonization and graph building step into one. Worth noting is the fact that the skeleton is not a one-dimensional structure, as the preselected paths in the PPO, which precludes us from applying the efficient 1D opening algorithm [15]. Also note that, for pathological cases, where the proportion of pixels *not* in the skeleton of the image is small, the USPO may not be faster than the traditional path opening. An example of such a case is an image filled with uniform noise.

One might consider other ways of preprocessing the input to reduce the number of nodes and edges in the graph. For example, simply applying a top hat can filter out some structures that are not line-like, thus resulting in a smaller graph [5], [11]. One might also use some other preprocessing to enhance curvilinear structures (see for example the paper by Schubert *et al.* [18]). However, neither approach solves the problem of zig-zagging in thicker lines. Moreover, reconstructing the image in such cases does not generally approximate the traditional path opening of the original image well. Thus, to generate a sparse graph one might use some other approach than the upper skeleton, but in order to work as a replacement for the upper skeleton, one should take care that the preprocessed image does not allow for zig-zagging, i.e. thick lines should be thinned. The method should also be fast, and the opened sparse graph should be able to be reconstructed as an approximation of the traditional path opening.

The algorithm as described is only applicable to 2D images. However, an extension to 3D should not prove problematic. The opening on the graph as described in Section III is dimensionality independent except, in the 3D case for example, where instead of the four adjacency graphs in Fig. 2, there are 13 possible orientations. Moreover, the optimal edge weights between neighboring nodes are different for different dimensionalities. However, the main issue is the selection of paths (i.e. the skeletonization). There are a number of 3D-skeletonization algorithms [6], [21] that, by combination with the ideas of Verwer *et al.* [30], we could adapt for this purpose.

Another limitation comes from the susceptibility to noise of the path opening. In order to get a more robust opening, we propose to use the method described in a recent paper by Merveille *et al.* [12]. The idea is to dilate the original image using a square SE of a size determined by the length of the largest gap caused by noise. The path opening (e.g. USPO) is then performed on this dilated image. Finally, the infimum of the original image and the opened, dilated image is the result of the robust opening. There are other approaches to creating robust path openings [5], [26], however, since the skeleton itself gets disturbed by noise, it is not clear that modifying the path opening of the graph is viable.

Lastly, we noticed that the USPO sometimes has problems finding long paths. If the paths selected by the skeleton are too tortuous (such that the lengths fail to propagate forward according to the adjacency relations used), the paths found will be shorter than expected. Since the PPO selects paths according to the adjacency relations used, it does not suffer from this problem. Therefore, depending on the image, the PPO may find longer paths than the USPO. This also explains part of the speed-up for the USPO for larger values of L seen on some images. This may perhaps be alleviated by modifying the adjacency graphs used, in order to allow for more extreme turns, e.g. by increasing the cone of adjacent vertices to 135°, i.e. by adding a fourth neighbor to the adjacency graphs in Fig. 2 in a similar manner to the adjacency graphs used by Cao *et al.* [4]. It may also be possible to change the adjacency relation used while propagating lengths (e.g. based on the direction of the previous step) as long as care is taken to not introduce cycles. Another possibility might be to make use of the approach used by van de Gronde *et al.* [28], who propose to create a directed graph by computing an orientation score on a 2D image, which is then used to connect nodes based on the angle between tangent vectors. This approach may lead to cyclical graphs, which in the paper is dealt with by contracting all cycles. Deriving the graph in this way is, however, quite expensive.

Acknowledgment

The authors would like to thank Vincent Morard for providing the base code used for the PPO implementation in the comparisons. They would also like to thank Jasper J. van de Gronde, Herman R. Schubert, and Jos B. T. M. Roerdink for making the code for the fast, stack-based path opening algorithm available. Finally, they would like to thank the reviewers for their constructive feedback.

Teo Asplund was funded through grant 2014-5983 from the Swedish Research Council.

