Published in Image Processing On Line on 2017–10–01.
Submitted on 2017–03–14, accepted on 2017–07–17.
ISSN 2105–1232 © 2017 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2017.207

2D Filtering of Curvilinear Structures by Ranking the Orientation Responses of Path Operators (RORPO)

Odyssée Merveille[1], Benoît Naegel[1], Hugues Talbot[2], Laurent Najman[2], Nicolas Passat [3]

[1] Université de Strasbourg, ICube, CNRS (UMR 7357), France ({merveille,b.naegel}@unistra.fr)
[2] Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM, Noisy-le-Grand, France ({hugues.talbot,laurent.najman}@esiee.fr)
[3] Université de Reims Champagne-Ardenne, CReSTIC, France (nicolas.passat@univ-reims.fr)

Communicated by Rafael Grompone von Gioi *Demo edited by* Thibaud Ehret

Abstract

We present a filtering method for 2D curvilinear structures, called RORPO (Ranking the Orientation Responses of Path Operators). RORPO is based on path operators, a recently developed family of mathematical morphology filters. Compared with state of the art methods, RORPO is non–local and well adapted to the intrinsic anisotropy of curvilinear structures. Since RORPO does not depend on a linear scale-space framework, it tends to preserve object contours without a blurring effect. Due to these properties, RORPO is a useful low-level filter and can also serve as a curvilinear prior in segmentation frameworks. In this article, after introducing RORPO, we develop the 2D version of the algorithm and present a few applications.

Source Code

The C++ implementation of RORPO 2D along with an online demo are available on the IPOL webpage of this article[1]

Keywords: filtering; curvilinear structure; mathematical morphology; path operators

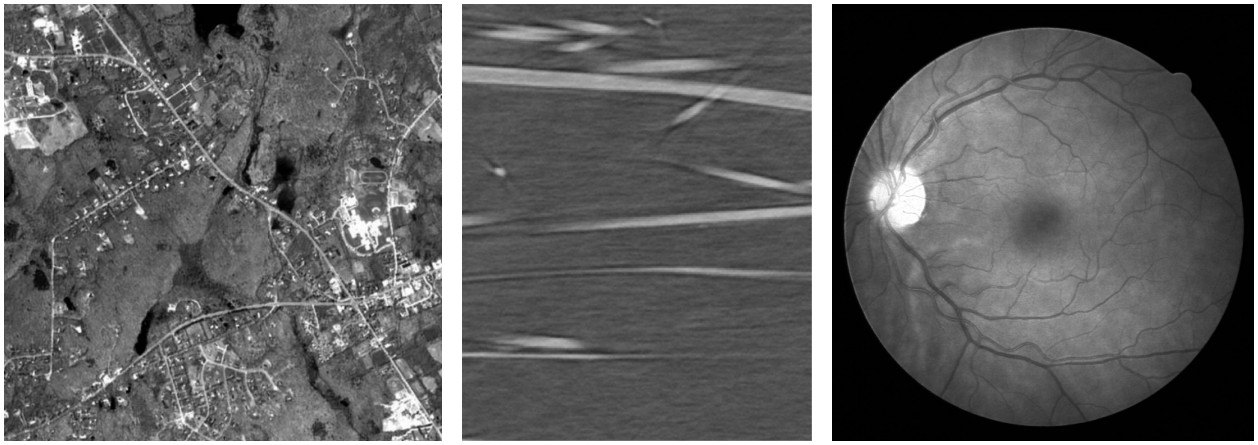
1 Introduction

Many image analysis applications involve the detection of curvilinear structures, for example roads in remote sensing, glass fibers in material sciences or blood vessels in medical imaging (see Figure 1).

Despite a large field of applications, the detection of curvilinear structures remains a difficult task, due to the sparsity of the image, their specific geometry and sensitivity to noise.

The majority of state of the art methods for detecting curvilinear structures are based on the image derivatives [6, 3, 7, 1, 13, 8, 4]. These methods generally compute the derivatives of the image

[1] https://doi.org/10.5201/ipol.2017.207



(a) (b) (c)

Figure 1: Examples of images involving curvilinear structures. (a) Roads in remote sensing, (b) glass fibers in CT-scan and (c) blood vessels in retinal images.

in a small neighborhood around a putative pixel belonging to a curvilinear structure. This local analysis is not well adapted to the anisotropy of such structures, and often results in false detections. By using path operators, our method called RORPO tends to avoid these false detections, and better detects curvilinear structures.

RORPO computes two features characterizing curvilinear structures: an *intensity feature* that can be seen as a curvilinearity measure, and a *directional feature*, providing local orientation. RORPO was originally proposed for 3D images[2] (see [11, 10]). In this article we propose a 2D implementation of this operator. We briefly present the methodology behind RORPO 2D in Section 2. Then, Section 3 develops the RORPO 2D algorithm. Section 4 explains how to use the online demo and the C++ code we provide. Finally, we present a few examples on synthetic and real images in Section 5.

2 Proposed Method

RORPO is based on path operators, which include the dual path opening and path closing [5]. In the following, as we consider bright objects on a dark background, we present our methodology with the path opening. For dark structures on a bright background, path closings should be used. In this section, we introduce the path opening, then we present RORPO and its two aforementionned features.

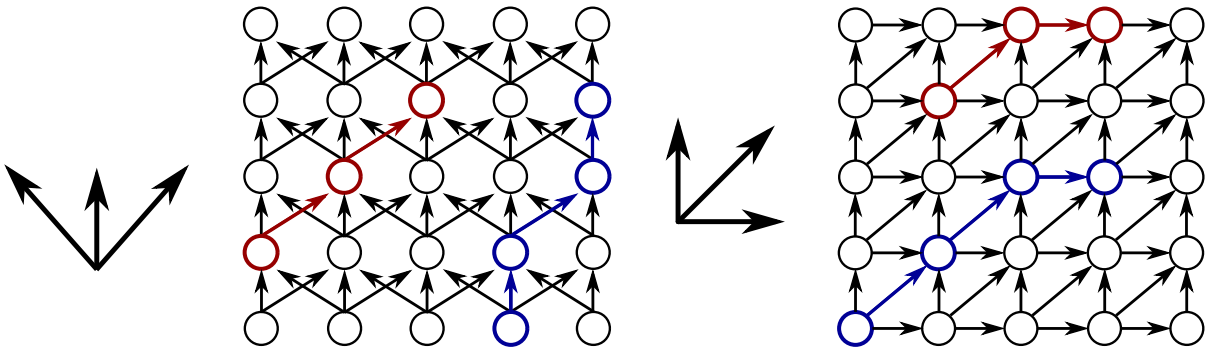
2.1 Path Opening

A path opening is an algebraic opening, but instead of using one structuring element with a fixed shape and size, the path opening uses a family of structuring elements called *paths*.

