Mask-Based Second-Generation Connectivity and Attribute Filters

Georgios K. Ouzounis and Michael H.F. Wilkinson, *Senior Member, IEEE*

Abstract—Connected filters are edge-preserving morphological operators, which rely on a notion of connectivity. This is usually the standard 4 and 8-connectivity, which is often too rigid since it cannot model generalized groupings such as object clusters or partitions. In the set-theoretical framework of connectivity, these groupings are modeled by the more general second-generation connectivity. In this paper, we present both an extension of this theory, and provide an efficient algorithm based on the Max-Tree to compute attribute filters based on these connectivities. We first look into the drawbacks of the existing framework that separates clustering and partitioning and is directly dependent on the properties of a preselected operator. We then propose a new type of second-generation connectivity termed *mask-based connectivity* which eliminates all previous dependencies and extends the ways the image domain can be connected. A previously developed Dual-Input Max-Tree algorithm for area openings is adapted for the wider class of attribute filters on images characterized by second-generation connectivity. CPU-times for the new algorithm are comparable to the original algorithm, typically deviating less than 10 percent either way.

Index Terms—Mathematical morphology, second-generation connectivity, connectivity class, clustering, partitioning, dual input max-tree, attribute filter.

1 Introduction

In discrete image analysis the set-theoretic concept of connectivity [1] describes the way pixels are grouped to form connected components or flat-zones in gray scale [2]. Connected components are image regions of constant intensity in which pixels are characterized by a path-wise connectedness relation. Typically, on the two-dimensional (2D) discrete space $\mathbb{Z}^2$ sets of pixels are either 4 or 8-connected [3], [4].

Based on the notion of connectivity, a family of morphological operators [5] known as *connected filters* [2], [6] has been developed which interact with the connected components rather than individual pixels. This prevents edge distortion, a property highly desirable in many applications. Connected components can either be removed or remain intact but new ones cannot emerge. Early members of this family were openings by reconstruction, for which efficient algorithms have been developed [7]. Furthermore, the concept of attribute filters [8] was introduced, which allows filtering based on the connected component attributes. Examples of this are attribute openings, closings, thinnings, and thickenings [6], [8], [9], [10].

In recent years, several theoretical developments concerning generalizations of the notion of connectivity have been presented [11], aiming to improve the robustness and increase the versatility of these filters. These generalizations aim at modeling object clusters and partitions in an edge preserving manner. A well-established approach known as *second-generation connectivity* [12], [13], [14] handles both cases independently by creating a "child" connectivity class, by using some operator. Second-generation connectivity can be classified as either *clustering* or *contraction*-based [14] depending on whether the operator expands, or contracts the original image. An example of filtering based on clustering connectivity is given in Fig. 1 illustrating an Anabaena complex. Assuming we target the largest complex, using standard 4 and 8-connectivity we can only retrieve the bigger fragment of the two, which are separated by the heterocyst—Fig. 1c. Instead, if the connected filter is defined on the more general clustering-based connectivity, the two fragments merge as illustrated in Fig. 1b, and the filter considers them as one object—Fig. 1d. The original image has been obtained from http://www.f-suiki.or.jp/suisitu/saikin/saikin.htm.

Second-generation connectivity is realized by means of a connectivity opening which is associated with a structural operator. The dependency on this operator imposes constraints as to how the image domain can be connected, and apart from clustering and partitioning, no further cases such as combinations of the two are supported. In our work, we counter this limitation by introducing a composite connectivity opening in which all dependencies to the structural operator are eliminated. Instead, we propose an association with a *connectivity mask* which is an image containing some arbitrary transformation of the original. This yields a single framework termed *mask-based connectivity* that accounts for all possible ways the image domain can be connected. This includes the two known cases of clustering and partitioning through the design of filters which yield results identical to the previous approach, even though formally based on different connectivity classes. The difference between these classes lies in the fact that, for a given operator, we generate a different connectivity class for each target image, rather than one, generally applicable connectivity class.

