Multi-scale component-tree: A hierarchical representation of sparse objects

Romain Perrin, Aurélie Leborgne, Nicolas Passat, Benoît Naegel, Cédric Wemmert

Multi-Scale Component-Tree: A Hierarchical Representation for Sparse Objects*

Romain Perrin[1][0009-0001-6534-9727], Aurélie Leborgne[1][0000-0001-8456-2745], Nicolas Passat[2][0000-0002-0320-4581], Benoît Naegel[1][0000-0002-7695-1473], and Cédric Wemmert[1][0000-0002-4360-4918]

[1] University of Strasbourg, ICube, France
[2] University of Reims Champagne-Ardenne, CReSTIC, France
romain.perrin@unistra.fr

Abstract. Component-trees are hierarchical structures developed in the framework of mathematical morphology. They model images via the inclusion relationships between the connected components of their successive threshold sets. There exist many variants of component-trees, but to the best of our knowledge, none of them deals with the representation of the image at different scales. In this article, we propose such a variant of component-tree that tackles this issue, namely the Multi-Scale Component-Tree (MSCT). We describe an algorithmic scheme for building the MSCT from the standard computation of component-trees of the image, seen from its lowest to its highest scale. At each step, a local upscaling is performed on relevant parts of the image, corresponding to nodes of the MSCT which are selected according to a stability analysis. The last step builds elements which are part of the standard component-tree (at the highest, native scale of the image). The MSCT provides a compact, efficient representation of images compared to the standard (single-scale) component-tree. In particular, the MSCT is especially suited to analyse images containing sparse objects, which require to be represented at a high scale, vs. large background regions that can be losslessly represented at a lower scale. We illustrate the relevance of the MSCT in the context of cellular image segmentation.

Keywords: Component-tree · Multi-scale · Hierarchical representation · Segmentation · Mathematical morphology

1 Introduction

The component-tree [22] is a morphological graph-based model defined by considering the connected components of binary sets obtained from successive thresholdings of an image. Initially proposed in the field of statistics [10], the component-tree has been redefined in the theoretical framework of mathematical morphology and involved, in particular, in the development of morphological tools [4, 22]. In particular, the component-tree allows to design connected operators [23] that can process images while preserving contour information.

* This work was supported by the French Agence Nationale de la Recherche (ArtIC, Grant ANR-20-THIA-0006, and Grant ANR-23-CE45-0015).

The component-tree led to the development of various filtering and segmentation approaches, mainly by building upon two paradigms: the selection of nodes by attribute filtering [11] or the computation of optimal cuts within the tree [9]. The component-tree was then involved in various application fields, e.g. medical imaging [24], astronomy [1], agriculture imaging [3], microscopic imaging [14].

The popularity of the component-tree relies, on the one hand, on its ability to encode an image in a compact and lossless fashion, and on the other hand to the low complexity of its construction and handling. In particular, the component-tree can be built in quasi-linear time [1, 5, 16]. Many algorithms have been proposed for its construction, based e.g. on watershed-based, flooding-based or merging-based strategies (see [5] for a survey). More recently, efforts were geared towards its efficient construction in the case of very large images. Götz et al. [8] presented a distributed-memory parallel method for computation of component-trees, while Moschini et al. [15] worked on shared-memory parallel computation at extreme dynamic ranges. Gazagnes et al. [6, 7] introduced novel parallel algorithms for max-tree construction on tera-scale images. Recently, Blin et al. [2] offered the first GPU implementation for building a component-tree.

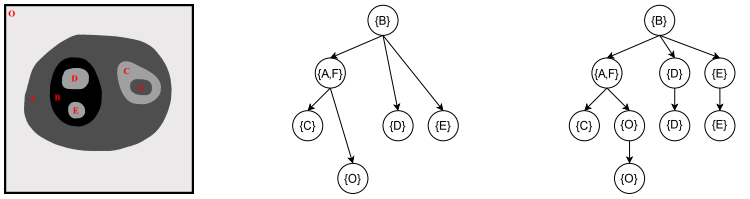
Many variants of the component-tree were proposed over the last years. Some of them are trees, e.g. the hyperconnection tree [21] (that extends the component-tree to the case of hyperconnections), the multivalued component-tree [12] (that handles values endowed with hierarchical orders), the shaping paradigm [25] (that builds component-trees on component-trees), the complete tree of shapes [17] (that unifies the min- and max-trees and links them to the tree of shapes). Others are directed acyclic graphs, e.g. the component graph [19] (that handles values endowed with partial orders), the asymmetric hierarchies [20] (that handle non-symmetric adjacency links). Nonetheless, to the best of our knowledge, there has been no attempt to design variants of the component-tree that deal with the multiscale modeling of an image. The closest works on that topic deal with the notion of component-hypertrees [18], that provide a forest of component-trees induced by a family of increasing connectivities. In this context, the notion of scale was considered at the topological level.

