A FIRST PARALLEL ALGORITHM TO COMPUTE THE MORPHOLOGICAL TREE OF SHAPES OF nD IMAGES

Sébastien Crozet, Thierry Géraud

EPITA Research and Development Laboratory (LRDE)
14–16, rue Voltaire, FR-94276 Le Kremlin-Bicêtre, France
firstname.lastname@lrde.epita.fr

ABSTRACT

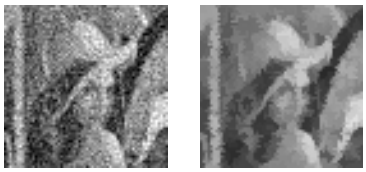
The tree of shapes is a self-dual tree-based image representation belonging to the field of mathematical morphology. This representation is highly interesting since it is invariant to contrast changes and inversion, and allows for numerous and powerful applications. A new algorithm to compute the tree of shapes has been recently presented: it has a quasi-linear complexity; it is the only known algorithm that is also effective for nD images with $n > 2$; yet it is sequential. With the increasing size of data to process, the need of a parallel algorithm to compute that tree is of prime importance; in this paper, we present such an algorithm. We also give some benchmarks that show that the parallel version is computationally effective. As a consequence, that makes possible to process 3D images with some powerful self-dual morphological tools.

***Index Terms*—** Mathematical morphology; Connected operators; Tree of shapes; Algorithms; Parallelization.

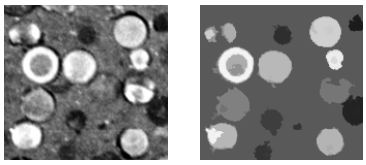
1. INTRODUCTION

Mathematical morphology operators can be divided into two large classes: the most known operators make use of structuring elements, whereas connected operators [1] are based on neighborhood and connected components. The proeminent property of the latter is that they do not shift contours. Many connected filters on gray level images are dual and can be defined from the min-tree and/or the max-tree. This couple of dual trees represent the image and encodes that the connected components obtained by respectively lower and upper thresholds (also called *cuts* [2]) form a tree w.r.t. inclusion. A self-dual tree has been defined in [3], called *tree of shapes*, that describes the image contents in a unique way; such a tree can be understood as the result of merging the dual tree components. The reason why the tree of shapes is interesting is reported by several authors who claim that, in gray level images, object contours coincide with level lines (see, e.g., [4]). That claim sounds very true when observing some applications based on the tree of shapes and depicted in Figure 1.

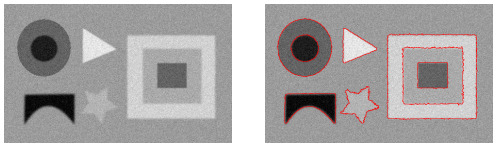
Some new connected operators can be derived from that tree, that make no assumption about the contrast of image components: the inclusion relationship can be due either to light objects surrounded by darker ones, or to the contrary.



(a) Denoising (self-dual grain removal) [5].



(b) Shape Filtering (keep round objects) [6].



(c) Object Detection (energy-based method) [7].

(d) Hierarchical Segmentation (saliency-based) [8].

Fig. 1: Sample uses of the tree of shapes (left column: input images; right column: state-of-the-art results).

As a consequence self-dual operators process the same way light and dark objects. The tree of shapes is a morphological tool with a strong potential, that has not been exploited a lot. Though its applications are numerous: texture indexing [9], object recognition [10], image filtering [5], simplification [11], and segmentation [12, 13, 14]. Some of them are illustrated in Figure 1.

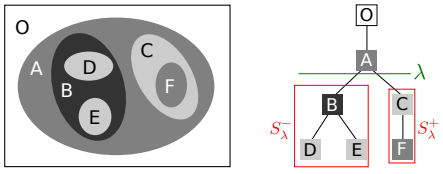
A first quasi-linear algorithm to compute the tree of shapes of nD images has been recently presented [2] (it is recalled in Section 2). Yet this algorithm is sequential. In

this paper we present a first parallel algorithm to compute the tree of shapes (Section 3). In Section 4 we give some numerical results. Last in Section 5 we conclude and discuss some perspectives of our work.