Algorithmic realizations of second-generation connectivity originally suggested the use of binary and gray-scale reconstruction operators for recovering the object clusters or partitions [14]. This introduced a family of filters based on

• *The authors are with the Institute for Mathematics and Computing Science, University of Groningen, PO Box 800, 9700 AV Groningen, The Netherlands. E-mail: m.h.f.wilkinson@rug.nl, georgios@cs.rug.nl.*

Manuscript received 19 May 2006; revised 22 Aug. 2006; accepted 11 Sept. 2006; published online 18 Jan. 2007.
Recommended for acceptance by P. Maragos.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0388-0506.
Digital Object Identifier no. 10.1109/TPAMI.2007.1045.

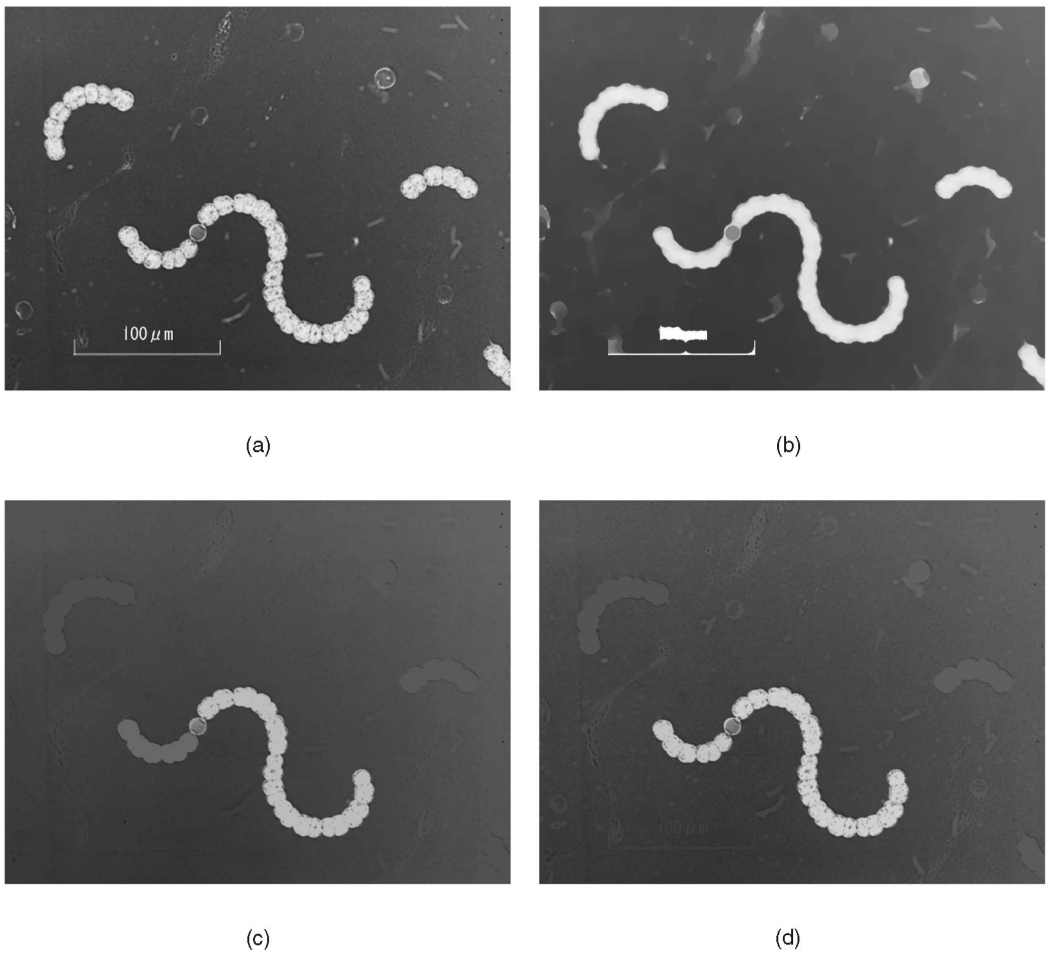


Fig. 1. Area opening using clustering-based connectivity: (a) original image, (b) the expanded set obtained by a structural closing, (c) the filtered image using an area criterion relying on the standard 4-connectivity, and (d) clustering-based connectivity.

width as the attribute criterion. An efficient algorithm for gray-scale area filters using second-generation connectivity has also been presented [15]. This method builds a hierarchical image representation based on gray-scale image pairs comprising the original image and its modified replica. In regions where the two images differ the algorithm evaluates the connectivity based on the properties of the structural operator and assigns pixels to the appropriate connected components. The algorithm which is inspired by Salembier et al. [16], is referred to as the Dual-Input Max-Tree algorithm [15] and supports both clustering and partitioning. In this work, we employ the Dual-Input Max-Tree modified for the more general family of gray-scale attribute filters on 2D and 3D data sets characterized by mask-based second-generation connectivity. We demonstrate its capacity on biomedical images and 3D data sets using nonincreasing shape filters based on moment invariants and provide the functionality to extend to other filter types.

