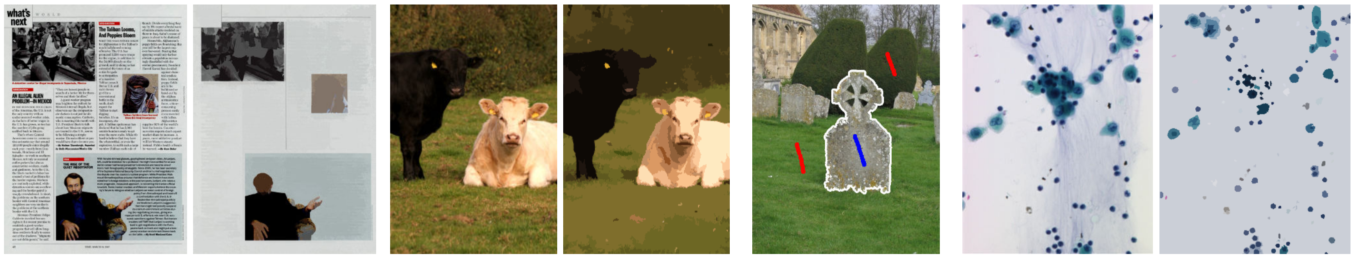
THE TREE OF SHAPES TURNED INTO A MAX-TREE: A SIMPLE AND EFFICIENT LINEAR ALGORITHM

Edwin Carlinet, Sébastien Crozet, Thierry Géraud

EPITA Research and Development Laboratory (LRDE),
14-16 rue Voltaire, FR-94270 Le Kremlin-Bicêtre, France.



(a) Grain filtering (b) Image simplification (c) Object picking (d) Shape-based filtering

Fig. 1. Some applications featuring the Tree of Shapes (ToS). (a) Grain filters are used for layout extraction by removing dark or bright text [1]. (b) Energy-based node selection for image simplification or segmentation [2, 3, 4]. (c) Interactive segmentation using few scribbles on the ToS [5]. (d) Shape-based attribute filtering for cytology, depending on a circularity criterion [6].

ABSTRACT

The Tree of Shapes (ToS) is a morphological, tree-based representation of an image, translating the inclusion of its level lines. It features many invariants to image changes, which make it well-suited for many applications in image processing and pattern recognition. In this paper, we propose a way of turning a ToS computation into a Max-Tree computation. The latter has been widely studied, and many efficient algorithms (including parallel ones) have been developed. Furthermore, we develop a specific optimization to speed-up the common 2D case. It follows a *simple* and *efficient* algorithm, running in *linear* time with a low memory footprint, that outperforms other currently used algorithms. For Reproducible Research purpose, we distribute our code as free software.

Index Terms— Tree of Shapes, Algorithm, Mathematical morphology

1. INTRODUCTION

The Tree of Shapes (ToS) is a hierarchical structure encoding the inclusion of the level lines of an image. Those level lines are the contours of shapes (see Fig. 2). The ToS is an interesting structure because level lines are invariant to contrast changes and to the inversion of contrast. These properties are particularly valuable in image processing, pattern recognition and computer vision [7], where the ToS has shown its robustness to deal with poorly contrasted images, with changes of illuminations and of scene view points [8, 9] (see Fig. 3). The strength of this representation lies in its versatility. Like min- and max-trees, it enables performing non-trivial advanced connected filtering [10, 11, 12, 6] (Figs. 1a and 1d) that are easy to tune for application specific tasks. Morphological attribute profiles over the ToS have been used in remote sensing for classification of hyperspectral images [13, 14, 15, 16]. In [17], the ToS is pruned to detect and track speckles in ultrasound images. Concerning natural images, the ToS is used as a feature detector to select stable regions in [18] and to perform some texture analysis in [19] avoiding managing two component trees as in the original MSER algorithm. In [20], it is used for object picking (Fig. 1c) and alpha-matting to compute shortest path distances free of any topological issues. By supporting advanced node selection strategies, some au-