In this article, we aim to design a variant of component-tree that handles the notion of multiple scales at the spatial level. This new tree is called the *Multi-Scale Component-Tree* (MSCT, for brief). By contrast with the standard component-tree, that models the image without taking into account the local informativeness, the MSCT aims to adapt the scale of modeling according to the carried information, with lower (resp. higher) scales where few (resp. many) details / structures of interest are available.

2 Component-Tree

Let $f : \mathbb{Z}^2 \to \mathbb{N}$ be a 2D image. In practice, this image is finite. Without loss of generality, we can them assume that it is defined on a square support $\mathbb{S}_n = [\![0, 2^n - 1]\!]^2 \subset \mathbb{Z}^2$ ($n \in \mathbb{N}$), and then composed of $N = 2^{2n}$ pixels.

We can also assume that it takes its values in a finite subset $\mathbb{V} \subset \mathbb{N}$, that can be chosen as $\mathbb{V} = [\![0, m-1]\!]$. The image f being finite, we assume that $f(x) = 0$ for any $x \notin \mathbb{S}_n$. We endow $\mathbb{N}$ (and thus $\mathbb{V}$) with the standard order $\leqslant$. We endow $\mathbb{Z}^2$ (and thus $\mathbb{S}_n$) with a connectivity framework inherited from the standard adjacencies in digital topology.



(a) Image (b) Max-tree (c) Component-tree

Fig. 1: The max-tree (b) and the component-tree (c) of a grey-level image (a).

For any nonempty subset $X \subseteq \mathbb{Z}^2$, we note $CC(X) \subset 2^{\mathbb{Z}^2}$ the set of the connected components (i.e. the maximal connected subsets) of X.

Let $\lambda \in \mathbb{V}$. We note the upper threshold set of f at value λ as $[f \geqslant \lambda] = \{x \in \mathbb{Z}^2 \mid f(x) \geqslant \lambda\}$. The family $\{CC([f \geqslant \lambda])\}_{\lambda \in \mathbb{V}}$ is increasing. Let $\lambda \in \mathbb{V}$ ($\lambda \neq 0$) and $X_{\lambda+1} \in CC([f \geqslant \lambda + 1])$ be a connected component of the threshold set of f at value $\lambda + 1$. There exists a unique $X_\lambda \in CC([f \geqslant \lambda])$ such that $X_{\lambda+1} \subseteq X_\lambda$.

This hierarchical organisation of the connected components can be represented in an inclusion tree, which is called the max-tree. (By considering the dual order $\geqslant$ on $\mathbb{N}$, one may define the dual min-tree).

More formally, the max-tree is defined as the Hasse diagram of the partially ordered set $(\bigcup_{\lambda \in \mathbb{V}} CC([f \geqslant \lambda], \subseteq)$. Its root (i.e. its maximum) is the set $\mathbb{Z}^2$, which is the unique connected component of $[f \geqslant 0]$. Its leaves (i.e. its minimal elements) are the flat zones of locally maximal value in the image. An example of max-tree is illustrated in Figure 1(b) for the image depicted in Figure 1(a). One may note that a set may be a connected component for many successive threshold sets. While the max-tree only models such an element once, we may also consider this connected component at each value where it appears, leading to a multiset of connected components instead of a set. This paradigm provides a less compact version of the max-tree, that is sometimes called the component-tree, illustrated in Figure 1(c).

3 Multi-Scale Component-Tree

In this section, we explain how to build the Multi-Scale Component-Tree. The purpose of this construction process is to represent background and/or non-relevant parts of the image within flat zones at the lowest scales, while representing relevant / fine detailed parts of the image within flat zones at the highest scales. The proposed method, summarized in Algorithm 1 and Figure 2, revolves around three steps:

1. downsampling of the gray-scale image (Section 3.1 and Algorithm 1, line 2);
2. definition of the *Base Component-Tree* at the lowest scale (Section 3.2 and Algorithm 1, line 3);
3. iterative upsampling that promotes significant regions from one scale to the next (Section 3.3 and Algorithm 1, lines 4–10).

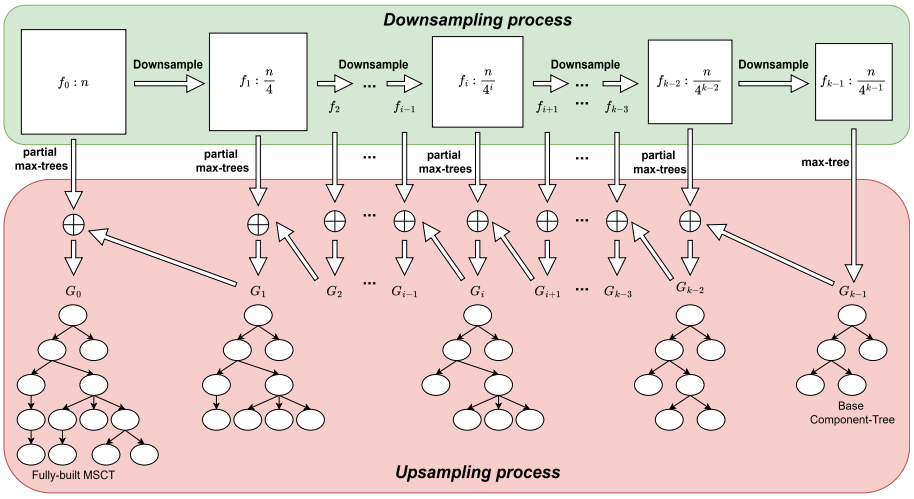


Fig. 2: Illustration of the MSCT construction (see Section 3 and Algorithm 1).