The structure of this paper is organized as follows: In Section 2, a number of preliminaries is presented. These are the fundamental concepts of connectivity classes and connectivity openings, the notion of second-generation connectivity, and attribute filters. In Section 3, we investigate the drawbacks of the existing second-generation connectivity framework. Section 4 introduces the mask-based connectivity scheme and formalizes an expression of attribute openings associated to it. Following this, Section 5 gives a short introduction on the Max-Tree structure complemented by the description of the Dual-Input Max-Tree algorithm adopted for mask-based connectivity representation. Section 6 gives a number of examples on attribute filtering and a brief discussion on the results while conclusions are summarized in Section 7.

2 Theoretical Background

2.1 Connectivity Classes and Connectivity Openings

This section briefly outlines the concept of connectivity from the set-oriented morphological perspective. For the purpose of this analysis, we assume a universal (nonempty) set E and denote by $\mathcal{P}(E)$ the collection of all subsets of E.

Definition 1. *A family $\mathcal{C} \subseteq \mathcal{P}(E)$ for any arbitrary set E, is called a* connectivity class *if it satisfies:*

1. *$\emptyset \in \mathcal{C}$ and for all $x \in E$, $\{x\} \in \mathcal{C}$ and*
2. *for any $\{A_i\} \subseteq \mathcal{C}$ for which $\bigcap A_i \neq \emptyset \Rightarrow \bigcup A_i \in \mathcal{C}$.*

This means that both the empty set and singleton sets are connected and any union of sets which have a nonempty intersection is also connected. Members of $\mathcal{C}$ are called *connected sets* [1], [13], [17]. The family of all singleton sets is denoted by $\mathcal{S} \subseteq \mathcal{C}$.

Addressing connected regions in binary images is often more practical by means of *connected components* or *grains* C [6]. If C is a grain of X, we denote this $C \Subset X$. A connected component C of a binary image X is a connected set of maximal extent, in the sense that there is no set $C' \supset C$ such that $C' \subseteq X$ and $C' \in \mathcal{C}$.

Connected components are accessed by means of a *connectivity opening* Γ_x, which is an operator extracting the union of all connected sets within X that have a point $x \in E$ in their intersection, i.e.,

$$\Gamma_x(X) = \bigcup \{A_i \in \mathcal{C} \mid x \in A_i, A_i \subseteq X\} \tag{1}$$

for every $X \subseteq E$. From (1), it is trivial to show that Γ_x is an algebraic opening [18] marked by x, i.e., it is an increasing, antiextensive and idempotent operator. Furthermore, $\forall x \notin X$, $\Gamma_x(X) = \emptyset$.

Evidently connectivity classes and connectivity openings are interrelated. This is formally given by the following theorem [1], [12], [13].

Theorem 1. *The datum of a connectivity class $\mathcal{C}$ in $\mathcal{P}(E)$ is equivalent to the family $\{\Gamma_x | x \in E\}$ of openings on x such that:*

1. *Γ_x is an algebraic opening marked by $x \in E$,*
2. *for all $x \in E$, we have $\Gamma_x(\{x\}) = \{x\}$,*
3. *for all $X \subseteq E$, and all $x \in E$, we have $x \notin X \Rightarrow \Gamma_x(X) = \emptyset$, and*
4. *for all $X \subseteq E$, $x, y \in E$, if $\Gamma_x(X) \cap \Gamma_y(X) \neq \emptyset \Rightarrow \Gamma_x(X) = \Gamma_y(X)$, i.e., $\Gamma_x(X)$ and $\Gamma_y(X)$ are equal or disjoint.*

This shows that connectivity openings characterize uniquely the connectivity class with which they are associated and that there is a one-to-one correspondence between the two. We see this from two points of view [12]:

1. From the connectivity class to the system of connectivity openings: Γ_x is the union of all sets A in the connectivity class $\mathcal{C}$, such that $x \in A$ and $A \subseteq X$.
2. From the system of connectivity openings to the connectivity class $\mathcal{C}$: The connectivity class $\mathcal{C}$ is formed of all $\Gamma_x(X)$ for $x \in E$ and $X \subseteq E$.