Let X be the set of points of an image. We note $\sigma(\pi)$ the set of points successively constituting a path π between two points of X with respect to an adjacency relation Γ. Figure 2 shows an example of two adjacency relations and their resulting graphs.

We note Π_L^Γ the set of all paths of length L on X with the adjacency relation Γ. The adjacency relation defines a global orientation for all paths of Π_L^Γ. For example, in Figure 2, Γ_1 allows for globally vertical paths, whereas Γ_2 allows for globally diagonal paths. In the following, we call Γ the

[2] http://path-openings.github.io/RORPO



(a) Γ_1 (b) G_1 (c) Γ_2 (d) G_2

Figure 2: Two examples of adjacency relations Γ_i and the graphs G_i built from them. On each graph, two paths of size 3 (in red) and 4 (in blue) points are shown.

orientation of the path opening.

The binary path opening α_L^Γ is defined as the union of all the paths of length L on a binary image I_b (see Equation (1)). This operator preserves each point of I_b belonging to at least one path of Π_L^Γ.

$$\alpha_L^\Gamma(I_b) = \bigcup \left\{ \sigma(\pi),\ \pi \in \Pi_L^\Gamma(I_b) \right\}. \tag{1}$$

A conceptually simple, but algorithmically inefficient, way to extend the path opening to gray-levels is to consider the set of all the successive thresholds of a gray-scale image, and perform, for each one, a binary path opening. Let G be the set of gray-levels of an image I, and $I_\lambda = \{x \in X, I(x) \geq \lambda\}$ the binary image resulting from the thresholding of I at gray-level $\lambda \in G$. The gray-level path opening A_L^Γ is defined by

$$A_L^\Gamma(I)(x) = \max_{\lambda \in G} \left\{ \lambda,\ x \in \alpha_L^\Gamma(I_\lambda) \right\}. \tag{2}$$

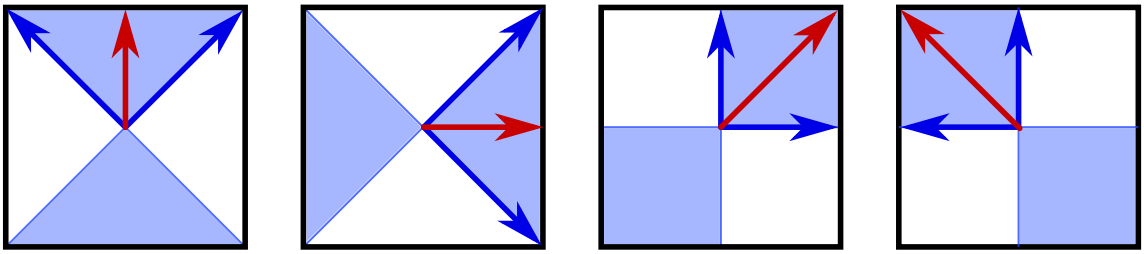
A path opening A_L^Γ with orientation Γ preserves the curvilinear structures which present a length of at least L pixels and an orientation compatible with Γ. To preserve the curvilinear structures lying in any orientation, the union of several path openings with different orientations is required

$$A_L(I) = \bigvee_{o \in \mathcal{O}} A_L^o(I), \tag{3}$$

where $\mathcal{O}$ is a set of path opening orientations. In 2D, four path opening orientations are generally used to preserve curvilinear structures in all orientations. These orientations are presented in Figure 3.

2.2 General Strategy

In 2D, two types of structures can be found: curvilinear structures which are elongated in a direction, and blob-like structures, which are approximately isotropic. A path opening in a given orientation preserves the structures where at least one path can fit. This means that a path opening does not only preserve curvilinear structures, but also the blob-like ones with a diameter greater than the path length. Our goal is to design a filter based on path openings which specifically detects curvilinear structures. To this aim, our strategy relies on counting the number of path opening orientations detecting a structure, to distinguish between a blob-like and a curvilinear structure.



(a) o_1: vertical orientation

(b) o_2 : horizontal orientation

(c) o_3: South/West - North/East diagonal orientation

(d) o_4: South/East - North/West diagonal orientation

Figure 3: The 4 path opening orientations in 2D. The arrows represent the adjacency relation. A path in a given orientation, going through the center point of the square, lies in the light blue shape. The red arrow is the main vector, i.e. the global orientation of any path lying in this path opening orientation.

2.3 Intensity Feature

To count the number of path opening orientations detecting a structure, we start by ranking the path opening responses. Let $\gamma_L^i(I)$ be the image obtained by applying on the path opening responses $\left(A_L{}^o\right)_{o\in\mathcal{O}}$ a point-wise rank filter RF_i, of order i

$$\gamma_L^i(I)(x) = RF_i\{A_L^o(I)(x),\ o \in \mathcal{O}\}. \tag{4}$$

In particular, RF_1 and RF_4 are respectively the point-wise maximum and minimum operators, and we have $\forall x \in X$, $\gamma_L^1(I)(x) \geq \gamma_L^2(I)(x) \geq \gamma_L^3(I)(x) \geq \gamma_L^4(I)(x)$.

A structure detected in only one path opening response is present in the first ranked path opening image $\gamma_L^1(I)$, but absent from the others. More generally, a structure detected in i path opening responses is present in the i^{th} first ranked path opening results.

Based on this observation, we define the RORPO intensity feature Φ_L as follows

$$\Phi_L(I) = \gamma_L^1(I) - \gamma_L^4(I). \tag{5}$$

$\gamma_L^1(I)$ contains all the structures detected in at least one path opening response, while $\gamma_L^4(I)$ contains the structures detected in all the path opening responses, i.e. all the isotropic, blob-like structures. Defining our intensity feature as the residue of these ranked images allows for preserving all the non-isotropic structures, i.e. the curvilinear structures.

2.4 Directional Feature

Along with the intensity feature, RORPO is also able to provide, for each point of a curvilinear structure, an estimation of its orientation. Experimentally, we observed that a curvilinear structure is usually detected by 1, 2 or 3 path opening orientation(s), called *orientation(s) of interest*, each one coded by its main vector (the red arrow in Figure 3). By combining the main vector of each orientation of interest, we are able to provide a useful estimation of the orientation of the curvilinear structure.