References

[1] B. Appleton and H. Talbot, “Efficient path openings and closings,” in *Proc. Int. Symp. Math. Morphol.*, vol. 30. 2005, pp. 33–42.

[2] E. J. Breen and R. Jones, “Attribute openings, thinnings, and granulometries,” *Comput. Vis. Image Understand.*, vol. 64, no. 3, pp. 377–389, 1996.

[3] M. Buckley and H. Talbot, “Flexible linear openings and closings,” in *Proc. Int. Symp. Math. Morphology*, vol. 18, 2000, pp. 109–118.

[4] G. Cao, S. Wang, and Y. Liu, “An improved algorithm for automatic road detection in high-resolution remote sensing images by means of geometric features and path opening,” in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2015, pp. 1861–1864.

[5] F. Cokelaer, H. Talbot, and J. Chanussot, “Efficient robust d-dimensional path operators,” *IEEE J. Sel. Topics Signal Process.*, vol. 6, no. 7, pp. 830–839, Nov. 2012.

[6] M. Couprie and G. Bertrand, “A 3D sequential thinning scheme based on critical kernels,” in *Proc. Int. Symp. Math. Morphology*, vol. 9082. 2015, pp. 549–560.

[7] L. Dorst and A. W. Smeulders, “Best linear unbiased estimators for properties of straight lines,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 2, pp. 276–282, Feb. 1986.

[8] M. E. Gegundez-Arias, A. Aquino, J. M. Bravo, and D. Marin, “A function for quality evaluation of retinal vessel segmentations,” *IEEE Trans. Med. Imag.*, vol. 31, no. 2, pp. 231–239, Feb. 2012.

[9] H. Heijmans, M. Buckley, and H. Talbot, “Path-based morphological openings,” in *Proc. Int. Conf. Image Process. (ICIP)*, vol. 5. Oct. 2004, pp. 3085–3088.

[10] A. Hoover, V. Kouznetsova, and M. Goldbaum, “Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response,” *IEEE Trans. Med. Imag.*, vol. 19, no. 3, pp. 203–210, Mar. 2000.

[11] C. L. Luengo Hendriks, “Constrained and dimensionality-independent path openings,” *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1587–1595, Jun. 2010.

[12] O. Merveille, H. Talbot, L. Najman, and N. Passat, “Ranking orientation responses of path operators: Motivations, choices and algorithmics,” in *Proc. Int. Symp. Math. Morphol.*, vol. 9082. 2015, pp. 633–644.

[13] A. Miettinen, C. L. Luengo Hendriks, G. Chinga-Carrasco, E. K. Gamstedt, and M. Kataja, “A non-destructive x-ray microtomography approach for measuring fibre length in short-fibre composites,” *Compos. Sci. Technol.*, vol. 72, no. 15, pp. 1901–1908, Oct. 2012.

[14] V. Morard, E. Decenciere, and P. Dokladal, “Geodesic attributes thinnings and thickenings,” in *Proc. Int. Symp. Math. Morphol.*, vol. 6671. 2011, pp. 200–211.

[15] V. Morard, P. Dokladal, and E. Decenciere, “One-dimensional openings, granulometries and component trees in $O(1)$ per pixel,” *IEEE J. Sel. Topics Signal Process.*, vol. 6, no. 7, pp. 840–848, Nov. 2012.

[16] V. Morard and P. Dokladal, and E. Decenciere, “Parsimonious path openings and closings,” *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1543–1555, Apr. 2014.

[17] D. Proffitt and D. Rosen, “Metrication errors and coding efficiency of chain-encoding schemes for the representation of lines and edges,” *Comput. Graph. Image Process.*, vol. 10, no. 4, pp. 318–332, 1979.