Concluding, we see that to prove a family of sets is a connectivity class, it is sufficient to show that the operator extracting these sets is a connectivity opening satisfying the four conditions of Theorem 1.

2.2 Second-Generation Connectivity

Given a connectivity class $\mathcal{C}$, it is possible to generate a *child* class with either reduced or enriched members by modifying its associated connectivity opening. This is referred to as *second-generation connectivity* [12], [13], [17] and aims at modeling object clusters or partitions that cannot be captured otherwise.

Each of the two cases is defined separately and we identify second-generation connectivity as either *clustering* or *contraction*-based [14]. In both cases, the connectivity openings Γ_x^ψ of the family associated to the child connectivity class are given in an expression dependent on a structural operator ψ such as a dilation, a closing, or an opening.

2.2.1 Clustering-Based Connectivity

The first case of second-generation connectivity describes groups of image objects that can be perceived as clusters of connected components if their relative distances are below a given threshold. This is controlled by the size of the structuring element used along with an operator ψ termed *clustering* [12], [13], [14]. Following is a list summarizing the properties required to define a clustering:

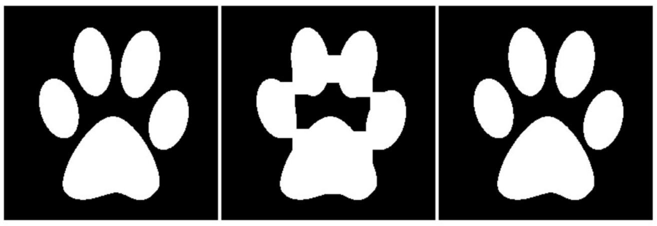

(a) (b) (c)

Fig. 2. Clustering Sets: (a) The original image X illustrates five separate objects which expanded by ψ yield (b) the sets making up the cluster. (c) By intersecting the connected components of $\psi(X)$ with X, the operator $\Gamma_x^\psi(X)$ extracts the cluster of the previously disconnected objects.

1. ψ is increasing and extensive.
2. $\psi(\mathcal{C}) \subseteq \mathcal{C}$.
3. For a family $\{X_i\}$ in $\mathcal{P}(E)$ such that $\psi(X_i) \in \mathcal{C}$, $\forall i$, and $\bigcap_i X_i \neq \emptyset \Rightarrow \psi(\bigcup X_i) \in \mathcal{C}$.
4. ψ does not create connected components; i.e., if $\forall x \in C$ and $C \Subset \psi(X) \Rightarrow X \cap C \neq \emptyset$.
5. ψ treats the clusters of X independently; i.e., if $\forall x \in C$ and $C \Subset \psi(X) \Rightarrow \psi(X \cap C) = C$.

Further analysis of each item is given in [14].

If the operator ψ satisfies the first three properties, it is referred to as a *weak clustering* or simply clustering.

Definition 2. *Let $\mathcal{C}$ be a connectivity class in $\mathcal{P}(E)$ and ψ be a clustering operator on $\mathcal{P}(E)$. Then,*

$$\mathcal{C}^\psi = \{X \in \mathcal{P}(E) \mid \psi(X) \in \mathcal{C}\} \tag{2}$$

is a clustering-based *connectivity class with*

$$\mathcal{C} \subseteq \mathcal{C}^\psi. \tag{3}$$

Operator ψ is a *strong clustering* if and only if it satisfies all five properties above. Typical examples of clustering operators are certain extensive dilations and closings, using connected structuring elements.