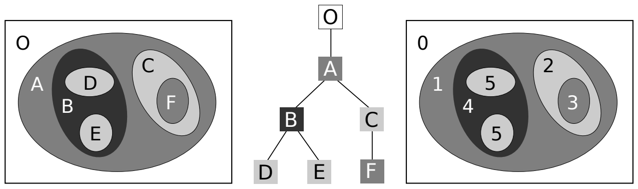
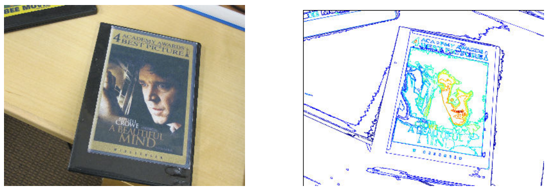
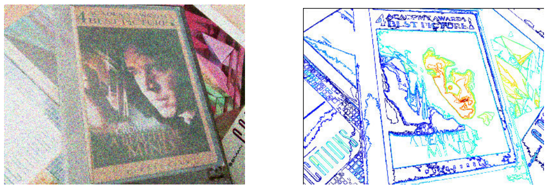


Fig. 2. An image (left), its ToS (middle), and an inclusion order map Ord (right).



(a) An image (left) and some meaningful level lines (right).



(b) A different view of the film cover contained in (a). The level lines of this new image globally match the ones of (a), thus the ToS properly encodes the contents of both images.

Fig. 3. Robustness of the level lines w.r.t. transforms in the value and domain spaces. The lines are colorized w.r.t. their depth of inclusion in the ToS.

thors [21, 3, 4] have developed energy-based node filtering for scene simplification and segmentation (see Fig. 1b). The ToS is not limited to gray-level images as it has been been extended to color and multivariate images with the MToS [22]. Lastly, some discrete topology considerations about the ToS are discussed in [23, 24], and its relationship with a very popular visual saliency measure are described in [25].

Nevertheless, the ToS is still under-exploited outside the Mathematical Morphology community. As its potential is not yet to demonstrate, this may to be due to the lack of a *fast* and *easy-to-write* algorithm. Nowadays, fast algoritms are of prime importance in embedded real-time systems. Currently, there are four ToS algorithms. The first approach, the Fast Level Line Transform [8] (FLLT), computes and merges the min- and max-trees. The second, the Fast Level Set Transform [26] (FLST), relying on a region-growing approach, extracts each branch of the tree starting from a leaf (regional extremum without holes) and expands it up to the root until at least one bifurcation is encountered, that is, until the region is no longer a regional extremum. The third algorithm, from Song [27] takes a top-down approach by tracking the level lines starting from the border. The algorithm is restricted to 2D images with hexagonal pixels, or emulates 6-connectivity on images with a square grid.

In this article, we propose a simpler and more efficient version of Géraud et al. [28]' algorithm. Our main contribution is to adapt their algorithm in order to turn a ToS computation problem into a Max-tree computation problem. The transformation is explained in Section 2. Consequently, it is simpler and efficient because there exist many well-known max-tree algorithms achieving good performances. The second contribution, explained in Section 3, is a simple 2D optimization that reduces the number of pixels to process and makes our algorithm run faster. In Section 4 we compare the efficiency of our approach with others.

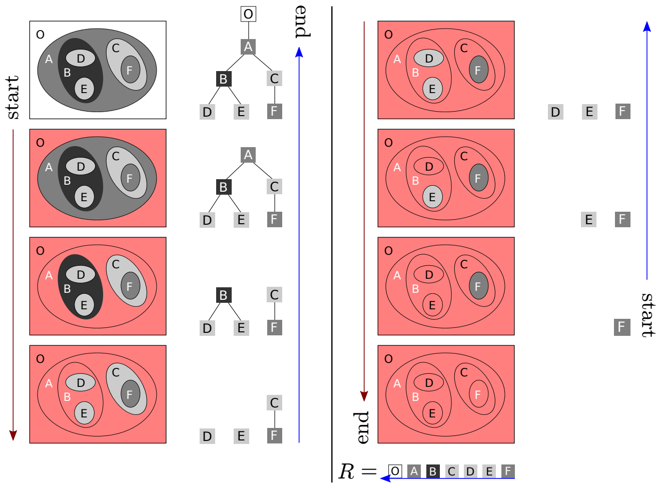


Fig. 4. Run of the quasi-linear algorithm [24].