An orientation of interest o is characterized by a high path opening value A_L^o, whereas the orientations which do not detect the curvilinear structure should present a low path opening value. The first step is then to classify, for each pixel, the four path opening responses into orientations of interest and other orientations. This is a binary classification problem that we chose to solve by minimizing the intraclass standard deviation. This approach is similar to the Otsu thresholding method. More

formally, let $\mathcal{C}$ be the set of all the combinations of 1, 2 or 3 orientations of $\mathcal{O}$. One combination of $\mathcal{C}$ is the set of the orientations of interest we seek. Let $P = \{p_1, p_2, \ldots, p_k\}$, $1 \leq k \leq 3$, be a combination of $\mathcal{C}$ ($P \in \mathcal{C}$). We note $\sigma_P(x)$, the standard deviation of the path opening responses with the orientations of P at pixel x

$$\sigma_P(x) = \sqrt{\frac{1}{k}\sum_{j=1}^{k}(A_L^{p_j}(I)(x) - \mu(x))^2}, \qquad \mu(x) = \frac{1}{k}\sum_{j=1}^{k} A_L^{p_j}(I)(x). \tag{6}$$

Then, we find the orientations of interest for each pixel x by solving the following problem

$$\underset{P \in \mathcal{C}}{\text{minimize}}\ \sigma_P(x) + \sigma_{\mathcal{O}\setminus P}(x). \tag{7}$$

Thanks to the ranking procedure of the path opening results required to compute the intensity feature, the number of combinations in $\mathcal{C}$ is small. We call O_i the orientation along which the ranked path opening $\gamma_L^i(I)$ was computed ($\gamma_L^i(I) = A_L^{O_i}(I)$). Then, we have $\mathcal{C} = \{\{O_1\}, \{O_1, O_2\}, \{O_1, O_2, O_3\}\}$.

The combination of the main vectors of each orientation of interest can not be performed by a simple vector sum. Indeed, a vector has a *direction*, which is more specific than an *orientation*. Each orientation may be encoded by 2 different vectors. For example, if a curvilinear structure has an horizontal orientation, both vectors $[0, 1]$ and $[0, -1]$ encode its orientation. However, a vector sum with either one vector or the other would not be the same (see Figure 4).

The combination of the orientations of interest first consists of a correction of the main vectors, then a vector sum. The pseudo-code of the correction is presented in Algorithm 6.

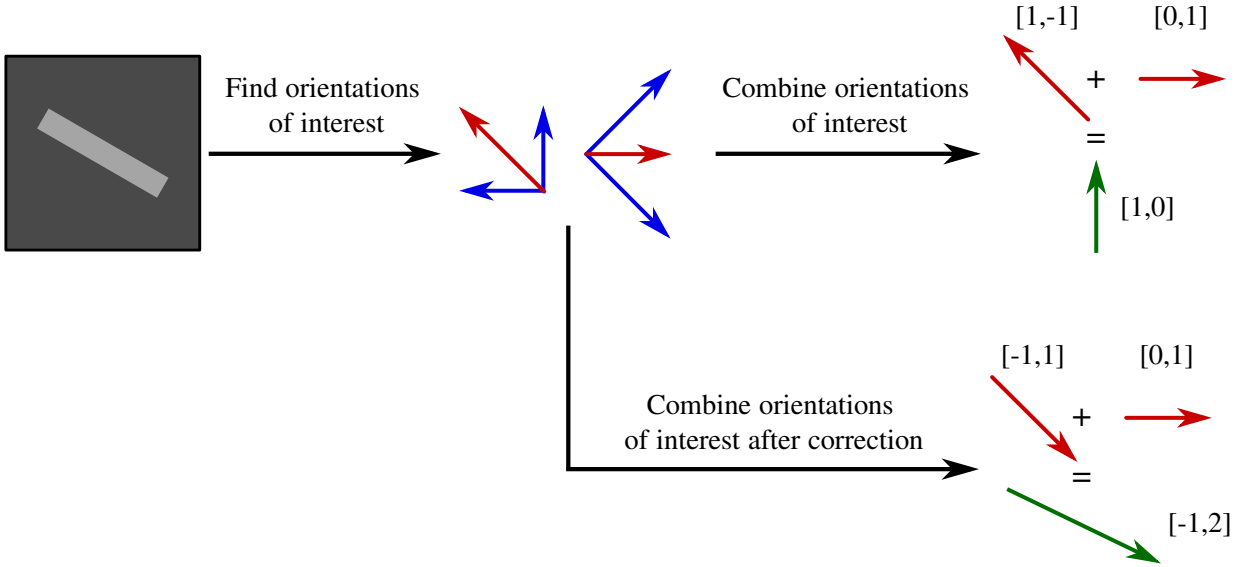


Figure 4: Without correction of the main vectors, the final curvilinear structure direction is incorrect ($[1, 0]$). We first need to switch one of the main vectors from $[1, -1]$ to $[-1, 1]$ (both have the same orientation), and then sum them to obtain the correct curvilinear direction ($[-1, 2]$).

2.5 Robustness to Noise

Path openings are sensitive to noise. A version of path opening robust to noise, called Robust Path Opening (RPO) [2], was developed by Cokelaer et al. However, it requires a relatively costly algorithmic layer.

We proposed an alternative to RPO based on a mask-based second-generation connectivity strategy [13]. A dilation by a square structuring element of size $N \times N$ is first performed on the initial image. This dilated image is later used to compute the regular path openings. To ensure anti-extensivity, the final result is the minimum between the path opening and the initial image. This is very similar to the RPO strategy which uses paths, where a fixed number of noisy pixels in a row can be ignored. Intuitively, instead of allowing for noisy pixels inside paths, we compute paths on the dilated image on which noisy pixels have been turned into structure pixels (see Figure 5).

