Planting, Growing, and Pruning Trees: Connected Filters Applied to Document Image Analysis

Guillaume Lazzara, Thierry Géraud, Roland Levillain
EPITA Research and Development Laboratory (LRDE)
14–16, rue Voltaire, FR-94276 Le Kremlin-Bicêtre Cedex, France
E-mail: *firstname.lastname*@lrde.epita.fr

***Abstract*—Mathematical morphology, when used in the field of document image analysis and processing, is often limited to some classical yet basic tools. The domain however features a lesser-known class of powerful operators, called connected filters. These operators present an important property: they do not shift nor create contours. Most connected filters are linked to a tree-based representation of an image's contents, where nodes represent connected components while edges express an inclusion relation. By computing attributes for each node of the tree from the corresponding connected component, then selecting nodes according to an attribute-based criterion, one can either filter or recognize objects in an image. This strategy is very intuitive, efficient, easy to implement, and actually well-suited to processing images of magazines. Examples of applications include image simplification, smart binarization, and object identification.**

***Keywords*—Document Image Processing, Mathematical Morphology, Tree of Shapes, Image Simplification, Binarization, Object Identification.**

I. Introduction

Mathematical morphology [1], [2], [3], [4] has been part of classical image processing techniques for more than 40 years. Morphological operators are regularly used in the field of document image processing and analysis [5], [6]. Indeed documents contain objects that may be identified thanks to prominent features of their shapes. Mathematical morphology comes with a panel of non-linear operators to transform the morphology of those objects, thus allowing their extraction. The following enumeration presents a simplistic and incomplete classification of morphological operators.

a) Operators on sets based on structuring elements: A binary image $1_F : \mathcal{D} \to \mathbb{B}$ is the indicator function of a set of points F. A structuring element is a set, say B, usually centered and with a limited definition domain; it acts like a filtering parameter for any transformation to be applied to F. Roughly put, the size of B influences the strength of the transformation, whereas the shape of B affects how the transformation modifies the shape of X. For instance, in the case of the dilation, defined by $\delta_B(F) = F \oplus B = \{ p + b \mid p \in F, b \in B \}$, when B is a centered horizontal line segment, the set X is enlarged horizontally of half the length of B. Many morphological operators have been defined, some of them being dual by set-complementation ($\complement$): for instance, the erosion operator ε_B is the dilation dual operator ($\varepsilon_B = \complement\, \delta_B\, \complement$).

b) Elementary operators on sets: Replacing the structuring element B by the notion of neighborhood, say $\mathcal{N}$, produces morphological operators whose behavior is elementary (the slightest transform effect). For instance $\delta_{\mathcal{N}}(F) = \{ p' \in \mathcal{N}(p) \cup \{p\} \mid p \in F \}$ is the elementary dilation.

c) Extension of the first two categories from sets to functions: Thanks to the threshold decomposition principle, any morphological operator on sets can be extended to functions, i.e., gray-level images: $f : \mathcal{D} \to \mathbb{N}$. Given a gray-level t, let us denote by $[f \geq t] = \{p \in \mathcal{D} \mid f(p) \geq t\}$ the subset of $\mathcal{D}$ obtained by thresholding f by t. Any morphological set operator φ^{set} can be extended to define a morphological operator on gray-level functions φ thanks to $\varphi(f)(p) = \max\{ t \mid p \in \varphi^{\text{set}}([f \geq t]) \}$.

d) Connected operators: Connected operators [7], [8] are morphological operators that do not split flat zones, i.e., that verify $\forall p, \forall p' \in \mathcal{N}(p),\ f(p') = f(p) \Rightarrow \varphi(f)(p') = \varphi(f)(p)$. As a consequence, they do not shift contours; they may just remove some of them while preserving the others. Two classes of connected operators are commonly used: filters by reconstruction, and algebraic openings and closings [9]. These operators are interesting in many ways, but we want to emphasize here one of their key features: their effect is computed by considering *all the connected components* obtained by thresholding the input image at different levels t. For instance, any algebraic closing works on the connected components ($\mathcal{CC}$) of the family of sets $\{ \Gamma \mid \Gamma \in \mathcal{CC}([f < t]) \}_{t \in \mathbb{N}}$.