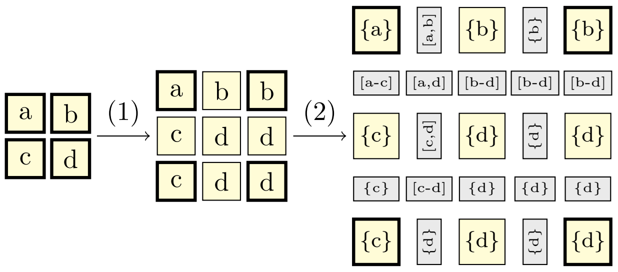


Fig. 5. Interpolation (1) and immersion (2) of an image. We suppose $a \leq b \leq c \leq d$ for the example. Original pixels have a thick border, 2-faces are in yellow and intermediate pixels (0/1-faces) are in gray.

2. A LINEAR TOS COMPUTATION ALGORITHM

2.1. The quasi-linear algorithm

Géraud et al. [28] describe a Union-Find based algorithm to compute the Tree of Shapes in quasi-linear time in four steps:

Interpolation. The image is rescaled by a factor 2, using min/max/median interpolation to get a well-composed image. This step ensures the uniqueness of the ToS and allows choosing from custom background/object connectivities. For example, the max-interpolation stands for the upper semi-continuity interpretation of the image; hence lower level sets are 4-connected and upper level sets are 8-connected. The median can be used to get a *real* self-duality [24].

Immersion. The image is transformed as an interval-valued map F in the Khalimsky grid. Intermediate pixels (namely 0- and 1-faces) are used to represent all level lines passing between any two original pixels (2-faces). Interpolation and immersion are illustrated in Fig. 5.

Pixel sorting. A continuous propagation is performed on F starting from the border (any point can actually be used to start the propagation and thus is used for rooting the tree); follow the red arrows in Fig. 4. The processsing order of the pixels is recorded in a vector R, as well as the level they have been

```
function ComputeOrderMap($f$, $F$, $p_\infty$)
    Q $\leftarrow \emptyset$; R $\leftarrow \emptyset$
    Ord($x$) $\leftarrow -1$ forall $x$
    $\lambda^{\text{old}} \leftarrow f(p_\infty)$
    INSERT(Q, $(\lambda^{\text{old}}, p_\infty)$)
    $d \leftarrow 0$
    while Q $\neq \emptyset$ do
        $(\lambda, p) \leftarrow$ POP(Q, $\lambda^{\text{old}}$)
        if $\lambda^{\text{old}} \neq \lambda$ then          $d \leftarrow d + 1$
        Ord($p$) $\leftarrow d$
        Push($R$, $p$)
        for all $n \in \mathcal{N}_4(p)$ such that Ord($n$) $= -1$ do
            $[a, b] \leftarrow F(n)$
            if $\lambda < a$ then          INSERT(Q, $(a, n)$)
            else if $\lambda > b$ then     INSERT(Q, $(b, n)$)
            else                           INSERT(Q, $(\lambda, n)$)
        $\lambda^{\text{old}} \leftarrow \lambda$
    return $(R, Ord)$
```

Fig. 6. Sort procedure.

visited with.

Tree construction. The pixels are processed in the reverse order of R and Union-Find is used to track disjoint sets of connected components to build the tree in a bottom-up fashion; follow the blue arrows in Fig. 4. This is the classic algorithm to build the component tree with the Union-Find approach.

2.2. Order map image transformation

In [29] and [22], the authors noticed an obvious but interesting feature of component trees. It is possible to recover the component tree from an image whose pixels are valued by the depth of their node; one just has to compute the max-tree of this *depth* image. Actually, not only the *depth*, but any topological ordering of the tree would make suffice.