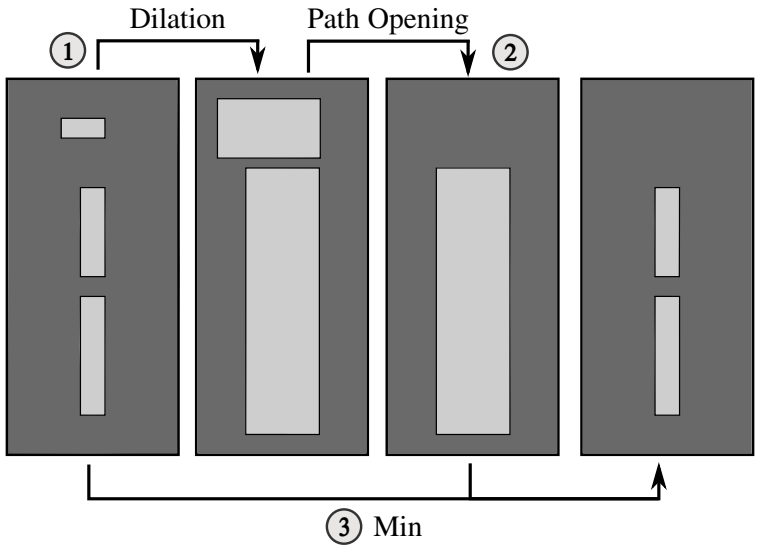


Figure 5: Illustration of the simplified robust path opening. A curvilinear structure is disconnected by a noisy pixel. After dilation, this pixel is no longer considered as noise, but belongs to the curvilinear structure, which allows the path opening to preserve the whole curvilinear structure. A minimum operator is finally applied to ensure anti-extensivity.

2.6 Multiscale Filtering

To preserve curvilinear structures of various scales, a multiscale procedure is required. Usually, the scale parameter for curvilinear structures is the diameter of the structure. However, we experimentally observed that a multiscale procedure based on the path length is equivalent in most applications.

The path length detecting a curvilinear structure depends on its degree of curvature. The greater the curvature, the lower the path length must be to detect the structure (see Figure 6).

In many applications, the diameter, curvature and length of curvilinear structures are highly correlated, for physical reasons. For example, small blood vessels are generally more tortuous and shorter than large vessels like the aorta. The same argument can be made for insulation fibers or roads vs. highways. Based on this hypothesis, we developed a multiscale procedure using the path length. The multiscale RORPO intensity feature $\Phi(I)$ is defined for a set of scales $S = \{L_1, L_2, \dots L_n\}$ as follows

$$\Phi(I) = \bigvee_{L \in S} \Phi_L(I). \tag{8}$$

This multiscale paradigm also applies to the RORPO directional feature. For each scale, a direction can be computed, but the final direction is the one associated to the greatest RORPO intensity response, and with the lowest scale.

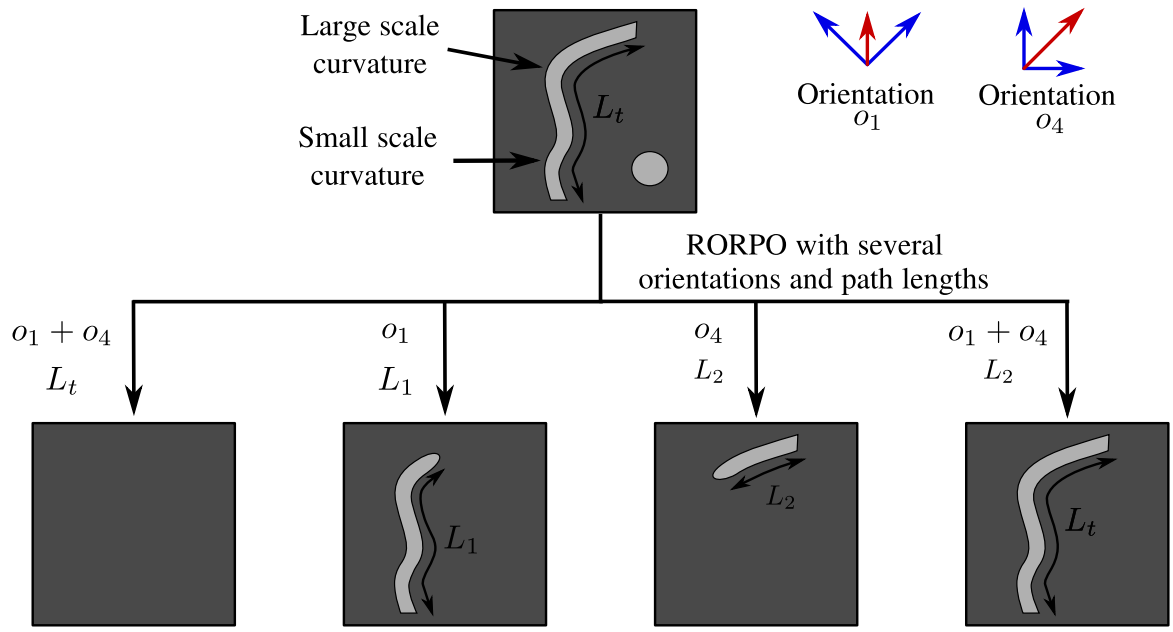


Figure 6: A curvilinear structure with a large-scale curvature cannot be preserved by only one path opening. Path openings with several orientations and a path length lower than the real curvilinear structure length are required.

3 Algorithm

In this section, we first recall the Luengo Hendriks Algorithm [9] for computing a path opening. Then we present the algorithm to compute both RORPO features.

3.1 Path Opening Algorithm

Our implementation uses the dimensionality-independent path opening algorithm of Luengo Hendriks. We recall the whole algorithm in this section (see Algorithms 1 and 2). The reader may refer to [9] for a complete presentation.

3.2 RORPO Algorithm

This section presents the RORPO algorithm. Algorithm 3 presents an overview of RORPO with the multiscale procedure. Algorithm 4 details the pointwise ranking to compute the intensity feature and Algorithm 5 shows the directional feature computation.

In Algorithm 5, before the combination of the main vectors (vector_sum), we required a vector correction due to the difference between orientations and directions (see Section 2.4). This correction, developed in Algorithm 6, consists of choosing the vectors with minimal sum of their pairwise angles.

3.3 Complexity

The RORPO algorithm complexity is dominated by the Luengo Hendriks path opening complexity. For one scale, it is $O(|\Omega| \log(L))$ with L the path length and $|\Omega|$ the number of pixels in the image. The robustness step only requires infimum/supremum operations which have a linear cost in $O(|\Omega|)$, like the pixelwise sorting procedure. Validations of the RORPO complexity in the 3D case may be found in [10].

An easy way to decrease the RORPO computation time is to parallelize the computation of the 4 path openings, which are completely independent.