In the light of this short classification, two remarks crop up. First, an overview of the literature shows that the community of document image processing mainly uses mathematical morphology for its structuring-element-based operators on binary images, i.e., operators from category a. Some very few authors rely on their gray-level extension, i.e. operators from category c. The two other categories are nearly ignored from the community (apart from recent exceptions [10], [11]). A second remark comes when considering the very classical scheme used in document image processing shown in Figure 1. The colored part depicts the pre-processing steps performed just before some high-level analysis leading to a page segmentation. This part, which can be justified by the need to rely on a small collection of binary objects to extract text lines and other entities, can be summarized by "thresholding first to get some components to start with". Such an approach highly contrasts with the way morphological connected operators work: those operators rely on *all the components* of gray-level images to take some decision (to filter), whereas the binarization step of Figure 1 *drastically reduces the number of components* to be taken into account for document analysis. This binarization step is already a first decision process, which *purposely* simplifies the data. By its very nature, this operation most probably creates an *a priori* loss of information and therefore

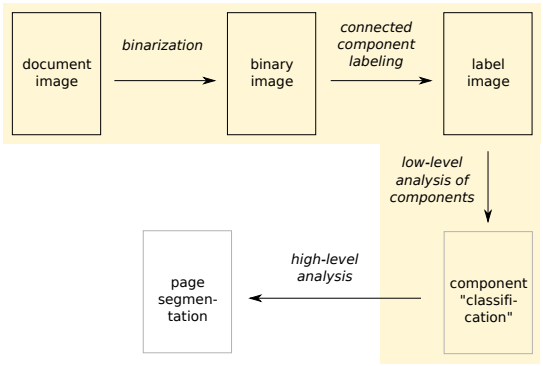
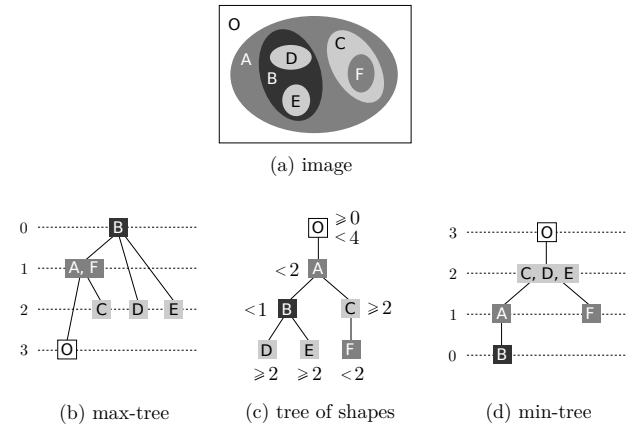


Fig. 1: A typical processing workflow starting from a document image and leading to a page segmentation.



(a) image (b) max-tree (c) tree of shapes (d) min-tree

Fig. 2: Three morphological trees of the same image. Light (resp. dark) gray values represent high (resp. low) integer values.

places limits on the efficiency of subsequent steps. Let us note that it is admittedly better to make decisions at the end of an entire process, where one can leverage information from many indicators, than at its beginning.

This paper is more a methodological article than a presentation of a specific research result. It focuses on the benefits of a little-known branch of mathematical morphology for the document image analysis community. Thus it does not contain any detail about how to precisely obtain some particular result (and *a fortiori* does not evaluate and compare them to those of the literature). This paper is structured as follows. First, Section II presents the tree-based image representations underpinning connected filters. Section III explains how to implement connected filters by pruning these trees. We then show applications of these morphological operators to document images in Section IV to illustrate their benefits. Section V concludes and opens up perspectives regarding the use of mathematical morphology in document image processing.

II. Morphological Tree-Based Image Representations

Since connected filters have been formalized [8], they have not received a great deal of attention beyond the mathematical morphology community, although a recent effort [12] has tried to popularize them to a broader audience.

Connected filters are related to the decomposition principle. We will thereafter consider a gray-level valued image f, viewed as a function $\mathcal{D} \to \mathcal{V}$, where $\mathcal{D}$ is the *domain* of f, and $\mathcal{V}$, its *value set*. We will fix $\mathcal{V} = \mathbb{N}$ in the remainder of this paper to represent the gray-level values of f; however the properties and strategies explained hereinafter are valid for any totally ordered value set. As for the domain, $\mathcal{D}$ is usually set to a subset of $\mathbb{Z}^2$ in the case of classical 2D images (based on a rectangular grid).