2. THE QUASI-LINEAR ALGORITHM

2.1. Definition of the Tree of Shapes

Let us consider a nD digital image as a function defined on a regular cubical grid, $f : \mathbb{Z}^n \to \mathbb{Z}$. To properly deal with some subsets of $\mathbb{Z}^n$ and with their complementary, we consider the dual connectivities c_{2n} and c_{3^n-1}. For any $\lambda \in \mathbb{Z}$, the respective lower and upper threshold sets of f are defined as $[u < \lambda] = \{x \in X \mid f(x) < \lambda\}$ and $[u \geq \lambda] = \{x \in X \mid f(x) \geq \lambda\}$. From them we deduce two sets, $\mathcal{T}_<(f)$ and $\mathcal{T}_\geq(f)$, composed of the connected components of respectively lower and upper cuts of f: $\mathcal{T}_<(f) = \{\Gamma \in \mathcal{CC}_{c_{2n}}([u < \lambda])\}_\lambda$ and $\mathcal{T}_\geq(f) = \{\Gamma \in \mathcal{CC}_{c_{3^n-1}}([u \geq \lambda])\}_\lambda$, where $\mathcal{CC}$ denotes the operator that gives the set of connected components of a set. The elements of $\mathcal{T}_<(f)$ and $\mathcal{T}_\geq(f)$ respectively give rise to two dual trees: the min-tree and the max-tree of f. We can then define two other sets, $\mathcal{S}_<(f)$ (set of lower shapes) and $\mathcal{S}_\geq(f)$ (set of upper shape), as the sets of components of resp. $\mathcal{T}_<(f)$ and $\mathcal{T}_\geq(f)$ after having filled the cavities of those components. With the cavity-filling (or saturation) operator denoted by Sat, we have: $\mathcal{S}_<(f) = \{\mathrm{Sat}_{c_{3^n-1}}(\Gamma);\ \Gamma \in \mathcal{T}_<(f)\}$ and $\mathcal{S}_\geq(f) = \{\mathrm{Sat}_{c_{2n}}(\Gamma);\ \Gamma \in \mathcal{T}_\geq(f)\}$. The set of all shapes $\mathfrak{S}(f) = \mathcal{S}_<(f) \cup \mathcal{S}_\geq(f)$ forms a tree, the so-called *tree of shapes* of f [3]. Indeed, for any pair of shapes X and X' in $\mathfrak{S}$, we have $X \subset X'$ or $X' \subset X$ or $X \cap X' = \emptyset$. Figure 2 depicts a simple image and its tree of shapes.



(a) Image (b) Tree of shapes

Fig. 2: An image (a) and its tree of shapes (b). The propagation of the level line λ ended, meaning that the nodes O and A have already been visited. The hierarchical queue contains the interior contour of B and C. Thus it can be partitioned in two sets $S^+_\lambda = \partial B$ and $S^-_\lambda = \partial C$. The propagation can proceed on both parts in parallel.

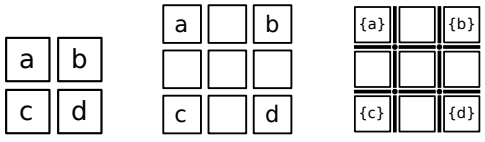
2.2. Sequential Algorithm

A recent paper [2] describes an "union-find"-based algorithm to compute the tree of shapes in quasi-linear time, inspired by the max-tree algorithm given in [15]. (Just note that it proves again the versatility of the union-find algorithm in the context of mathematical morphology [16].) The tree of shapes algorithm is composed of four steps, as depicted by Figure 3.

function COMPUTETREE(f, p_∞)
  $\mathcal{F} \leftarrow$ IMMERSE(f)
  $(\mathcal{R}, \mathcal{F}^\flat) \leftarrow$ SORT($\mathcal{F}$, p_∞)
  $par \leftarrow$ UNIONFIND(reverse($\mathcal{R}$))
  return CANONICALIZE(par, $\mathcal{R}$, $\mathcal{F}^\flat$)

Fig. 3: Algorithmic scheme of the quasi-linear algorithm.