Algorithm 1: compute_PO

input : Dilated image I_d, orientation o and path length L.
output: Path opening result I^o_{PO}.
Create list of offsets n^+ to upstream neighbors and n^- to downstream neighbors according to orientation o.
Create list i of indices to all pixels of image I_d except border pixels.
Sort i according to value of $I_d(i)$.
Create temporary images b, λ^+ and λ^-.
Initialize for each pixel x, $b(x) \leftarrow true$, $\lambda^+(x) \leftarrow L$, $\lambda^-(x) \leftarrow L$ and $I^o_{\mathrm{PO}}(x) \leftarrow I_d(x)$.
Initialize for each border pixel p_b, $b(p_b) \leftarrow false$.
Initialize a First In First Out (FIFO) queue Q_c.
foreach $p \in i$ *for which* $b(p) = true$ **do**
    $Q_c \leftarrow \mathrm{propagate}(p, \lambda^-, n^+, n^-, Q_c)$
    $Q_c \leftarrow \mathrm{propagate}(p, \lambda^+, n^-, n^+, Q_c)$
    foreach $q \in Q_c$ **do**
        if $\lambda^+(q) + \lambda^-(q) - 1 < L$ **then**
            $I^o_{\mathrm{PO}}(q) \leftarrow I^o_{\mathrm{PO}}(p)$
            $b(q) \leftarrow false$, $\lambda^+(q) \leftarrow 0$, $\lambda^-(q) \leftarrow 0$

Algorithm 2: propagate

input : pixel p, path propagation λ, offsets lists n_f, n_b, FIFO queue Q_c.
output: FIFO queue Q_c.
$\lambda(p) \leftarrow 0$
Initialize a FIFO queue Q_q.
Enqueue in Q_q all neighbors $p_f = p + n_f$ for which $b(p_f) = true$.
foreach $q \in Q_q$ **do**
    $l \leftarrow \bigvee_i \lambda(q + n_b(i)) + 1$
    if $l < \lambda(q)$ **then**
        $\lambda(q) \leftarrow l$
        Enqueue in Q_q all neighbors $q_f = q + n_f$ for which $b(q_f) = true$
        Enqueue q in Q_c

4 Implementation

In this section, we explain how to use the C++ implementation we provide along with the online IPOL demo.

4.1 Requirements

Our C++ implementation only depends on the *libpng* library (`http://www.libpng.org/pub/png/libpng.html`). We also provide a `CMakeList` to compile the code, requiring at least `cmake 2.8` (`https://cmake.org/download/`).

4.2 Input/Output

The code requires 6 inputs:

Algorithm 3: RORPO algorithm

input : Image $I : E \to G$, set of scales $S = \{L_1, L_2, \ldots, L_n\}$,
set of the 4 orientations $O = \{o_1, o_2, o_3, o_4\}$, size k of the structuring element.
output: Intensity feature RORPO and directional feature d.
Initialize for each pixel x, $RORPO(x) \leftarrow 0$.
foreach *scale* $L \in S$ **do**
  $I_d \leftarrow \delta_{B_k}(I)$; *Dilation of I with structuring element B_k.*
  foreach $o \in O$ **do**
    $I^o_{\mathrm{PO}} \leftarrow \mathrm{compute\_PO}(I_d, o, L)$; *Compute the 4 PO.*
    $I^o_{\mathrm{PO}} \leftarrow \min(I, I^o_{\mathrm{PO}})$; *Minimum with the initial image for the robust PO (RPO).*
  $\left(I^i_{\mathrm{rank}}, O^i\right)_{i\in[1,4]} \leftarrow \mathrm{ranking}\Big(\left(I^o_{\mathrm{PO}}\right)_{i\in[1,4]}, (o_i)_{i\in[1,4]}\Big)$; *Pointwise rank filter.*
  foreach $x \in E$ **do**
    $\mathrm{RORPO}_L(x) \leftarrow \left(I^1_{\mathrm{rank}}(x) - I^4_{\mathrm{rank}}(x)\right)$; *Intensity feature at scale L.*
    $d_L(x) \leftarrow \mathrm{compute\_directions}\Big(\left(I_{\mathrm{rank}}(x), O^i(x)\right)_{i\in[1,4]}\Big)$; *Directional feature at scale L.*
    if $\mathrm{RORPO}_L(x) > \mathrm{RORPO}(x)$ **then**
      $\mathrm{RORPO}(x) \leftarrow \mathrm{RORPO}_L(x)$; *Multiscale intensity feature.*
      $d(x) \leftarrow d_L(x)$; *Multiscale directional feature.*

Algorithm 4: ranking

input : The 4 path opening images $\left(I^{o_i}_{\mathrm{PO}}\right)_{i\in[1,4]}$ and the 4 path opening orientations $(o_i)_{i\in[1,4]}$.
output: The 4 ranked path opening results $\left(I^i_{\mathrm{rank}}\right)_{i\in[1,4]}$ and the orientations along which it were computed $\left(O^i\right)_{i\in[1,4]}$.
foreach $x \in E$ **do**
  foreach $k \in [1, 4]$ **do**
    $I^k_{\mathrm{rank}}(x) \leftarrow \underset{p\in[1,4]}{\mathrm{RF}_k}\left(I^{o_p}_{\mathrm{PO}}(x)\right)$; RF_k *returns the k^{th} maximal value.*
    $p^\star \leftarrow \underset{p\in[1,4]}{\mathrm{argRF}_k}\left(I^{o_p}_{\mathrm{PO}}(x)\right)$; /* argRF_k *returns the orientation indice p*
    $O^k(x) \leftarrow o_{p^\star}$; *giving the k^{th} maximum value $I^{o_p}_{PO}(x)$.*/

- **the path of the initial image**: it should point to a 2D gray-level 8 bits PNG image;
- **the minimum path length,** L_{min}: an integer;
- **the scale factor,** $factor$: a floating point value;
- **the number of scales,** $nbScales$: an integer;
- **the robustness parameter,** R: an integer, usually 0 or 1;
- **the path of the resulting image**.

Remark on the initial image. The code is based on path openings. The initial image should then contain bright structures on dark background. For images with dark structures on bright background the reader should first invert the image.