A. Morphological Trees

Three types of morphological trees are used to represent images and implement connected operators.

1) A Couple of Dual Trees: The Max- and Min-Trees: For any $t \in \mathbb{N}$, the upper cuts (or upper thresholds) of f are defined by $[f \geq t] = \{x \in \mathcal{D} \mid f(x) \geq t\}$; we have $t_2 > t_1 \Rightarrow [f \geq t_2] \subseteq [f \geq t_1]$. The set of all connected components of every cut of f is $\mathcal{T}_{\geq}(f) = \{\Gamma \in \mathcal{CC}([f \geq t])\}_{t \in \mathbb{N}}$ and verifies that $\forall (\Gamma, \Gamma') \in (\mathcal{T}_{\geq}(f))^2$ such as $\Gamma \neq \Gamma'$, we either have $\Gamma \subset \Gamma'$ or $\Gamma' \subset \Gamma$ or $\Gamma \cap \Gamma' = \varnothing$. Therefore the elements of $\mathcal{T}_{\geq}(f)$ (the connected components of upper cuts) can be arranged into a tree, called the *max-tree* of f. Figure 2b is an example of max-tree, computed from the input image shown in Figure 2a. When the lower cuts $[f < t] = \{x \in \mathcal{D} \mid f(x) < t\}$ are considered, the elements of $\mathcal{T}_{<}(f) = \{\Gamma \in \mathcal{CC}([f < t])\}_{t \in \mathbb{N}}$ form the *min-tree* of f (see Figure 2c). Both trees are dual by complementation: the max-tree of f is the min-tree of $\complement f$.

2) One Self-Dual Tree: The Tree of Shapes: Two other sets, $\mathcal{S}_{<}(f)$ and $\mathcal{S}_{\geq}(f)$, respectively the sets of lower and upper shapes, can be defined respectively as the sets of components of $\mathcal{T}_{<}(f)$ and $\mathcal{T}_{\geq}(f)$ after filling the holes of these components. If we refer to the hole-filling (or saturation) operator as Sat, we have: $\mathcal{S}_{<}(f) = \{\mathrm{Sat}(\Gamma);\ \Gamma \in \mathcal{T}_{<}(f)\}$ and $\mathcal{S}_{\geq}(f) = \{\mathrm{Sat}(\Gamma);\ \Gamma \in \mathcal{T}_{\geq}(f)\}$. The set of all shapes $\mathfrak{S}(f) = \mathcal{S}_{<}(f) \cup \mathcal{S}_{\geq}(f)$ forms a tree, called the *tree of shapes* of f [13]. Indeed, for any couple of shapes $(\Gamma, \Gamma') \in (\mathfrak{S}(f))^2$ such as $\Gamma \neq \Gamma'$ we either have $\Gamma \subset \Gamma'$ or $\Gamma' \subset \Gamma$ or $\Gamma \cap \Gamma' = \varnothing$. Actually the shapes of f are the holes of the elements of $\mathcal{T}_{<}(f)$ and $\mathcal{T}_{\geq}(f)$. For instance, if we consider a lower component $\Gamma \in [f < t]$ and a hole H of Γ, this hole is an upper shape, i.e., $H \in \mathcal{S}_{\geq}(f)$ (more precisely $\exists \Gamma' \in \mathcal{CC}([f \geq t])$ such as $H = \mathrm{Sat}(\Gamma')$). Figure 2d shows the tree of shapes computed from the image from Figure 2a. This tree is *self-dual*: the tree of shapes of f is also the tree of shapes of $\complement f$.