The IMMERSE function adds informations in-between pixels of the input image f. First, f is subdivided (see Figure 4b), multiplying its size by 4^n (so by 16 in 2D). Then, the subdivided image is immersed into the Khalimsky grid [17]. The Khalimsky grid is a decomposition of the image domain into elements of different dimensions: a pixel of a 2D image can be decomposed into four points (called 0-faces), four edges (called 1-face) and one square (the pixel interior, called 2-face) (see Figure 4c). We call the immersed image $\mathcal{F}$. $\mathcal{F}$ is a set-valued map in order to be the upper semi-continue version of f. 2-faces created by the subdivision process are valued by the maximum of their 8-neighboring faces while 1-faces and 0-faces created by the immersion are valued by the span of their neighboring 2-faces. Note that some 2-faces correspond to the interior of pixels already existing on the original image: they are called *original* faces. The other faces added by the subdivision and immersion are called *artificial faces* in the following. The resulting set-valued image, defined over the Khalimsky grid, has some very strong properties that are given in [18].



(a) Input (b) Subdivided (c) Immersed

Fig. 4: (a) is the input image. (b) is the result of the subdivision. (c) is the result of the immersion into the Khalimsky grid. 0-faces are represented by dots, 1-faces by segments and 2-faces by squares.

The image faces are sorted by the SORT function (see Figure 5). The sort is performed in linear time using a hierarchical queue Q. It begins the propagation with p_∞ chosen arbitrarily, and explores the flat zone at the level λ. Each time a face p is visited at a level λ, faces n on its 4-neighborhood are pushed on the hierarchical queue at a level determined by the position of its value with regard to λ. If $\lambda \in \mathcal{F}(n)$ then there exists a level line at the level λ passing through n; thus n is pushed at the level λ. If $\forall k \in \mathcal{F}(n), \lambda > k$, then the next level to be reached after λ is $m = \max_{\lambda \in \mathcal{F}(n)}(\lambda)$. Thus, n is pushed at this level m; this ensures a continuous propagation of gray levels. If $\forall k \in \mathcal{F}(n), \lambda < k$, then the next level to be reached after λ is $m = \min_{\lambda \in \mathcal{F}(n)}(\lambda)$. Thus, n is pushed at this level m; this, too, ensures a continuous propagation

```
function SORT(f, p_∞)
    Q ← list of queues
    R ← empty array
    λ ← mean(F(p_∞))
    Q[λ] ← p_∞
    while any queue of Q is not empty do
        while Q[λ] is not empty do
            p ← POP(Q[λ])
            PUSH(R, p)
            for all n ∈ N_4(p) that has not been visited yet do
                if λ ∈ F(n) then
                    F^♭(n) ← λ
                else if λ < min(F(n)) then
                    F^♭(n) ← min(F(n))
                else
                    F^♭(n) ← max(F(n))
                PUSH(Q[F^♭(n)], n)
    return (R, F^♭)
```

Fig. 5: Sort procedure used by COMPUTETREE. $\min$ and $\max$ design the lower and high bound of an interval.

of gray levels. The propagation plays two roles: it sorts the faces in a way that it is virtually following the "childhood" relationship of the tree of shapes (this gives the array $\mathcal{R}$), and flattens the interpolated image choosing deterministically a single value in place of ranges on artificial faces. We call $\mathcal{F}^\flat$ the flattened $\mathcal{F}$.

The tree is built by the UNIONFIND function, which relies on the union-find algorithm studied by Tarjan in [19], reading the sorted faces stored in $\mathcal{R}$ in reverse order. It returns a *parent* function, named par, which associates to each face its parent on the tree of shapes. Path compression and union-by-rank are used to ensure the quasi-linear complexity $\mathcal{O}(\alpha(n)n)$, where $\alpha(n)$ designs the inverse ackermann function. Because the propagation visits faces virtually starting from the root of the tree of shapes down to its leaves, the union-find will build the tree starting from the leaves up to the root. In that way, we have the guaranty that no node of the tree of shapes will be seen before its children by the union-find. This ensures that the whole tree is built correctly at the end of the union-find. However this tree contains faces which do not belong to the original image. Therefore, artificial faces must be removed by the CANONICALIZE function. More information about the use of the union-find algorithm in mathematical morphology can be find in [16] and [20].

The CANONICALIZE function removes every artificial face from the tree and compresses it ensuring that if for some face p, if $\mathcal{F}^\flat(par(p)) = \mathcal{F}^\flat(p)$, then $\mathcal{F}^\flat(par(par(p))) \neq \mathcal{F}^\flat(p)$ (unless $par(p)$ is the root of the tree). The pseudo-code of the serial canonicalization is not given here to save some room.