This property is the key foundation for our approach as many efficient algorithms have been proposed for Max-tree constructions and the optimization task is delegated to the previous steps. In particular, the *pixel sorting* is now responsible for computing a map Ord (in place of K) corresponding to a topological sort of the ToS. The algorithm is given in Fig. 6, and a resulting order map is illustrated in Fig. 2 (right). The algorithm processes the pixels level-wise with a propagation front. The latter has to keep the pixels sorted by their level through the ordered associative map Q so that we can access pixels at any level from the front. Each time the current level is changed, the number of processed level d is incremented and when a pixel p is visited, we store its processing order in Ord(p) and push it back into R.

In Fig. 6, we assume the Abstract Data Type for Q:
INSERT(Q, (λ, p)): add the value (pixel) p at key (level) λ.
POP(Q, k): retrieve and remove the lowest pair (λ, p) such that $k \leq \lambda$ or greatest pair (λ, p) such that $\lambda < k$. This is typically implemented with hierarchical queues for low quantized images (and red-black-trees for high-quantized images).

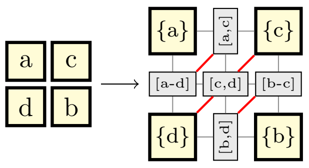


Fig. 7. Immersion for the optimized version in case of a saddle-point configuration.

2.3. The ToS turned into a Max-tree

Once the order map has been computed, one has to build its Max-tree to get the ToS. In most cases, the maximum value is such that we can apply a linear Max-tree algorithm such as in [30]. Otherwise, we fall back to a quasi-linear Union-find based Max-tree algorithm [31] (avoiding the sorting step since R has already been computed). The reader is referred to [32] for a comprehensive survey.

3. 2D OPTIMIZATION

The *interpolation* followed by the *immersion* implies a prohibitive time and memory overhead, since the number of elements is multiplied by 16. Hopefully, this representation simulating 4/8-connectivity, can be optimized for 2D images by avoiding the *interpolation* step. A connectivity map replaces the inter-pixels. To get the same results as the algorithm running with the original representation, we must ensure that the optimized representation has the same level sets as the original one. Hence, we use a specific *immersion* which depends on the local configuration of the faces.

For 1- or 2-faces, the rules remain the same. However, they differ for a 0-face p if the four neighboring pixels a, b, c, d form a saddle point. Without loss of generality, we suppose $a \leq b < c \leq d$, then $F(p) = [c, d]$. It simulates upper-semi-continuity by preventing level lines lower than c from passing through the saddle point using the 4-connectivity.

Using a 4-connectivity together with this new interpolation is too restrictive. Indeed, the 2-faces valued by c and d on a saddle point might not be able to connect properly if they are reached during the propagation of increasing levels of grey. Therefore in addition to the 4-connectivity, a connectivity map is used to add diagonal connections depending on the local configuration of 2-faces as shown in Fig. 7. The role of this map is to act as if we were using the interpolated inter-pixels. Thus it has be designed so that the lower and upper level-sets are the same in both cases. The equivalence of both representations is shown in Fig. 8. This map is used afterward as the connectivity graph of pixels during the *ordered map* and *max-tree* computation.

4. PERFORMANCE ANALYSIS

4.1. Complexity and memory analysis

Let n be the number of pixels *after* immersion. This means if k is the number of pixels of f, $n = 4.k$.