Algorithm 1: MSCT construction.

Data: $f : \mathbb{S}_n \to \mathbb{V}$ (gray-scale image), $k \in \mathbb{N}^\star$ number of scales
Result: $G = (V, E)$ Multi-Scale Component-Tree of f

begin
  $F \longleftarrow \{f_0, \ldots, f_{k-1}\}$ with $f_0 = f$ and $\forall i \in [\![1, k-1]\!], f_i \longleftarrow MaxPool(f_{i-1})$
  $G \longleftarrow G_{k-1} \longleftarrow MaxTree(f_{k-1})$
  for *i from* $k-2$ *down to* 0 **do**
    $C_i \longleftarrow NodeSelection(G, i)$
    foreach $N \in C_i$ **do**
      $G_i(N) \longleftarrow PartialMaxTree(f_i, N)$
      $Merge(G, G_i(N), N)$
    end
  end
end

3.1 Downsampling process

The input of the construction procedure is an image $f : \mathbb{S}_n \to \mathbb{V}$ such as defined in Section 2. From f, we define a set of downsampled images $F = \{f_i : \mathbb{S}_{n-i} \to \mathbb{V}\}_{i=0}^{k-1}$ where k is the number of scales ($1 \leqslant k \leqslant n$). For each $i \in [\![0, k-1]\!]$, the image f_i is the i-th downsampled version of f, i.e. the image at scale $\frac{1}{2^i}$.

We have $f_0 = f$. For each $i \in [\![0, k-1]\!]$, the image f_{i+1} is obtained from the image f_i by collapsing the 2×2 sets of points of $\mathbb{S}_{n-i}$ into one point of $\mathbb{S}_{n-(i-1)}$. The max-tree is an image model which is especially well-fitted for modeling images where the relevant information is related to the local maxima in the image. Based on this fact, we assume that in the MSCT, it is relevant to preserve the areas of greatest values vs. a background

of lower values. As a consequence, we use a maximum policy to define f_{i+1} from f_i. More precisely, for any $x \in \mathbb{S}_{n-(i+1)}$, we set

$$f_{i+1}(x) = \max_{0 \leqslant a,b, \leqslant 1} \{f_i(2x + (a, b))\} \tag{1}$$

This operation that defines f_{i+1} from f_i will be noted $MaxPool$ (by reference to the analogue operation usually considered in deep learning).

3.2 Base Component-Tree computation

The purpose of the MSCT is to represent the relevant information carried by a (potentially large) image while minimizing its space cost and then the time cost required for handling it. As a consequence, the construction of the MSCT starts from the max-tree at the lowest scale, i.e. the max-tree of f_{k-1}, which is defined on the set $\mathbb{S}_{k-1}$ that contains only $\frac{1}{4^{k-1}}N$ points, compared to f that contains N points. The max-tree of this lowest scale image f_{k-1} is called the Base Component-Tree.

We note $MaxTree : \mathbb{V}^{\mathbb{Z}^2} \rightarrow \mathbb{T}$ the function that maps any (finite) image $f : \mathbb{Z}^2 \rightarrow \mathbb{V}$ onto its max tree $G = (V, E) \in \mathbb{T}$ (where $\mathbb{T}$ is the set of all the finite rooted trees). We recall that the computational cost of optimal algorithms that implement $MaxTree$ is $O(n \log n)$ where n is the number of points of the image support [5].

Here, the Base Component-Tree $G_{k-1} = (V_{k-1}, E_{k-1}) \in \mathbb{T}$ is computed from the image $f_{k-1} : \mathbb{S}_{k-1} \rightarrow \mathbb{V}$, and the induced time cost is then $\frac{N}{4^{k-1}} \log \frac{N}{4^{k-1}}$.

3.3 Upsampling process

The MSCT $G = (V, E)$ is initialized as the Base Component-Tree G_{k-1} of f_{k-1}. At this stage, the whole image f, including both informative and non-informative parts, is represented at the lowest scale. The purpose is now to modify this tree G in order to represent the informative regions of the image at higher scales, according to their degree of relevance.