3. DESCRIPTION

To our knowledge no tree of shapes computation algorithm has been parallelized yet. Here we propose a new parallel algorithm that works the same way as the quasi-linear algorithm [2] presented in Figure 2, with the union-find sub-algorithm replaced by a parallel max-tree computation algorithm. The scheme of the proposed parallel algorithm is given by Figure 6.

```
function COMPUTETREE(f, p_∞)
    F ← PARALLELIMMERSE(f)
    Q ← list of queues
    λ ← mean(F(p_∞))
    Q[λ] ← p_∞
    F^ord ← PARALLELSORT(F, Q, λ, 0)
    par ← PARALLELMAXTREE(F^ord)
    return CANONICALIZE(par, F^ord)
```

Fig. 6: Algorithmic scheme of the parallel algorithm.

The PARALLELIMMERSE function is the same as the IMMERSE function described in Figure 2 though this time the valuation of each face is done in parallel. Since the valuation of each face does require only informations about its local configuration, this parallelization is straightforward.

The PARALLELSORT procedure (see Figure 7) sorts the faces in a very similar way as described by Figure 5 with two major changes. First, the sort is done in parallel. This is possible because of the following property:

```
procedure PARALLELSORT(F, Q, F^ord, λ, ord)
    Q[λ] ← p_∞
    while any queue of Q is not empty do
        while Q[λ] is not empty do
            p ← POP(Q[λ])
            Ford(p) ← ord
            for all n ∈ N_4(p) that has not been visited yet do
                if λ ∈ F(n) then
                    PUSH(Q[λ], n)
                else if λ < min(F(n)) then
                    PUSH(Q[min(F(n))], n)
                else
                    PUSH(Q[max(F(n))], n)
        ord ← ord + 1
        S_λ^+ ← Q[λ..max value]
        S_λ^- ← Q[0..λ]
        λ' ← highest level having faces on S_λ^-
        Run PARALLELSORT(F, S_λ^-, F^ord, λ', ord) on another thread.
                                    ▷ This thread continues with S_λ^+
        Q ← S_λ^+
        λ ← smallest level having faces on S_λ^+
    Wait all child.
```

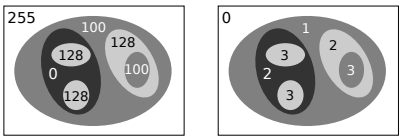
Fig. 7: The parallel face sorting procedure used. $\min$ and $\max$ design the lower and high bound of an interval.

Property 1 *After each propagation step each unvisited distinct sub-tree of the Tree of Shapes correspond to a distinct connected component of the remaining pixels.*

During the sort, after the propagation of a level λ, let $\mathcal{S}$ be the set of faces contained by the hierarchical queue Q. Note that this queue contains every faces of the interior contours of the holes of the flat zones extracted so far. Let $\mathcal{S}_\lambda^+ = \{x \in \mathcal{S} \mid \mathcal{F}^\flat(x) > \lambda\}$ and $\mathcal{S}_\lambda^- = \{x \in \mathcal{S} \mid \mathcal{F}^\flat(x) < \lambda\}$. Since the propagation at the level λ just ended, we have $\{x \in \mathcal{S} \mid \mathcal{F}^\flat(x) = \lambda\} = \emptyset$ hence $\mathcal{S}_\lambda^+ \cup \mathcal{S}_\lambda^- = \mathcal{S}$. Thus, continuing

the propagation on $\mathcal{S}_\lambda^-$ in parallel to the propagation on $\mathcal{S}_\lambda^+$, each thread using its own hierarchical queue, is possible since $\mathcal{S}_\lambda^+$ and $\mathcal{S}_\lambda^-$ contain the borders of disjoint connected components of the holes due to the Property 1. Figure 2b shows an example of this partitioning.

Second, PARALLELSORT does not output $\mathcal{F}^\flat$, nor $\mathcal{R}$ as SORT did in Figure 3. Instead, it returns a $\mathcal{F}^{\text{ord}}$ function which associates to each face its level on the tree of shapes. Therefore, the deeper a shape is on the tree, the higher is the value of $\mathcal{F}^{\text{ord}}$ associated to its faces. Since the propagation sub-algorithm starts on the root of the tree of shapes, and propagates down to the leaves continuously, it appears that the max-tree of $\mathcal{F}^{\text{ord}}$ is the same as the tree of shapes of $\mathcal{F}$ as illustrated by Figure 8.