B. Building and Using Morphological Trees

Any of the previous trees is an exact representation of the image it is computed from. Hence the following fundamental concept: these trees are actually a way to represent an image through its contents. Put differently, an image can be encoded as a tree through the components obtained by successive thresholding operations. As a consequence, reconstructing an image from any morphological tree is straightforward. Another important idea is that those representations are "rich", for they are *structured* and *fine*. Indeed node parenthood maps

```
FIND-ROOT(x)
 1  if zpar(x) = x  then return  x
 2     else { zpar(x) ← FIND-ROOT(zpar(x)) ; return zpar(x) }

COMPUTE-TREE(f)
 1  for each p, zpar(p) ← undef
 2  R ← REVERSE-SORT(f)    // maps R into an array
 3  for  each p ∈ R in direct order
 4     parent(p) ← p ; zpar(p) ← p
 5     for  each n ∈ N(p) such as zpar(n) ≠ undef
 6        r ← FIND-ROOT(n)
 7        if  r ≠ p  then { parent(r) ← p ; zpar(r) ← p }
 8  DEALLOCATE(zpar)
 9  return pair(R, parent)    // a ''parent'' function

CANONIZE-TREE(parent, f)
 1  for  each p ∈ R in reverse order
 2     q ← parent(p)
 3     if  f(parent(q)) = f(q)  then  parent(p) ← parent(q)
 4  return parent    // a ''canonized'' parent function
```

Fig. 3: Union-find-based max-tree computation algorithm [14].

components inclusion and there are about as many tree nodes as there are image pixels.

The min- and max-trees are easy to compute and can be efficiently implemented with only a few lines of code using various algorithms [14], [15]. Figure 3 is an example of such an algorithm, computing an image's max-tree thanks to a union-find strategy [16]. This algorithm relies on a total ordering relation $\mathcal{R}$ between the pixels of the input image f, based on decreasing gray levels, and for pixels having the same level, on the classical video scan order (see an example in Figure 4). The routine COMPUTE-TREE produces a max-tree (see the left-hand side of Figure 5). This is a rooted tree: every node points to its parent, except the root node (J), pointing to itself. The routine CANONIZE-TREE is an optimization applied to the tree produced by COMPUTE-TREE, generating a canonized and faster tree (see the right-hand side of Figure 5). Figure 6 shows the encoding of these trees as "parent" images.

The tree of shapes can also be obtained fairly easily thanks to a recently proposed algorithm [17]. All those trees are computable with a quasi-linear time complexity, when (gray-levels) pixels values have a low quantization (typically when represented by 12-bit values or less). Finally, let us note that encoding an image by a tree is very compact in memory: the "parenthood" relation between pixels can be represented as an image having the same size as the input image (see Figure 6).

III. Connected Operators as Tree Filtering

Let us consider an attribute function $\mathcal{A}$, defined on connected components and which is increasing (that is, $\forall(\Gamma,\Gamma'), \Gamma \subset \Gamma' \Rightarrow \mathcal{A}(\Gamma) < \mathcal{A}(\Gamma')$). Given any morphological tree $\mathfrak{T}$, an operator filters out a component $\Gamma \in \mathfrak{T}$ if it verifies the criterion $\mathcal{A}(\Gamma) < \lambda$, where λ is the "strength" of the filter, acting as a threshold. Since the attribute function is increasing, this filtering operation is a pruning of the (leaves of the) tree. Figure 7 sums up the workflow of this tree-based filtering approach. As a consequence, it just removes some connected components of the input image and the other components are preserved. The output image thus contains some of the original

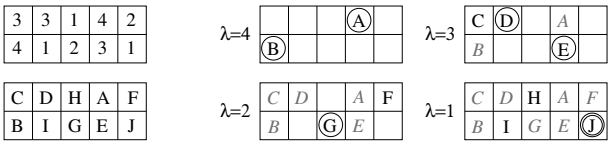


Fig. 4: Algorithm from Figure 3 applied on a sample image. Left: input image (top) and the ordering $\mathcal{R}$ between pixels (bottom). Right: level sets for different values of λ; canonical points (representative of connected components) are circled.

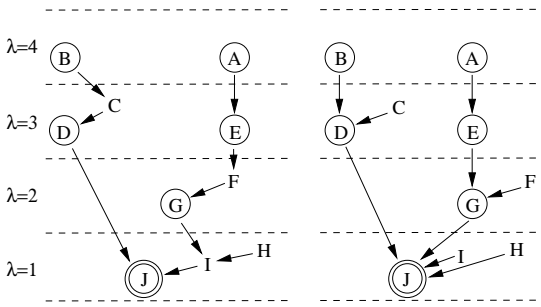


Fig. 5: Max tree (left) computed from the input image of Figure 4 and its canonical form (right). The parenthood between points is denoted by arrows; for instance, $parent(C) = D$.