This process is carried out iteratively, scale by scale, from the lower to the higher (Algorithm 1, line 4). At each iteration / scale $i \in [\![0, k-2]\!]$, a set of nodes $C_i \subseteq V$ is selected from the tree G. These nodes are those assumed as containing a relevant information that motivates the computation of a local max-tree on their associated region. The procedure of selecting these nodes in G at scale i (noted $NodeSelection(G, i)$ in Algorithm 1, line 5) is described in the next subsection.

For each selected node $N \in C_i$ (Algorithm 1, line 6), the max-tree of the image f_i restricted to the region $N \subseteq \mathbb{S}_{n-i}$ is computed. This "local" max-tree computation, noted $PartialMaxTree(f_i, N)$ is nothing but the computation of a standard max-tree on a given (connected) region N. Note in particular that algorithmically, the behaviour of $MaxTree(f_{k-1})$ (Algorithm 1, line 3) is the same as $PartialMaxTree(f_k, \mathbb{S}_{n-k-1})$ (In our algorithmic scheme, we used a version of Najman and Couprie's algorithm [16][3]). Once this new partial max-tree $G_i(N) = (V_i(N), E_i(N))$ is computed, it must be embedded in the MSCT G. This embedding and its side effects on the structure of G (Algorithm 1, line 8) are detailed at the end of this section.

[3] Although we only consider integer values, Najman and Couprie's algorithm uses Tarjan's Union-Find method and is able to efficiently process floating-point values as well.

Nodes selection for local upsampling At each iteration / scale i, a set C_i of nodes is selected for an upsampling procedure. The choice of the most relevant nodes is carried out based on a priority score assigned to each node $N \in V$ of the current MSCT $G = (V, E)$. This score is computed based on the notion of Maximally Stable Extremal Regions (MSER) [13]. The MSER stability value of a node $N \in V$, that belongs to a threshold set at value $v \in \mathbb{V}$ is defined as $MSER(N) = \frac{|N_{-\Delta}| - \sum_p |N^p_{+\Delta}|}{|N|}$ where $N_{-\Delta} \in V$ is the ancestor node at value $v - \Delta$ such that $N \subseteq N_{-\Delta}$, and the $N^p_{+\Delta} \in V$ are all the descendant nodes of N at value $v + \Delta$, i.e. such that $N \supseteq N^p_{+\Delta}$. The nodes of V are then sorted by decreasing stability.

The set of selected nodes is defined by choosing the nodes by decreasing stability in the list. The selected nodes have to be non-overlapping, which means that for any two distinct nodes $N_1, N_2 \in C_i$ we have neither $N_1 \subseteq N_2$ nor $N_2 \subseteq N_1$. Indeed, each of these nodes will become the root of a partial tree that will be embedded in the MSCT G. It is then required that none of these new trees be part of another. As a consequence the ancestors and descendants of the selected nodes are progressively discarded from the list. The selection ends once the list is empty.

Local upsampling For each node $N \in V$ selected from the above process, a max-tree $G_i(N) = (V_i(N), E_i(N))$ of the image f_i restricted to the support N is computed. This new max-tree $G_i(N)$ is then dedicated to replace the current subtree of G starting at node N. However, the node N which is the root of the subtree to be replaced is defined at a given value $v \in \mathbb{V}$, and its parent node N' is defined at a value $v' \in \mathbb{V}$ with $v' < v$. By contrast, the new partial tree built in the region N has its root at a value lower than v. In this context, the tree $G_i(N) = (V_i(N), E_i(N))$ has to be split into two parts, that must be processed distinctly. On the one hand, all the nodes $N_u \in V_i(N)$ that are defined for a value $u \in \mathbb{V}$ with $u \leqslant v'$ have to be merged with the ancestor node $N'_u \in V$ at value u, which is the ancestor of N in G. The "new" version of the node N'_u then corresponds to $N'_u \cup N_u$. On the other hand, all the nodes $N_u \in V_i(N)$ such that their parent node $N_w \in V_i(N)$ is defined for a value $w \leqslant v'$ now become the root of a partial tree of f_i restricted to N_u, and this root N_u has to be connected to the node $N_{v'}$. In other words, the node N is replaced by a forest extracted from its max-tree $G_i(N)$, that is connected to the parent node of N, whereas the remainder of the tree is absorbed by the branch of G located between N and the root. This policy leads in particular to a structure of MSCT where each node may encode pixels at different scales. An example of such replacement is illustrated in Figure 3.