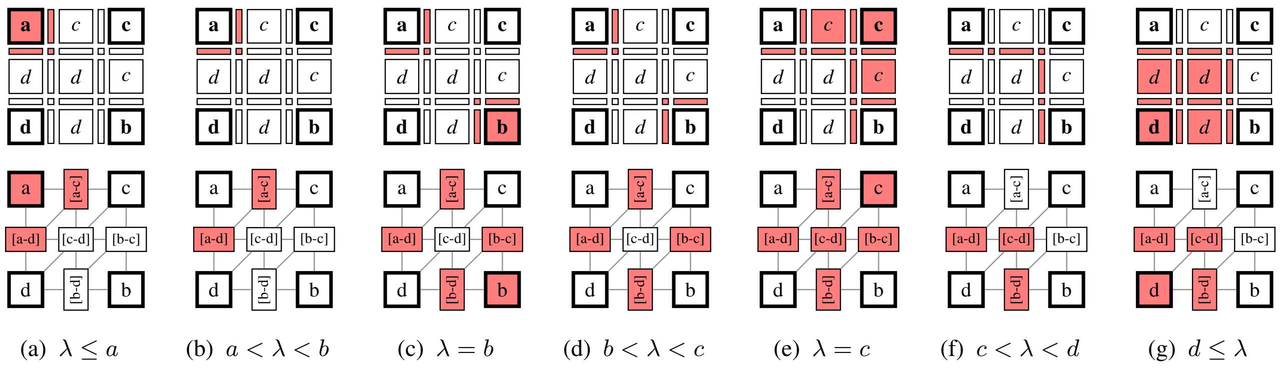


Fig. 8. Equivalence between the optimized and naive immersion with a saddle point configuration for every possible level λ. For each level, the selected set and its complement contain the same connected components (in term of original faces).

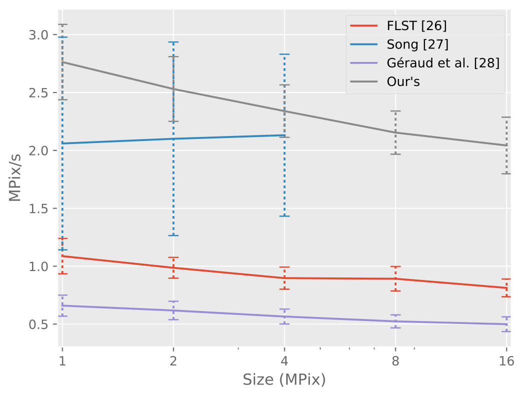


Fig. 9. Comparaison of the speed (in *pixels processed by sec.*, the higher the better) of ToS algorithms w.r.t. the image size.

For low-quantized images, the *order map* computation is linear using hierarchical queues, and the max-tree computation has a worst-case quasi-linear complexity. In practice, for natural images the maximum is such that we can apply a linear max-tree algorithm. Hence, the whole process is linear on average (and quasi-linear at worse).

For high-quantized images, the *order map* computation is $n \log n$ because storing the pixels sorted in the front typically uses red-black-trees. The Max-tree computation remains unchanged. The whole process is thus $n \log n$.

Memory-wise, let I be the size of `int` in bytes. The computation of the *order map* requires $4 \cdot n \cdot I$ or $5 \cdot n \cdot I$ bytes for $\mathrm{Q}, \mathrm{R}, \mathrm{Ord}$ (depending on the implementation) and the connectivity map. The memory used for the Max-tree construction depends on the chosen algorithm (see [32]) but requires between $2 \cdot n \cdot I$ and $3 \cdot n \cdot I$ bytes.

4.2. Comparaison with existing algorithms

We compare our algorithm with Megawave's FLST, Song [27]'s (implementation provided by the author) and Géraud et al. [28]'s algorithms. The outputs slightly differ between the methods (either because of the connectivity or the tree rooting strategy) but they remain similar. We tested every algorithm on a natural image dataset (20-MPix images) cropped to a given size varying from 1 to 16 MPix, the minimum of 5 runs has been kept. These benchmarks were run on an Intel core i7 7500U, 2.7GHz and 8GB of RAM. Figure 9 shows the average speed and the standard deviation on the dataset.

A first observation is that we were not able to run Song [27]'s method on images larger than 4 MPix because of a too large memory consumption. Next, all algorithms have, in practice, a linear behavior w.r.t the size, even if some of them have a quadratic worst-case complexity. However, the speed of the FLST, Géraud et al. [28] and our algorithm depends on the image size and decreases by 5-10% when the size doubles.

Overall, our algorithm is 4x faster than Géraud et al. [28], and 2.5x faster than the FLST. It is also more stable than Song [27] w.r.t the image content that shows highly varying processing times. On the average, our algorithm is also faster but the difference in speed tends to lower as the image size increases.