With these parameters, RORPO is computed with several scales. The path length for each scale

Algorithm 5: compute_directions

input : The 4 ranked path opening results $(I^i_{\text{rank}})_{i\in[1,4]}$ and ranked orientations, $(O^i)_{i\in[1,4]}$.
output: Directional feature d.
foreach $x \in E$ **do**

$g_1 \leftarrow \sigma\Big((I^i_{\text{rank}}(x))_{i\in[2,4]}\Big)$; /*Compute the intraclass*
$g_2 \leftarrow \sigma\Big((I^i_{\text{rank}}(x))_{i\in[1,2]}\Big) + \sigma\Big((I^i_{\text{rank}}(x))_{i\in[3,4]}\Big)$; *standard deviations*
$g_3 \leftarrow \sigma\Big((I^i_{\text{rank}}(x))_{i\in[1,3]}\Big)$; *according to Equation 7.*/*
$g^\star \leftarrow \underset{i\in[1,3]}{\text{argmin}}(g_i)$; *Keep the class minimizing the intraclass std. deviation.*
$\text{Vec} \leftarrow \text{vector\_correction}\Big((O_i(x))_{i\in[1,g^\star]}\Big)$
$d(x) \leftarrow \text{vector\_sum}\Big(\text{Vec}\Big)$; *Compute the final direction.*

is computed with respect to the following equation

$$L_n = L_{min} \times factor^{n-1}, \tag{9}$$

with L_n the path length at scale n, $\forall n \in [1, nbScales]$.

Three gray-level PNG images are written on the file system by RORPO2D. The first is the resulting intensity feature, which is written at the location provided by the first argument. The other two images encode the x and y coordinates of the directional feature respectively. For example, if the intensity feature is called `output.png`, then `output_vx.png` (resp. `output_vy.png`) is the image where each pixel value is the x (resp. y) coordinate of the vector representing the local orientation of a curvilinear structure at this pixel. The directional feature is usually encoded by normalized vectors (i.e. with values between -1 and 1). However, PNG images support neither negative nor floating-point values. The directional feature values written in the PNG images are then normalized between 0 and 200. To retrieve the original value, one should first divide the PNG value by 100, then substract 1.

4.3 Setting the Parameters

RORPO only requires two parameters: the path length L (or several path lengths for the multiscale processing) and the robustness parameter R.

The path length is a meaningful parameter which is related to the size of the structures one wants to detect. In the following, we present a few advices to set the path length:

- The lower the path length, the more undesired structures may be preserved.
- The lower the path length, the better the local accuracy of the directional feature, but the more sensitive to noise the directional feature.
- One or two scales are usually sufficient, however, the greater the curvilinear structure tortuosity, the more scales should be used.

The robustness parameter R controls the amount of noise accepted during the path opening computation. The greater R, the higher the number of detected structures (curvilinear structures, but also possibly noise or blob-like structures). Usually, we set R to 0 (the noise is not allowed in the path opening computation) or 1 (a maximum of 2 noise pixels in a row are accepted in a path). Higher values of R generally result in too many false positive detections.

Algorithm 6: vector_correction

input : The set of ranked orientations $(O^i(x))_{i\in[1,g^*]}$.
output: A $g^* \times 2$ array containing the set of corrected vectors Vec.
sumAngle $\leftarrow 360$
Create one (if $g^* = 1$), two (if $g^* = 2$) or three (if $g^* = 3$) 1×2 arrays (v_1, v_2, v_3), containing the main vectors of $(O^i(x))_{i\in[1,g^*]}$. For example if $O^1(x) = o_1$ the vertical orientation, then $v_1[0] \leftarrow 1$ and $v_1[1] \leftarrow 0$.
$Vec[0,0] \leftarrow v_1[0]$; *Fix the first vector and correct the others according to this one.*
$Vec[0,1] \leftarrow v_1[1]$
if $g^\star = 1$; *# 1 orientations of interest #*
then
  // *No correction is required*
if $g^\star = 2$; *# 2 orientations of interest #*
then
  for $i \in [1, -1]$ **do**
    sum $\leftarrow$ angle$(v_1, i.v_2)$; *Compute the angle between v_1 and $i.v_2$.*
    if *sum < sumAngle;* *Look for the minimal sum of pairwise angles.*
    then
      sumAngle $\leftarrow$ *sum*
      $Vec[1,0] \leftarrow i.v_2[0]$; *Correct the main vectors.*
      $Vec[1,1] \leftarrow i.v_2[1]$
else if $g^\star = 3$; *# 3 orientations of interest #*
then
  Create three 1×2 arrays v_1, v_2 and v_3, containing the main vectors of $(O^i(x))_{i\in[1,g^*]}$.
  for $i \in [1, -1]$ **do**
    for $j \in [1, -1]$ **do**
      //*Compute the sum of pairwise angle between v_1, $i.v_2$, $j.v_3$.*
      sum $\leftarrow$ angle$(v_1, i.v_2)$ + angle$(v_1, j.v_3)$ + angle$(i.v_2, j.v_3)$
      if *sum < sumAngle ;* *Look for the minimal sum of pairwise angles.*
      then
        sumAngle $\leftarrow$ *sum*
        $Vec[1,0] \leftarrow i.v_2[0]$; *Correct the main vectors.*
        $Vec[1,1] \leftarrow i.v_2[1]$
        $Vec[2,0] \leftarrow j.v_3[0]$
        $Vec[2,1] \leftarrow j.v_3[1]$

4.4 Usage

To compile the code with `cmake`:

- Create a build directory
- Inside this directory, precompile the project: `cmake path_to_the_CMakeLists.txt`
- Compile the project: `make`

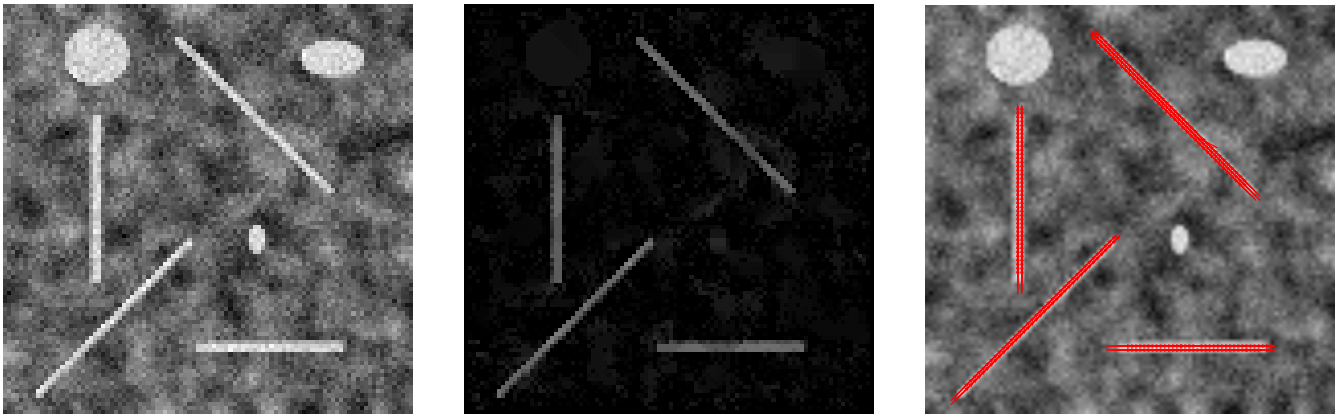
The compilation generates an executable called RORPO2D which should be used as follows:

```
RORPO2D <inputPath> <Lmin> <factor> <nbScales> <R> <outputPath>
```

Usage examples:

```
RORPO2D input.png 25 1.5 3 0 output.png
```

To visualize the directional feature, we provide a Python script `plot_directional_feature.py` which uses `matplotlib`.