4 Object segmentation using the MSCT

The built MSCT $G = (V, E)$ now contains a hierarchy of flat-zones with pixels at different scales. The MSCT can be used, in particular, for segmentation tasks. A possible use-case is cell segmentation, where high contrast denotes the presence of a bio-marker (foreground) while low contrast denotes its absence (background). We propose a segmentation method illustrated by Algorithm 2 which involves three steps. First, a set of nodes of interest is extracted from the MSCT (line 3). Second, for each node of interest,

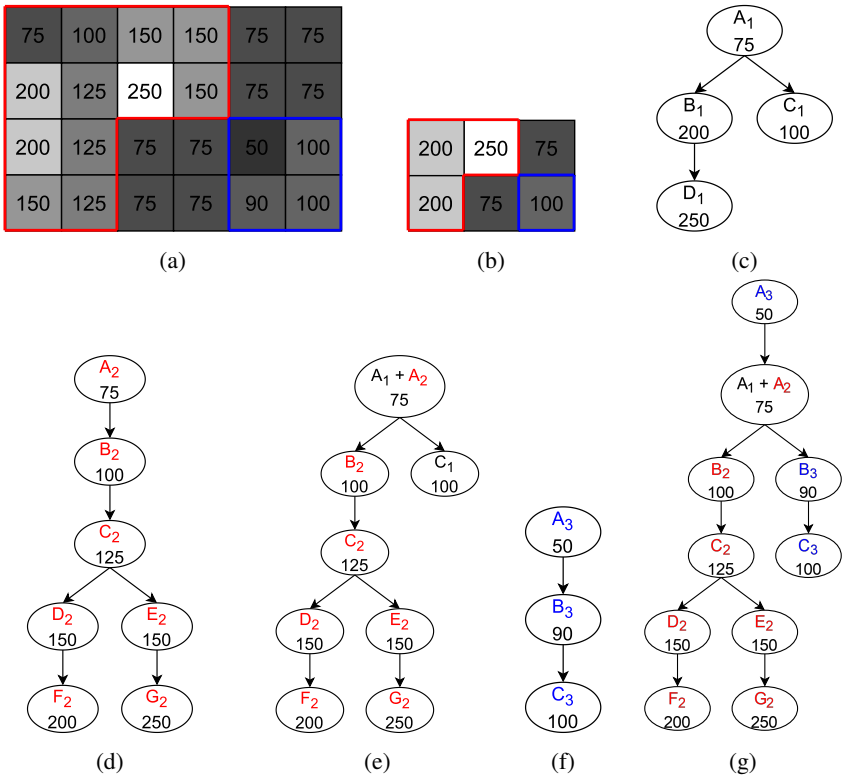


Fig. 3: Example of MSCT sub-tree replacement during upsampling. (a) is the input image and (b) is the downsampled image at half scale. (c) is the max-tree of (b) also referred to as the base component-tree of the MSCT. (d) is the partial max-tree of the red L-shaped area of (a). (e) is the merging of the partial max-tree (d) on the MSCT (c). (f) is the partial max-tree of the blue square-shaped area of (b). (g) is the merging of the MSCT (e) with the partial max-tree (f) and is also the fully-built MSCT. It is important to note that (f) introduces gray-levels that do not exist in the MSCT, even changing the root of the MSCT (e) (levels 50 and 90) prior to being merged with (f).

its local sub-tree is filtered (line 4) according to a maximum MSER value. The filtered nodes are intermediate clusters containing either objects or clusters of touching objects. Third, each cluster is cleaned to remove the background around the object(s) (line 6) and then undergoes a watershed to separate the cluster into individual objects (line 7).

4.1 Nodes selection for segmentation

The retrieval of objects of interest is analogous to the upsampling process described in Section 3.3, given that the extracted flat-zones inherently encompass these objects. Subsequently, the same function is employed to define a set of disjoint sub-trees, facilitating the separation into distinct objects (line 3).

Algorithm 2: MSCT segmentation

Data: $G = (V, E)$ MSCT, $MSER_{max} \in \mathbb{N}$ maximum MSER value, $f : \mathbb{Z}^2 \to \mathbb{N}$ input image, $n \in \mathbb{N}$ subdivision factor
Result: $P = \{P_0, \ldots, P_k\}$

```
1  begin
2      P ⟵ ∅
3      C_0 ⟵ UpsampleNodeSelection(G)
4      C'_0 ⟵ FilterTree(G, C_0, MSER_max)
5      for c ∈ C'_0 do
6          c' ⟵ FillHoles(Otsu(GaussianFilter(C_0)))
7          S ⟵ Watershed(f, c, UltimateErosion(c'))
8          P ⟵ P ∪ S
9      end
10 end
```

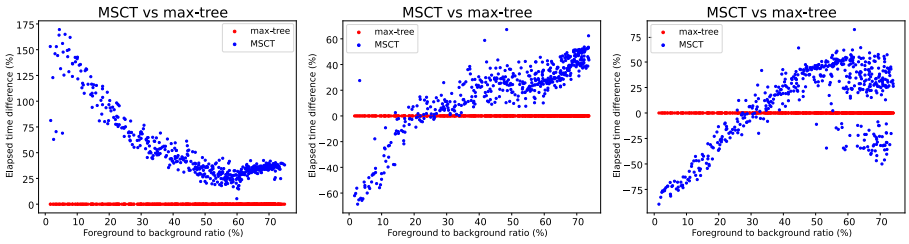
4.2 MSCT filtering

A first cluster separation can be performed by filtering nodes of the MSCT. A simplified version of the MSCT sub-tree is computed. It consists of reducing each branch to a single node representing a local flat-zone and being assigned the minimum MSER value among all nodes of the branch. This simplified tree is filtered using a maximum MSER value criterion (line 4). The function *KeepNode* (Equation (2)) is recursively called on each node of the sub-tree starting at its root. Leaves of the filtered sub-tree are the intermediate clusters.