(a) Original image (b) Re-valued image

Fig. 8: The original image (a) and the associated $\mathcal{F}^{\text{ord}}$ (b) ; the max-tree of (b) coincides with the tree of shapes of (a).

Because of the equivalence of the tree of shapes of $\mathcal{F}$ and the max-tree of $\mathcal{F}^{\text{ord}}$, any parallel max-tree computation algorithm may be used in place of the PARALLELMAXTREE function. Several parallel max-tree computation algorithms exist in the literature [21, 22] and may be used here. A comprehensive comparison of those algorithms has been proposed in [20].

Finally, assuming the max-tree computation algorithm returns a par function which associates to each face of $\mathcal{F}$ its parent on the max-tree, the final tree is obtained after removal of every artificial faces by the CANONICALIZE function (Figure 9). It is very similar to the serial version, except that it is split into two passes since we no longer have access to the array $\mathcal{R}$. Note that the canonicalization is not parallel.

4. COMPARISON

We use a test set of 14 very classical images (including lena, pepper, baboon, house, etc.) All images have been resized to 2500×2500 to make their computation times comparable and significant. For each different algorithm, the minimum and maximum times, respectively obtained on an image of the test set, are depicted by ticks, and the central bar depicts the median of every computation times. Those benchmarks were run under Arch Linux using a 6-core processor Intel core i7 at 3.2GHz with 16GB of RAM. The parallel version uses *four* threads and is about *three* times as fast as the serial version. That clearly means that the parallelization of the quasi-linear tree of shapes computation algorithm is effective.

For this comparison, we rely on our C++ image processing library, Milena, described in [24]. This library, particularly well-suited to experiment both with classical images and

```
function CANONICALIZE(F^♭, par)
    for all point p do
        if p ≠ par(p) and F^♭(par(p)) = F^♭(p) then
            FIND_REPR(F^♭, par, p, p)
    for all point p do
        if F^♭(par(par(p))) = F^♭(par(p)) and F^♭(par(p)) ≠ F^♭(p)
then
            par(par(p)) = par(p)
function FIND_REPR(F^♭, par, p, r)
    if par(p) = p or F^♭(par(p)) ≠ F^♭(p) then
        if r is primary and p is not primary then
            if par(p) = p then
                par(r) ← r
            else
                par(r) ← par(p)
            par(p) ← r
            return r
        return p
    else
        if p is primary then
            par ← FIND_REPR(F^♭, par, par(p), p)
        else
            par ← FIND_REPR(F^♭, par, par(p), r)
        if par ≠ p then
            par(p) ← par
    return par(p)
```

Fig. 9: Canonicalization procedure.

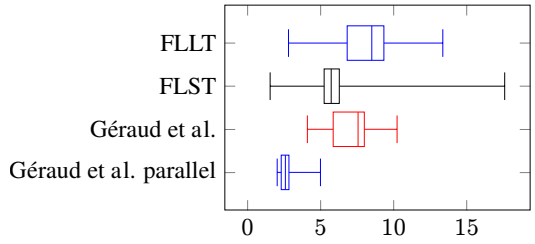


Fig. 10: Computation times (in seconds) on a classical image test set of the following algorithms: FLLT [3], FLST [23], Géraud *et al.* [2], and this paper proposal.

with advanced topological image structures [25], is available as *Free Software* under the GNU GPL v2.

5. CONCLUSION

In this paper we have presented a first parallel version of an algorithm to compute the morphological tree of shapes of nD images. On a test set of classical 2D images, we have shown that the parallelization proposed here is really effective. The perspectives of this work is to perform intensive tests on 3D images, for the quasi-linear algorithm [2] is the only algorithm that is usable when $n > 2$. Another perspective is to get a parallel algorithm to compute the recent proposals of a tree of shapes for color images [26, 27].

Since we advocate *reproducible research*, all the materials used for this paper (images, diagrams, and source code) is available on the Internet from:

`http://www.lrde.epita.fr/wiki/Publications/crozet.14.icip`

References

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

[2] T. Géraud, E. Carlinet, S. Crozet, and L. Najman, "A quasi-linear algorithm to compute the tree of shapes of n-D images," in *Proc. of the Intl. Symposium on Mathematical Morphology (ISMM)*. 2013, vol. 7883 of *Lecture Notes in Computer Science*, pp. 98–110, Springer.

[3] 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.

[4] F. Cao, J.-L. Lisani, J.-M. Morel, P. Musé, and F. Sur, *A Theory of Shape Identification*, vol. 1948 of *Lecture Notes in Mathematics*, Springer, 2008.