(a) Image (b) Intensity feature (c) Directional feature

Figure 7: A synthetic image (a), its intensity (b) and directional (c) features from RORPO with parameters $L_{min} = 30$, $factor = 1$, $nbScales = 1$, $R = 1$. The directional feature (red vectors) is plotted upon the initial image.

5 Examples

In this section, we present a few results obtained with RORPO. The reader should note that we plotted the directional feature only inside the curvilinear structures. When the ground truth of the curvilinear structures is not available, one may consider the directional feature only inside a threshold of the intensity feature.

We applied RORPO on a synthetic image containing both curvilinear and blob-like structures, and on two real images: a retinal image of the Drive database [15] and an image of surface fissures from very-high resolution aerial images [12, 16]. The results are shown in Figures 7, 8 and 9.

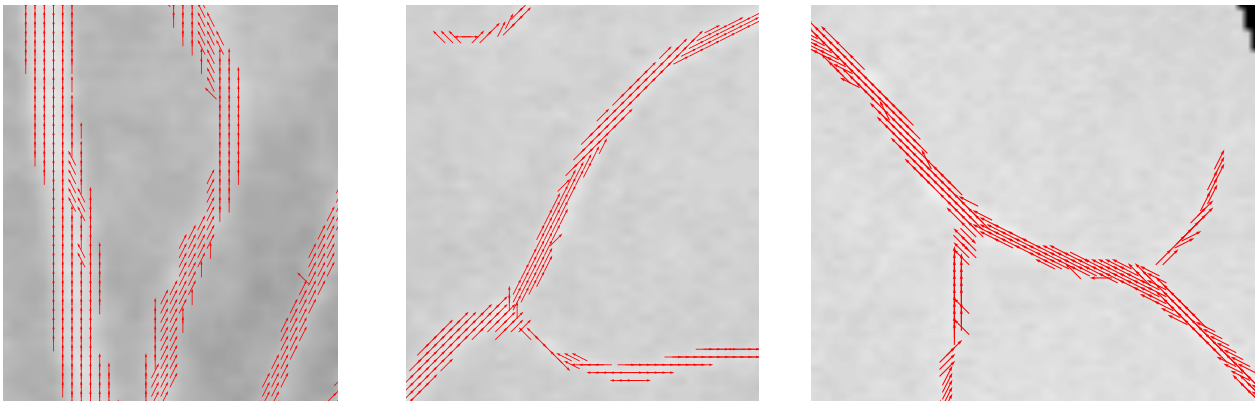
RORPO successfully removes the blob-like structures, while preserving the curvilinear structures. Moreover, most of the noise and the background intensity was also reduced, which significantly enhances the local contrast of the curvilinear structures. The directional feature provides a good estimation of the local orientation of the curvilinear structures, despite the small number of orientations used.

6 Limitations

RORPO is a powerful filter; however it presents a few limitations:

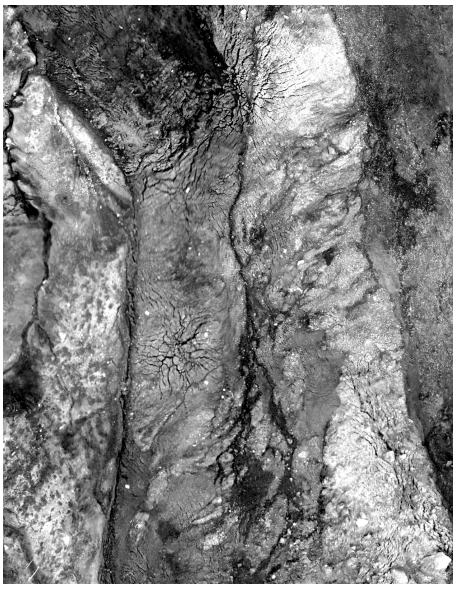
- RORPO detects separately bright structures on dark background or dark structures on bright background. If the structures of interest are both dark and white, both results must be combined, for instance by supremum.
- The orientation estimation may present outliers, usually due to phantom structures induced by the noise or low contrast. The weight of these outliers can be reduced by averaging the orientations in local patches.

(a) Image (b) Intensity feature

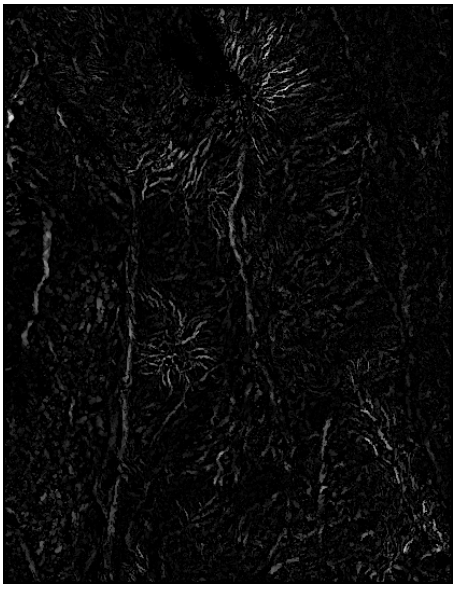


(c) Directional feature, zoom 1 (d) Directional feature, zoom 2 (e) Directional feature, zoom 3

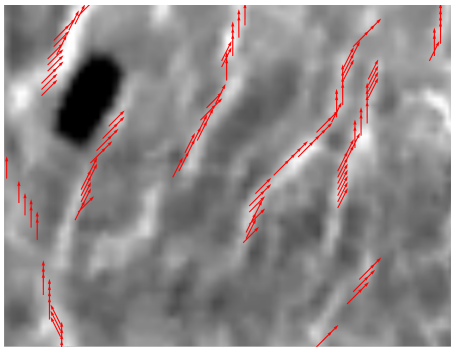
Figure 8: A retinal image (a), its intensity (b) and directional (c–e) features from RORPO, with parameters $L_{min} = 25$, $factor = 1.5$, $nbScales = 2$, $R = 1$. For the sake of clarity, only three zooms of the directional feature are shown, and are plotted upon the initial image.