$$KeepNode(n, v) = \begin{cases} True, & \text{if } n.nChildren = 0 \wedge n.mser \leq v \\ True, & \text{if } n.children > 0 \wedge n.mser \leq v \wedge \forall n' \in n.children, n'.mser \leq v \\ False, & \text{otherwise} \end{cases} \tag{2}$$

4.3 Object segmentation

Each intermediate cluster $c \in C'_0$ is processed separately (lines 5–9). In a first step, the intermediate cluster is cleaned to retrieve the exact contour of its underlying object(s) and get rid of background pixels (line 6). The presence of these background pixels is a side effect of the upsampling steps due to the maximum nature of the downsampling operations. They are removed using a Gaussian filter followed by a Otsu thresholding and a hole filling operation. In a second step, the number of individual objects inside an intermediate cluster is estimated by performing an ultimate erosion on its previously cleaned flat-zone (line 7). The centroid of each ultimate erosion connected component is then used to initialize a watershed algorithm resulting in a complete partition of said intermediate cluster. The set of all watershed partitions of all intermediate clusters P is the final segmentation result, with each partition being a connected component representing one individual object.



(a) MSCT $k = 2$ (b) MSCT $k = 3$ (c) MSCT $k = 4$

Fig. 4: Computing-time differences for several downsampling factors. The MSCT (blue curves) are shown relative to the max-trees (red curves).

5 Experimental results

5.1 Implementation

To assess the validity of the Multi-Scale Component-Tree and its ability to produce satisfactory segmentations of cellular images, we have implemented it in Python. In this section, we discuss important choices regarding algorithms and global parameters.

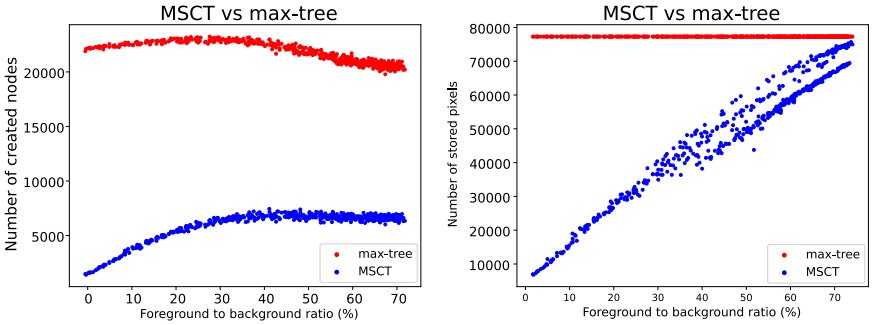
The main parameter to build the MSCT is the number of scales k that also defines the number of downsampling and upsampling steps. It defines the size of the image f_{k-1} used to build the Base Component-Tree G_{k-1}. The MSER parameter δ is computed relative to the tree height at scale i (cardinal of the support set of f_i) using a parameter $MSER_{height}$. Node selection is constrained by two parameters: the maximum area nodes is set to A_{max} while the maximum stability of nodes is set to $MSER_{select}$. Regarding the segmentation, $MSER_{filter}$ sets the maximum stability of nodes when filtering the tree.

5.2 Computational cost evaluation

The computation time required to build the MSCT depends on multiple factors: the size of the input image, the number of objects or clusters of objects and their respective size. An experiment (Figure 4) is made by generating synthetic images of a fixed size composed of a variable number of bright objects on a dark background. We can observe that the MSCT computation outperforms the regular max-tree when 30% of the image surface consists of objects, using a downsampling factor $k > 2$ (Figures 4(b) and 4(c)). When using a downsampling of $k = 2$, the size of the downsampled image f_1 upon which the Base Component-Tree is computed is only $\frac{N}{4}$ and the cost of choosing and computing partial max-trees outweights the benefit of working with downsampled images (Figure 4a). This illustrates the effectiveness of building the Base Component-Tree (Section 3.2) on a low scale version of the image especially when said image is sparse.