C	D	H	A	F
B	I	G	E	J

D	J	I	E	G
C	J	I	F	J

D	J	J	E	G
D	J	J	G	J

Fig. 6: Original sample image (left); parent image (center) and canonized parent image (right) corresponding to the trees of Figure 5.

contours of the input—such connected operators do not shift nor create contours. It is important to note that that it is *not* the behavior of most of the classical "well-known" morphological operators, which intentionally modify the shapes of the objects.

Depending on the considered tree, filters have different behaviors. Using a min-tree, the suppressed components are those that are darker than their surrounding areas in the image; such an operator is an algebraic *closing*. With a max-tree, we

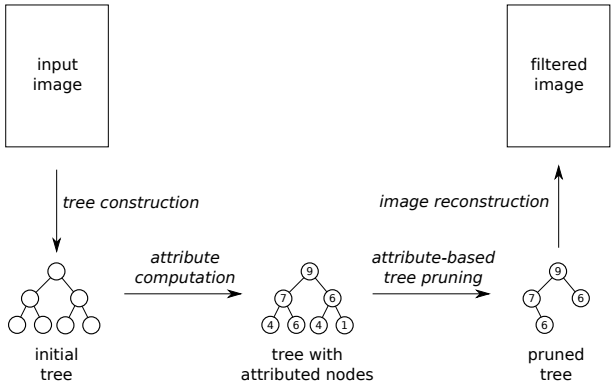


Fig. 7: Workflow of a tree-based strategy to implementing connected filters (with parameter λ set to 5).

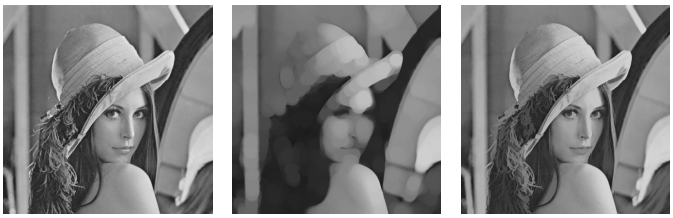


Fig. 8: Comparison of an opening based on a structuring element and an algebraic opening: initial image (left); opening using a disc with a radius of 15 pixels as structural element (center); algebraic opening using an area of approximately $\pi 15^2$ pixels (right), removing any component smaller than this area (especially noise). Although both filters alter the same "surface", the latter does not move nor create contours, whereas the former does.

have the dual behavior with light components being filtered out and we obtain an algebraic *opening*. Finally, filtering a tree of shapes exhibits both behaviors at the same time, realizing a *grain* filter.

Connected operators are interesting for many reasons:

1) In contrast with many classical filters, they preserve contours so they are able to simplify images without being "destructive" for non-filtered data (Figure 8 illustrates this fact).
2) Their mathematical properties are sound and strong [3], [4].
3) They take into account all connected components of the input image, which is the opposite attitude of the classical scheme of document image processing (given in Figure 1) where an early binarization step gets rid of most image components.
4) They are really intuitive to use.
5) Lastly, since connected operators are able to select some components while filtering out others, they are very close to segmentation and object recognition tools.

IV. Connected Filters Applied

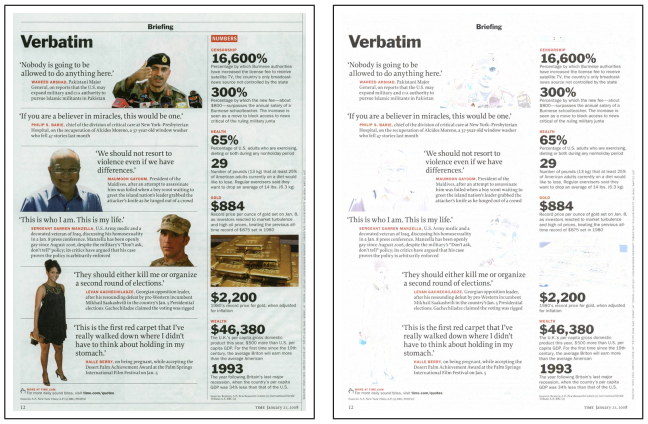
Figure 9 depicts some results of connected operators filtering[1]. In order to take into account all possible input information, we have run these operators on color images. For this purpose, we have used a simple *marginal* approach— even if there are more elaborated methods to apply mathematical morphology methods to color images [4, Chapter 11]. We have applied a "gray-level" filter ϕ to every channel (red, green, blue) of the input image independently of each other and recomposed the output color image from the filtered channel images: $\phi^{\text{color}}(f) = (\phi^{\text{gray}}(f_r), \phi^{\text{gray}}(f_g), \phi^{\text{gray}}(f_b))$. Changing the attribute function $\mathcal{A}$, computed from connected components, enables the design of filters that target different kinds of objects in document images.