Definition 3. *Let $\{\Gamma_x \mid x \in E\}$ be the connectivity openings associated with $\mathcal{C}$. If ψ is a strong clustering on $\mathcal{P}(E)$, the family of connectivity openings $\{\Gamma_x^\psi \mid x \in E\}$ associated to $\mathcal{C}^\psi$ are given by*

$$\Gamma_x^\psi(X) = \begin{cases} \Gamma_x(\psi(X)) \cap X, & \text{if } x \in X \quad \text{(4a)} \\ \emptyset, & \text{otherwise.} \quad \text{(4b)} \end{cases}$$

An example of clustering sets is illustrated in Fig. 2a which shows a set of five individual connected components and Fig. 2b, an expanded replica by a structural closing. The connectivity opening of (4) for $(x, y) = (128, 200)$ extracts the cluster of the connected components, as illustrated Fig. 2c.

2.2.2 Contraction-Based Connectivity

The second case is a partitioning scheme in which wide object regions bridged in the original image by narrow elongated structures can be treated as separate objects [12], [14], [18]. The "narrowness" of these structures is determined by the

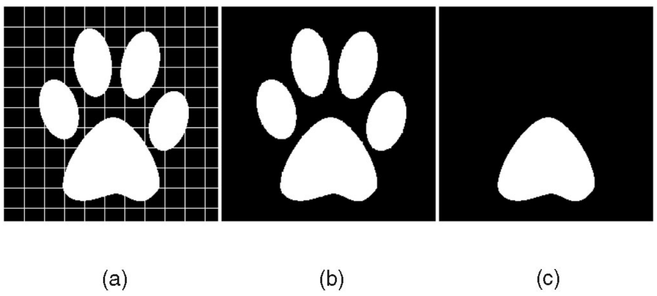


Fig. 3. Partitioning Sets: (a) The original image containing a single connected component, (b) the set of stable components given by $\psi(X)$, and (c) an independent connected component previously connected to the grid.

size the structuring element used along with an increasing and antiextensive operator ψ on $\mathcal{P}(E)$, called a *contraction*. Furthermore, any set $X \subseteq E$ which is invariant to ψ is called *stable*, i.e., $\psi(X) = X$.

Restricting the original connectivity class $\mathcal{C}$ by turning all connected members that are not invariant to ψ to connected singleton sets, yields a child connectivity class defined as follows:

Definition 4. *Let $\mathcal{C}$ be a connectivity class in $\mathcal{P}(E)$ and ψ be a contraction on $\mathcal{P}(E)$. Then,*

$$\mathcal{C}^{\psi} = \{\emptyset\} \cup \mathcal{S} \cup \{X \in \mathcal{C} \mid \psi(X) = X\} \tag{5}$$

is a contraction-based *connectivity class with*

$$\mathcal{C}^{\psi} \subseteq \mathcal{C}. \tag{6}$$

Contractions are typically structural openings. A necessary condition to define the family of connectivity openings associated with $\mathcal{C}^{\psi}$ is for ψ to be locally invariant with respect to $\mathcal{C}$ for any $X \subseteq E$ [14], i.e.,

$$\psi(X) = X \Rightarrow \psi(\Gamma_x(X)) = \Gamma_x(X),\ \forall\, x \in E. \tag{7}$$

This means that for any set X invariant to ψ, all connected components of X must also be invariant to ψ. An example of a locally invariant opening is a structural opening with a connected structuring element.

Definition 5. *Let $\{\Gamma_x \mid x \in E\}$ be the connectivity openings associated with $\mathcal{C}$. If ψ is an opening on $\mathcal{P}(E)$ locally invariant with respect to $\mathcal{C}$, the family of connectivity openings $\{\Gamma_x^{\psi} \mid x \in E\}$ associated to $\mathcal{C}^{\psi}$ are given by*

$$\Gamma_x^{\psi}(X) = \begin{cases} \Gamma_x(\psi(X)) & \text{if } x \in \psi(X) \quad (8a)\\ \{x\} & \text{if } x \in X \setminus \psi(X) \quad (8b)\\ \emptyset & \text{otherwise.} \quad (8c)\end{cases}$$

Partitioning an image with this scheme does not modify the existing edges and the union of all stable sets with the singletons that complement them yields back the original image. An example is illustrated in Fig. 3 where there exists a single connected component which we supposingly would like to handle as five separate objects disconnected from the grid. The grid, in this example, represents a background object connecting the objects of interest in a nondesirable way. Applying a structural opening ψ on X removes the grid and yields the set of all the components invariant to ψ, termed *stable* by [14] (Fig. 3b). Elements of the grid removed by ψ are treated as singletons in $\mathcal{C}^{\psi}$. Applying $\Gamma_x^{\psi}(X)$ will extract each of the five objects seen in Fig. 3b separately (Fig. 3c for $(x, y) = (128, 200)$).