5. CONCLUSION & PERSPECTIVE

We have introduced a new algorithm[1] to compute the ToS with a practical linear complexity. The main idea was to turn the ToS computation into a *max-tree* computation to benefit from efficient component-tree implementations. A specific 2D optimization has been developed to reduce the memory footprint and improve processing speed by avoiding the need for inter-pixels interpolation. Our proposal outperforms the current state-of-the-art implementations. The parallelization of our algorithm has not been done as a matter of fairness with the other sequential implementations. Yet, the parallelization strategy for the *immersion* described in [29] can be applied and many efficient parallel max-tree algorithms have been proposed for both low and high-quantized images [32, 33]. Our algorithm can be generalized in nD (see [28] and [34]) but the 2D memory optimization no longer holds and requires multiplying the image size by 4^n which may become intractable for large images. For such a case, the only viable alternative remains the FLLT [35]. As a consequence, as future work, we will try to extend our memory optimization to nD grids.

[1]Source code at `http://publications.lrde.epita.fr/carlinet.18.icip`

References

[1] G. Lazzara, T. Géraud, and R. Levillain, "Planting, growing and pruning trees: Connected filters applied to document image analysis," in *Proc. of IAPR DAS*, 2014, pp. 36–40.

[2] Y. Xu, T. Géraud, and L. Najman, "Salient level lines selection using the Mumford-Shah functional," in *Proc. of IEEE Intl. Conf. on Image Processing (ICIP)*, 2013, pp. 1227–1231.

[3] ——, "Hierarchical image simplification and segmentation based on Mumford-Shah-salient level line selection," *Pattern Recognition Letters*, vol. 83, no. 3, pp. 278–286, Nov. 2016.

[4] Y. Xu, E. Carlinet, T. Géraud, and L. Najman, "Hierarchical segmentation using tree-based shape spaces," *IEEE Trans. on PAMI*, vol. 39, no. 3, pp. 457–469, Apr. 2017.

[5] E. Carlinet and T. Géraud, "Morphological object picking based on the color tree of shapes," in *Proc. of IPTA*, 2015, pp. 125–130.

[6] Y. Xu, T. Géraud, and L. Najman, "Connected filtering on tree-based shape-spaces," *IEEE Trans. on Pattern Analysis and Machine Intel.*, vol. 38, no. 6, pp. 1126–1140, Jun. 2016.

[7] V. Caselles, B. Coll, and J. Morel, "Topographic maps and local contrast changes in natural images," *International Journal of Computer Vision*, vol. 33, no. 1, pp. 5–27, 1999.

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

[9] F. Cao, J.-L. Lisani, J.-M. Morel, P. Musé, and F. Sur, *A Theory of Shape Identification*, ser. LNM. Springer, 2008, vol. 1948.

[10] R. Jones, "Connected filtering and segmentation using component trees," *CVIU*, vol. 75, no. 3, pp. 215–228, Sep. 1999.

[11] E. Urbach and M. Wilkinson, "Shape-only granulometries and grey-scale shape filters," in *Proc. of the Intl. Symp. on Mathematical Morphology (ISMM)*. CSIRO, 2002, pp. 305–314.

[12] G. Ouzounis and M. Wilkinson, "Mask-based second-generation connectivity and attribute filters," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 990–1004, 2007.

[13] G. Cavallaro, M. Dalla Mura, J. A. Benediktsson, and A. Plaza, "Remote sensing image classification using attribute filters defined over the tree of shapes," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 54, no. 7, pp. 3899–3911, July 2016.

[14] G. Cavallaro *et al.*, "Region-based classification of remote sensing images with the morphological tree of shapes," in *Proc. of IEEE IGARSS*, 2016, pp. 5087–5090.

[15] G. Cavallaro, N. Falco, M. Dalla Mura, and J. A. Benediktsson, "Automatic attribute profiles," *IEEE Trans. on Image Processing*, vol. 26, no. 4, pp. 1859–1872, Apr. 2017.

[16] L. Gueguen and R. Hamid, "Toward a generalizable image representation for large-scale change detection: Application to generic damage analysis," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 54, no. 6, pp. 3378–3387, June 2016.