On the first example, shown in Figure 9a, a component Γ is removed when its bounding box is too small.

[1]Full-size versions of images shown in this section are available online, together with additional material [18].

(a) Filtering out everything but boxes.

(b) Showing filtered lines.



(c) An image featuring almost only text.

Fig. 9: Sample uses of connected operators. Left: input images; right: filtered images (results).

Specifically, we have filtered the input using the attribute $\mathcal{A}(\Gamma) = (\text{width}(\Gamma), \text{height}(\Gamma))$ so that components Γ for which $\mathcal{A}(\Gamma) <_{\text{or}} (\lambda, \lambda)$ are filtered out, where the $<_{\text{or}}$ relation is defined by $(a, b) <_{\text{or}} (c, d) \Leftrightarrow a < c$ or $b < d$. The only remaining components are those whose bounding box is larger than λ width- and height-wise. It is now admittedly simpler, from the resulting image, to search for pictures and boxes contained in the document. Moreover, the shape of these objects has been preserved, thanks to the very nature

of connected filters.

The second example, from Figure 9b, uses a technique almost similar to the first one. However, instead of showing the filtered image, we show the *difference* between the result of the filter and the input image. This operation is called a morphological *top-hat*. What appears on the output image is therefore what has been removed by the connected filter; that is, separator lines. We can observe that even very thin and poorly contrasted objects are retrieved by such a filter. This feature is a consequence of the fact that morphological operators are invariant by contrast change, meaning that the ordering of values matters more than the values themselves.

The third example, illustrated by Figure 9c, is also a top-hat. Here, $\mathcal{A}(\Gamma) = \text{area}(\Gamma)$, that is, the chosen attribute is the component area, which amounts to the very simple operation of counting the number of pixels belonging to a component. The resulting image thus contains very small components of noise from the input image, some components belonging to pictures, and textual contents of the document. Yet text components are more contrasted than other components. Should a binarization be applied to retrieve text (following the classical approach shown in Figure 1), we argue that using this image as input, in lieu of the original input image, would produce better binarization results. Note that an interesting property of an algebraic connected filter where $\mathcal{A}(\Gamma) = \text{area}(\Gamma)$ and where λ is the (area) threshold used in the filter's criterion, is that this filter is equivalent to merging the results of all filters based on a structuring element B with $\text{area}(B) < \lambda$.

V. Conclusion

This paper presents connected filters, their implementation using morphological trees and some examples of their use in the domain of document image analysis. We have highlighted the benefits of morphological connected operators and presented their underlying tree-based representations. We have advocated a strategy preserving the whole components of a document image, arranged in a tree structure, instead of restricting subsequent processing steps to a only handful of components produced by an early binarization. We provide full-size results of our experiments online to demonstrate the effectiveness and the robustness of the connected filters approach [18].

Our future work includes a morphological tree-based object recognition strategy to extract text from natural images. Indeed having a simplified representation allows us to limit the number of text false positives, while self-dual filters based on the tree of shapes are able to deal with text in reverse video.

We advocate the idea of *reproducible research* [19], [20]. Therefore, the tools presented in this paper will be available in the next release of the Scribo module [21], as part of the Olena image processing platform [22], [23], along with an online demonstrator.

References

[1] J. Serra, *Image Analysis and Mathematical Morphology*. London: Academic Press, 1982, vol. 1.

[2] ——, *Image Analysis and Mathematical Morphology*. London: Academic Press, 1988, vol. 2: Theoretical Advances.

[3] P. Soille, *Morphological Image Analysis: Principles and Applications*, 2nd ed. Springer, 2004.