[18] H. Schubert, J. J. van de Gronde, and J. B. T. M. Roerdink, “Efficient computation of greyscale path openings,” *Math. Morphol.-Theory Appl.*, vol. 1, no. 1, pp. 189–202, 2015.
[19] J. Serra, *Image Analysis and Mathematical Morphology*. New York, NY, USA: Academic, 1983.
[20] E. M. Sigurõsson, S. Valero, J. A. Benediktsson, J. Chanussot, H. Talbot, and E. Stefánsson, “Automatic retinal vessel extraction based on directional mathematical morphology and fuzzy classification,” *Pattern Recognit. Lett.*, vol. 47, pp. 164–171, Oct. 2014.
[21] A. Sobiecki, A. Jalba, and A. Telea, “Comparison of curve and surface skeletonization methods for voxel shapes,” *Pattern Recognit. Lett.*, vol. 47, pp. 147–156, Oct. 2014.
[22] P. Soille, *Morphological Image Analysis: Principles And Applications*, New York, NY, USA: Springer-Verlag, 1999.
[23] P. Soille, E. J. Breen, and R. Jones, “Recursive implementation of erosions and dilations along discrete lines at arbitrary angles,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 5, pp. 562–567, May 1996.
[24] J. Staal, M. Abramoff, M. Niemeijer, M. A. Viergever, and B. van Ginneken, “Ridge-based vessel segmentation in color images of the retina,” *IEEE Trans. Med. Imaging*, vol. 23, no. 4, pp. 501–509, Apr. 2004.
[25] T. Statella, P. Pina, and E. A. da Silva, “Image processing algorithm for the identification of Martian dust devil tracks in MOC and HiRISE images,” *Planetary Space Sci.*, vol. 70, no. 1, pp. 46–58, 2012.
[26] H. Talbot and B. Appleton, “Efficient complete and incomplete path openings and closings,” *Image Vis. Comput.*, vol. 25, no. 4, pp. 416–425, 2007.
[27] S. Valero, J. Chanussot, J. A. Benediktsson, H. Talbot, and B. Waske, “Advanced directional mathematical morphology for the detection of the road network in very high resolution remote sensing images,” *Pattern Recognit. Lett.*, vol. 31, no. 10, pp. 1120–1127, 2010.
[28] J. J. van de Gronde, M. Lysenko, and J. B. Roerdink, “Path-based mathematical morphology on tensor fields,” in *Proc. Vis. Higher Order Descriptors Multi-Valued Data*, 2015, pp. 109–127.
[29] J. J. van de Gronde, H. R. Schubert, and J. B. T. M. Roerdink, “Fast computation of greyscale path openings,” in *Proc. Int. Symp. Math. Morphology*, vol. 9082. 2015, pp. 621–632.
[30] B. J. Verwer, L. J. van Vliet, and P. W. Verbeek, “Binary and grey-value skeletons: Metrics and algorithms,” *Int. J. Pattern Recognit. Artif. Intell.*, vol. 7, no. 5, pp. 1287–1308, 1993.
[31] L. Vincent, “Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms,” *IEEE Trans. Image Process.*, vol. 2, no. 2, pp. 176–201, Apr. 1993.

Teo Asplund received the M.Sc. degree in computer science from Uppsala University, Sweden, in 2015, where he is currently pursuing the Ph.D. degree with the Centre for Image Analysis. His research interests include image processing and mathematical morphology.

Cris L. Luengo Hendriks (S’01–M’04–SM’10) received the M.Sc. and Ph.D. degrees from the Department of Applied Physics, Delft University of Technology, Delft, The Netherlands, in 1998 and 2004, respectively.

He was a Research Fellow with the Lawrence Berkeley National Laboratory, CA, from 2004 to 2008, where he collaborated with a large group of people to build the first 3-D, quantitative atlas of gene expression in the fruit fly embryo. He was an Associate Professor with the Centre for Image Analysis, Uppsala University, Sweden, from 2008 to 2015. He is currently the Director of Image Analysis with Flagship Biosciences, a company that specializes in delivering precise measurements of quantitative endpoints in digital pathology. He is the founding author of DIPimage, a MATLAB toolbox for scientific image analysis, and contributing author of DIPlib, the library on which DIPimage is built.