[17] N. Widynski, T. Géraud, and D. Garcia, "Speckle spot detection in ultrasound images: Application to speckle reduction and speckle tracking," in *Proc. of IEEE IUS*, 2014, pp. 1734–1737.

[18] P. Bosilj, E. Kijak, and S. Lefèvre, "Beyond MSER: Maximally stable regions using tree of shapes," in *Proc. of the British Machine Vision Conf. (BMVC)*, 2015, pp. 169.1–169.13.

[19] G. Xia, J. Delon, and Y. Gousseau, "Shape-based invariant texture indexing," *International Journal of Computer Vision*, vol. 88, no. 3, pp. 382–403, 2010.

[20] A. Dubrovina, R. Hershkovitz, and R. Kimmel, "Image editing using level set trees," in *Proc. of IEEE ICIP*, 2014, pp. 4442–4446.

[21] J. Cardelino, G. Randall, M. Bertalmio, and V. Caselles, "Region based segmentation using the tree of shapes," in *Proc. of IEEE ICIP*, 2006, pp. 2421–2424.

[22] E. Carlinet and T. Géraud, "MToS: A tree of shapes for multivariate images," *IEEE Trans. on Image Processing*, vol. 24, no. 12, pp. 5330–5342, Dec. 2015.

[23] N. Boutry, T. Géraud, and L. Najman, "On making nD images well-composed by a self-dual local interpolation," in *Proc. of DGCI*, ser. LNCS, vol. 8668. Springer, 2014, pp. 320–331.

[24] T. Géraud, E. Carlinet, and S. Crozet, "Self-duality and discrete topology: Links between the morphological tree of shapes and well-composed gray-level images," in *Proc. of ISMM*, ser. LNCS, vol. 9082. Springer, 2015, pp. 573–584.

[25] T. Géraud, Y. Xu, E. Carlinet, and N. Boutry, "Introducing the Dahu pseudo-distance," in *Proc. of ISMM*, ser. LNCS, vol. 10225. Springer, 2017, pp. 55–67.

[26] V. Caselles and P. Monasse, *Geometric Description of Images as Topographic Maps*, ser. LNM. Springer, 2009, vol. 1984.

[27] Y. Song, "A topdown algorithm for computation of level line trees," *IEEE Trans. on Image Processing*, vol. 16, no. 8, pp. 2107–2116, Aug. 2007.

[28] T. Géraud, E. Carlinet, S. Crozet, and L. Najman, "A quasi-linear algorithm to compute the tree of shapes of nD images," in *Proc. of ISMM*, ser. LNCS, vol. 7883. Springer, 2013, pp. 98–110.

[29] S. Crozet and T. Géraud, "A first parallel algorithm to compute the morphological tree of shapes of nD images," in *Proc. of IEEE ICIP*, 2014, pp. 2933–2937.

[30] P. Salembier, A. Oliveras, and L. Garrido, "Antiextensive connected operators for image and sequence processing," *IEEE Trans. on Image Processing*, vol. 7, no. 4, pp. 555–570, 1998.

[31] L. Najman and M. Couprie, "Building the component tree in quasi-linear time," *IEEE Trans. on Image Processing*, vol. 15, no. 11, pp. 3531–3539, Nov. 2006.

[32] E. Carlinet and T. Géraud, "A comparative review of component tree computation algorithms," *IEEE Trans. on Image Processing*, vol. 23, no. 9, pp. 3885–3895, Sep. 2014.

[33] M. Götz, G. Cavallaro, T. Géraud, M. Book, and M. Riedel, "Parallel computation of component trees on distributed memory machines," *IEEE Trans. on Parallel and Distributed Systems*, 2018, online preview available.

[34] N. Boutry, T. Géraud, and L. Najman, "How to make nD functions well-composed in a self-dual way," in *Proc. of ISMM*, ser. LNCS, vol. 9082. Springer, 2015, pp. 561–572.

[35] V. Caselles, E. Meinhardt-Llopis, and P. Monasse, "Constructing the tree of shapes of an image by fusion of the trees of connected components of upper and lower level sets," *Positivity*, vol. 12, no. 1, pp. 55–73, 2008.