[4] L. Najman and H. Talbot, Eds., *Mathematical Morphology—From Theory to Applications*. ISTE & Wiley, 2010.

[5] P. Maragos, Ed., *Mathematical Morphology and its Applications to Image and Signal Processing – Proceedings of the 3rd International Symposium on Mathematical Morphology (ISMM)*. Kluwer, 1996.

[6] L. O'Gorman and R. Kasturi, *Document Image Analysis*. Los Alamitos, CA: IEEE Computer Society Press, 1997.

[7] J. Crespo, J. Serra, and R. Schafer, "Theoretical aspects of morphological filters by reconstruction," *Signal Processing*, vol. 47, no. 2, pp. 201–225, 1995.

[8] P. Salembier and J. Serra, "Flat zones filtering, connected operators, and filters by reconstruction," *IEEE Transactions on Image Processing*, vol. 4, no. 8, pp. 1153–1160, 1995.

[9] L. Vincent, "Morphological grayscale reconstruction in image analysis: Efficient algorithms and applications," *IEEE Transactions on Image Processing*, vol. 2, no. 2, pp. 176–201, 1993.

[10] B. Naegel and L. Wendling, "Document binarization based on connected operators," in *Proceedings of International Conference on Document Analysis and Recognition (ICDAR)*, 2009, pp. 316–320.

[11] M. H. Wilkinson and J. Oosterbroek, "Mask-edge connectivity: Theory, computation, and application to historical document analysis," in *Proceedings of International Conference on Pattern Recognition (ICPR)*, 2012, pp. 1334–1337.

[12] P. Salembier and M. Wilkinson, "Connected operators," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 136–157, Nov. 2009.

[13] P. Monasse and F. Guichard, "Fast computation of a contrast invariant image representation," *IEEE Transactions on Image Processing*, vol. 9, no. 5, pp. 860–872, May 2000.

[14] C. Berger, T. Géraud, R. Levillain, N. Widynski, A. Baillard, and E. Bertin, "Effective component tree computation with application to pattern recognition in astronomical imaging," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 4, 2007, pp. 41–44.

[15] E. Carlinet and T. Géraud, "A comparison of many max-tree computation algorithms," in *Mathematical Morphology and Its Application to Signal and Image Processing – Proceedings of the 11th International Symposium on Mathematical Morphology (ISMM)*, ser. Lecture Notes in Computer Science Series, C. L. Hendriks, G. Borgefors, and R. Strand, Eds., vol. 7883. Heidelberg: Springer, 2013, pp. 37–48.

[16] R. E. Tarjan, "Efficiency of a good but not linear set union algorithm," vol. 22, no. 2, pp. 215–225, 1975.

[17] T. Géraud, E. Carlinet, S. Crozet, and L. Najman, "A quasi-linear algorithm to compute the tree of shapes of n-D images." in *Mathematical Morphology and Its Application to Signal and Image Processing – Proceedings of the 11th International Symposium on Mathematical Morphology (ISMM)*, ser. Lecture Notes in Computer Science Series, C. L. Hendriks, G. Borgefors, and R. Strand, Eds., vol. 7883. Heidelberg: Springer, 2013, pp. 98–110.

[18] "Planting, growing, and pruning trees: Connected filters applied to document image analysis," http://publis.lrde.epita.fr/201404-DAS-Ressources, additional resources related to this paper.

[19] J. B. Buckheit and D. L. Donoho, "WaveLab and reproducible research," Stanford University, Stanford CA 94305, USA, Tech. Rep. 474, 1995.

[20] S. Fomel and J. F. Claerbout, "Guest editors' introduction: Reproducible research," *Computing in Science and Engineering*, vol. 11, no. 1, pp. 5–7, 2009.

[21] G. Lazzara, R. Levillain, T. Géraud, Y. Jacquelet, J. Marquegnies, and A. Crépin-Leblond, "The SCRIBO module of the Olena platform: a free software framework for document image analysis," in *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 2011.

[22] R. Levillain, T. Géraud, and L. Najman, "Why and how to design a generic and efficient image processing framework: The case of the Milena library," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2010, pp. 1941–1944.

[23] EPITA Research and Developpement Laboratory (LRDE), "The Olena image processing platform," http://olena.lrde.epita.fr.