[5] V. Caselles and P. Monasse, "Grain filters," *J. of Math. Imaging and Vision*, vol. 17, no. 3, pp. 249–270, 2002.

[6] Y. Xu, T. Géraud, and L. Najman, "Morphological filtering in shape spaces: Applications using tree-based image representations," in *Proc. of the Intl. Conference on Pattern Recognition (ICPR)*. IAPR, 2012, pp. 485–488.

[7] Y. Xu, T. Géraud, and L. Najman, "Context-based energy estimator: Application to object segmentation on the tree of shapes," in *Proc. of the IEEE Intl. Conference on Image Processing (ICIP)*, 2012, pp. 1577–1580.

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

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

[10] Y. Pan, J.D. Birdwell, and S.M. Djouadi, "Preferential image segmentation using trees of shapes," *IEEE Trans. on Ima. Proc.*, vol. 18, no. 4, pp. 854–866, 2009.

[11] C. Ballester, V. Caselles, L. Igual, and L. Garrido, "Level lines selection with variational models for segmentation and encoding," *Journal of Mathematical Imaging and Vision*, vol. 27, pp. 5–27, 2007.

[12] A. Pardo, "Semantic image segmentation using morphological tools," in *Proc. of the IEEE Intl. Conference on Image Processing (ICIP)*, 2002, vol. 2, pp. 745–748.

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

[14] Y. Pan, "Top-down image segmentation using the mumford-shah functional and level set image representation," in *Proc. of the IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 2009, pp. 1241–1244.

[15] C. Berger *et al.*, "Effective component tree computation with application to pattern recognition in astronomical imaging," in *Proc. of the IEEE Intl. Conf. on Image Processing (ICIP)*, 2007, vol. 4, pp. 41–44.

[16] T. Géraud, "Ruminations on Tarjan's Union-Find algorithm and connected operators," in *Proc. of the Intl. Symp. on Math. Morphology (ISMM)*. 2005, vol. 30 of *Comp. Imaging and Vision*, pp. 105–116, Springer.

[17] E Khalimsky and R Kopperman, "Computer graphics and connected topologies on finite ordered sets," *Topology and its Applications*, vol. 36, pp. 1–17, 1990.

[18] L. Najman and T. Géraud, "Discrete set-valued continuity and interpolation," in *Proc. of the International Symposium on Mathematical Morphology (ISMM)*. 2013, vol. 7883 of *LNCS*, pp. 37–48, Springer.

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

[20] E. Carlinet and T. Géraud, "A comparison of many max-tree computation algorithms," in *Proc. of the Intl. Symp. on Mathematical Morphology (ISMM)*. 2013, vol. 7883 of *LNCS*, pp. 73–85, Springer.

[21] P. Matas *et al.*, "Parallel algorithm for concurrent computation of connected component tree," in *Adv. Concepts for Intelligent Vision Systems*, 2008, pp. 230–241.

[22] M.H.F. Wilkinson, H. Gao, W.H. Hesselink, J.E. Jonker, and A. Meijster, "Concurrent computation of attribute filters on shared memory parallel machines," *IEEE Trans. on PAMI*, vol. 30, no. 10, pp. 1800–1813, 2008.

[23] V. Caselles and P. Monasse, *Geometric Description of Images as Topographic Maps*, vol. 1984 of *Lecture Notes in Computer Science*, Springer-Verlag, 2009.

[24] 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 *Proc. of the IEEE Intl. Conf. on Image Processing (ICIP)*, 2010, pp. 1941–1944, `http://olena.lrde.epita.fr`.

[25] R. Levillain, T. Géraud, and L. Najman, "Writing reusable digital topology algorithms in a generic image processing framework," in *Applications of Discrete Geometry and Mathematical Morphology*. 2012, vol. 7346 of *LNCS*, pp. 140–153, Springer-Verlag.

[26] E. Carlinet and T. Géraud, "A morphological tree of shapes for color images," in *Proceedings of the 22nd International Conference on Pattern Recognition (ICPR)*, 2014, To appear.

[27] E. Carlinet and T. Géraud, "Getting a morphological tree of shapes for multivariate images: Paths, traps, and pitfalls," in *Proc. of the 21st IEEE International Conference on Image Processing (ICIP)*, 2014, To appear.