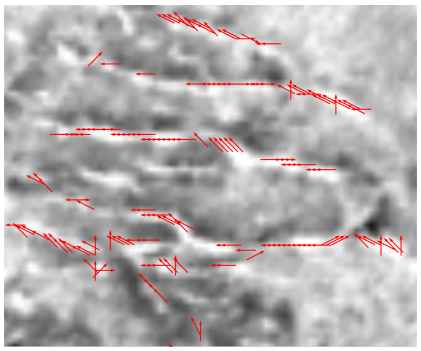
(a) Image of fissures



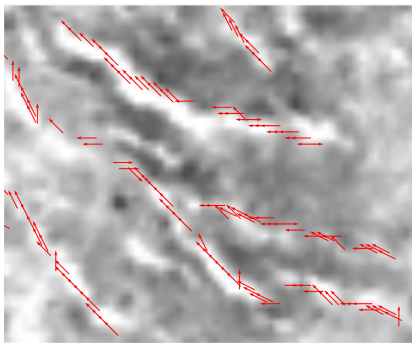
(b) Intensity feature



(c) Directional feature, zoom 1



(d) Directional feature, zoom 2



(e) Directional feature, zoom 3

Figure 9: An image of surface fissures [12, 16] (a), its intensity (b) and directional (c–e) features from RORPO, with parameters $L_{min} = 10$, $factor = 1$, $nbScales = 1$, $R = 1$. For the sake of clarity, only three zooms of the directional features are shown on the negative of the initial image.

Acknowledgments

This work was partially funded with French *Agence Nationale de la Recherche* grant agreement ANR-12-MONU-0010 (VIVABRAIN Project: http://vivabrain.fr).

Image Credits

Images by the authors except:

 ©American Society for Photogrammetry and Remote Sensing [14]

 ©IEEE [15]

 ©Elsevier [16]

References

[1] G. Agam and C. Wu, *Probabilistic modeling based vessel enhancement in thoracic CT scans*, in IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, 2005, pp. 649–654. http://dx.doi.org/10.1109/CVPR.2005.280.

[2] F. Cokelaer, H. Talbot, and J. Chanussot, *Efficient robust d-dimensional path operators*, IEEE Journal of Selected Topics in Signal Processing, 6 (2012), pp. 830–839. http://dx.doi.org/10.1109/JSTSP.2012.2213578.

[3] A.F. Frangi, W.J. Niessen, K.L. Vincken, and M.A. Viergever, *Multiscale vessel enhancement filtering*, in Medical Image Computing and Computer-assisted Intervention, vol. 1496, 1998, pp. 130–137. http://dx.doi.org/10.1007/BFb0056195.

[4] G. González, F. Aguet, F. Fleuret, M. Unser, and P. Fua, *Steerable features for statistical 3D dendrite detection*, in Medical Image Computing and Computer-assisted Intervention, 2009, pp. 625–632. http://dx.doi.org/10.1007/978-3-642-04271-3_76.

[5] H.J.A.M. Heijmans, M. Buckley, and H. Talbot, *Path openings and closings*, Journal of Mathematical Imaging and Vision, 22 (2005), pp. 107–119. http://dx.doi.org/10.1007/s10851-005-4885-3.

[6] T. M. Koller, G. Gerig, G. Szekely, and D. Dettwiler, *Multiscale detection of curvilinear structures in 2-D and 3-D image data*, in IEEE International Conference on Computer Vision, 1995, pp. 864–869. http://dx.doi.org/10.1109/ICCV.1995.466846.

[7] K. Krissian, *Flux-based anisotropic diffusion applied to enhancement of 3-D angiogram*, IEEE Transactions on Medical Imaging, 21 (2002), pp. 1440–1442. http://dx.doi.org/10.1109/TMI.2002.806403.

[8] M.W.K. Law and A.C.S. Chung, *Three dimensional curvilinear structure detection using optimally oriented flux*, in European Conference on Computer Vision, 2008, pp. 368–382. http://dx.doi.org/10.1007/978-3-540-88693-8_27.

[9] C.L. Luengo Hendriks, *Constrained and dimensionality-independent path openings*, IEEE Transactions on Image Processing, 19 (2010), pp. 1587–1595. http://dx.doi.org/10.1109/TIP.2010.2044959.

[10] O. Merveille, *RORPO: A morphological framework for curvilinear structure analysis. Application to the filtering and segmentation of blood vessels*, thesis, Université Paris–Est, 2016. https://hal.archives-ouvertes.fr/tel-01462887/document.

[11] O. Merveille, H. Talbot, L. Najman, and N. Passat, *Curvilinear structure analysis by ranking the orientation responses of path operators*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2017). http://dx.doi.org/10.1109/TPAMI.2017.2672972.

[12] U. Niethammer, M.R. James, S. Rothmund, J. Travelletti, and M. Joswig, *UAV-based remote sensing of the Super-Sauze landslide: Evaluation and results*, Engineering Geology, 128 (2012), pp. 2–11. http://dx.doi.org/10.1016/j.enggeo.2011.03.012.

[13] G.K. Ouzounis and M.H.F. Wilkinson, *Mask-based second-generation connectivity and attribute filters*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 990–1004. http://dx.doi.org/10.1109/TPAMI.2007.1045.

[14] M. Song and D. Civco, *Road extraction using SVM and image segmentation*, Photogrammetric Engineering & Remote Sensing, 70 (2004), pp. 1365–1371. http://dx.doi.org/10.14358/PERS.70.12.1365".

[15] J.J. Staal, M.D. Abramoff, M. Niemeijer, M.A. Viergever, and B. van Ginneken, *Ridge based vessel segmentation in color images of the retina*, IEEE Transactions on Medical Imaging, 23 (2004), pp. 501–509. http://dx.doi.org/10.1109/TMI.2004.825627.

[16] A. Stumpf, J.P. Malet, N. Kerle, U. Niethammer, and S. Rothmund, *Image-based mapping of surface fissures for the investigation of landslide dynamics*, Geomorphology, 186 (2013), pp. 12–27. http://dx.doi.org/10.1016/j.geomorph.2012.12.010.