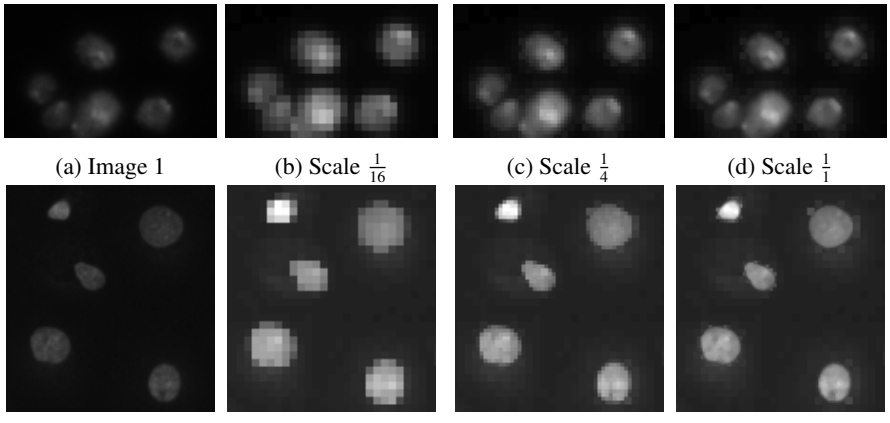
5.3 Space complexity analysis

To complete the second experiment in Section 5.2, the number of created nodes and stored pixels have been measured for growing numbers of objects in the image. We



(a) Created nodes ($k = 2$) (b) Stored pixels ($k = 2$)

Fig. 5: Space complexity measures.



(a) Image 1 (b) Scale $\frac{1}{16}$ (c) Scale $\frac{1}{4}$ (d) Scale $\frac{1}{1}$

(e) Image 2 (f) Scale $\frac{1}{16}$ (g) Scale $\frac{1}{4}$ (h) Scale $\frac{1}{1}$

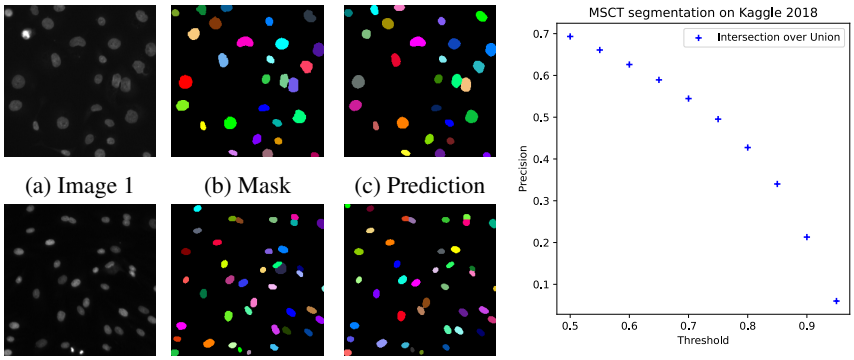
Fig. 6: Examples of images with their reconstruction after each upsampling step.

can observe that the number of created nodes is significantly lower for the MSCT (blue curve in Figure 5(a)) compared to a regular max-tree (red curve in Figure 5(a)). The number of pixels is roughly linear with respect to the sparsity of the image for the MSCT (blue curve in Figure 5(b)) as opposed to constant and equal to the image size for the max-tree (red curve in Figure 5(b)). The reason behind the low number of pixels resides in the multi-scale nature of those pixels as only highly-contrasted flat-zones are promoted from low scale to high scale during the upsampling process (Section 3.3).

5.4 Kaggle 2018

Our method has been tested on the Kaggle 2018 Data Science Bowl dataset[4]. It contains gray-scale images of various sizes, blurriness, cell types and sizes. Only gray-scale

[4] www.kaggle.com/competitions/data-science-bowl-2018



(a) Image 1 (b) Mask (c) Prediction

(d) Image 2 (e) Mask (f) Prediction (g) Intersection over union curve

Fig. 7: Examples of Kaggle 2018 images (a,d) with ground truth masks (b,e) and the MSCT segmentation results (c,f).

images with bright objects on dark background are considered. The following parameters have been used: $k = 3$, $MSER_{height} = 10\%$, $A_{max} = 10\%$, $MSER_{select} = 1$, $MSER_{filter} = 1$.

Some examples of input images and their reconstruction at different steps of the upsampling process are illustrated in Figure 6. A total number of $n = 547$ gray-scale images have been segmented using the method described in Section 4. Figure 7 shows some segmentation results. Kaggle uses an intersection over union (IoU) metric with thresholds between 0.50 and 0.95. A prediction is deemed a true positive if the intersection IoU score of its mask with the ground truth mask is at least equal to the threshold. Figure 7(g) shows the curve of precision per threshold. The global score for Kaggle 2018 is the average of the curve and equals 0.465. The large variability in acquisition techniques and cell sizes presents a significant challenge in determining optimal parameters. Depending on the specific configuration of the image under consideration, these methods frequently result in either under-segmentation or over-segmentation.

6 Discussion

We have introduced the concept of the Multi-Scale Component-Tree (MSCT) and presented an algorithm for its construction on gray-scale images. Our results demonstrate the MSCT ability in storing flat-zones across various scales and its enhanced capability to distinguish between flat-zones encompassing objects of interest and those constituting the background, especially when compared to its single-scale counterpart. The efficiency of the MSCT has been further exemplified through its application in a cellular segmentation task. More complex filtering schemes could be applied prior to the segmentation step. Other attributes could be used as well such as the border gradient, complexity or compactness. When dealing with non-increasing attributes, shaping could be employed, that is building a component-tree of the MSCT, filtering it and re-

constructing the MSCT. In terms of efficient building, the upsampling step could benefit from parallel computing as all selected sub-trees are mutually disjoint by virtue of the selection function. Indeed, partial max-trees could be computed in parallel on different sets of pixels and only the replacement step in the MSCT has to be performed sequentially. Other segmentation methods could be defined to avoid resorting to use watershed like an ellipse fitting model that might offer improvements for the given examples.

References

1. Berger, C., Géraud, T., Levillain, R., Widynski, N., Baillard, A., Bertin, E.: Effective component tree computation with application to pattern recognition in astronomical imaging. In: ICIP. pp. 41–44 (2007)
2. Blin, N., Carlinet, E., Lemaitre, F., Lacassagne, L., Geraud, T.: Max-tree computation on GPUs. IEEE Transactions on Parallel & Distributed Systems **33**, 3520–3531 (2022)
3. Bosilj, P., Duckett, T., Cielniak, G.: Connected attribute morphology for unified vegetation segmentation and classification in precision agriculture. Computers in Industry **98**, 226–240 (2018)
4. Breen, E.J., Jones, R.: Attribute openings, thinnings, and granulometries. Computer Vision and Image Understanding **64**(3), 377–389 (1996)
5. Carlinet, E., Géraud, T.: A comparative review of component tree computation algorithms. IEEE Transactions on Image Processing **23**, 3885–3895 (2014)
6. Gazagnes, S., Wilkinson, M.H.F.: Distributed component forests in 2-D: Hierarchical image representations suitable for tera-scale images. International Journal of Pattern Recognition and Artificial Intelligence **33**, 1940012 (2019)
7. Gazagnes, S., Wilkinson, M.H.F.: Distributed connected component filtering and analysis in 2D and 3D tera-scale data sets. IEEE Transactions on Image Processing **30**, 3664–3675 (2021)
8. Götz, M., Cavallaro, G., Géraud, T., Book, M., Riedel, M.: Parallel computation of component trees on distributed memory machines. IEEE Transactions on Parallel and Distributed Systems **29**, 2582–2598 (2018)
9. Guigues, L., Cocquerez, J.P., Le Men, H.: Scale-sets image analysis. International Journal of Computer Vision **68**, 289–317 (2006)
10. Hartigan, J.A.: Statistical theory in clustering. Journal of Classification **2**, 63–76 (1985)
11. Jones, R.: Connected filtering and segmentation using component trees. Computer Vision and Image Understanding **75**, 215–228 (1999)
12. Kurtz, C., Naegel, B., Passat, N.: Connected filtering based on multivalued component-trees. IEEE Transactions on Image Processing **23**, 5152–5164 (2014)
13. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide-baseline stereo from maximally stable extremal regions. Image and Vision Computing **22**, 761–767 (2004)
14. Meyer, C., Baudrier, É., Schultz, P., Naegel, B.: Combining max-tree and CNN for segmentation of cellular FIB-SEM images. In: RRPR (2022)
15. Moschini, U., Meijster, A., Wilkinson, M.H.F.: A hybrid shared-memory parallel max-tree algorithm for extreme dynamic-range images. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**, 513–526 (2018)
16. Najman, L., Couprie, M.: Building the component tree in quasi-linear time. IEEE Transactions on Image Processing **15**, 3531–3539 (2006)
17. Passat, N., Mendes Forte, J., Kenmochi, Y.: Morphological hierarchies: A unifying framework with new trees. Journal of Mathematical Imaging and Vision **65**(5), 718–753 (2023)

18. Passat, N., Naegel, B.: Component-hypertrees for image segmentation. In: ISMM. pp. 284–295 (2011)
19. Passat, N., Naegel, N.: Component-trees and multivalued images: Structural properties. Journal of Mathematical Imaging and Vision **49**, 37–50 (2014)
20. Perret, B., Cousty, J., Tankyevych, O., Talbot, H., Passat, N.: Directed connected operators: Asymmetric hierarchies for image filtering and segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**, 1162–1176 (2015)
21. Perret, B., Lefèvre, S., Collet, C., Slezak, É.: Hyperconnections and hierarchical representations for grayscale and multiband image processing. IEEE Transactions on Image Processing **21**, 14–27 (2012)
22. Salembier, P., Oliveras, A., Garrido, L.: Anti-extensive connected operators for image and sequence processing. IEEE Transactions on Image Processing **7**, 555–570 (1998)
23. Salembier, P., Serra, J.: Flat zones filtering, connected operators, and filters by reconstruction. IEEE Transactions on Image Processing **4**, 1153–1160 (1995)
24. Wilkinson, M.H.F., Westenberg, M.A.: Shape preserving filament enhancement filtering. In: MICCAI. pp. 770–777 (2001)
25. Xu, Y., Géraud, T., Najman, L.: Connected filtering on tree-based shape-spaces. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**, 1126–1140 (2016)