2.3 Attribute Openings

Binary attribute openings [8] are a family of connected filters [2], [6] that incorporate a trivial opening Γ_Λ on the output of a connectivity opening Γ_x. The trivial opening accepts or rejects connected components subject to an increasing and shift invariant criterion Λ given by:

$$\Lambda(C) = (Attr(C) \geq \lambda) \tag{9}$$

with $Attr(C)$ some real-value attribute of the connected component C and λ an attribute threshold. The increasingness of Λ implies that, if a set A satisfies Λ, then any set B such that $B \supseteq A$ satisfies Λ as well. We can summarize the definition of Γ_Λ to the following $\Gamma_\Lambda : \mathcal{C} \to \mathcal{C}$ operating on $C \in \mathcal{C}$ yields C if $\Lambda(C)$ is true, and $\emptyset$ otherwise. Furthermore, $\Gamma_\Lambda(\emptyset) = \emptyset$. The binary attribute opening can be defined as follows:

Definition 6. *The binary attribute opening Γ^{Λ} of a set X with increasing criterion Λ is given by:*

$$\Gamma^{\Lambda}(X) = \bigcup_{x \in X} \Gamma_\Lambda(\Gamma_x(X)). \tag{10}$$

An example is the area opening [19]. Note that, if Λ is nonincreasing, we have an *attribute thinning* [8] or *nonincreasing grain filter* [6] rather than an attribute opening. An example is the scale-invariant elongation criterion in which $Attr(C) = I(C)/A^2(C)$, with $I(C)$ the moment of inertia of C and $A(C)$ the area [10], [20], [21]. For the volume set that we demonstrate later on, the moment of inertia is given by

$$I(C) = \frac{V(C)}{4} + \sum_{\mathbf{x} \in C} (\mathbf{x} - \overline{\mathbf{x}})^2 \tag{11}$$

in which $V(C)$ denotes the volume of C, and the elongation (or noncompactness) [21] is measured by the ratio

$$Attr(C) = \frac{I(C)}{V^{5/3}(C)}. \tag{12}$$

$Attr(C)$ has a minimum for a sphere (0.23) and increases rapidly with elongation.

3 Drawbacks of Conventional Second-Generation Connectivity

This section summarizes some drawbacks of second-generation connectivity due to the dependency of the associated connectivity openings on the properties of the structural operators used along with them. The objective is to demonstrate that there exist useful structural operators that are excluded because they do not comply with the requirements listed in Section 2.2.

The connectivity openings associated with the two cases of second-generation connectivity reviewed in Section 2.2 can be summarized in a unified formalism given by:

$$\Gamma_x^{\psi}(X) = \begin{cases} \Gamma_x(\psi(X)) \cap X & \text{if } x \in X \cap \psi(X) \quad (13a)\\ \{x\} & \text{if } x \in X \setminus \psi(X) \quad (13b)\\ \emptyset & \text{otherwise.} \quad (13c)\end{cases}$$

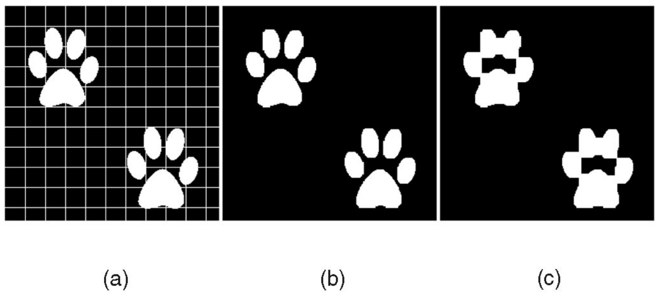


Fig. 4. Clustering Partitioned Sets: (a) original image X, (b) the contracted set by ψ_p, and (c) $\psi_{cp}(X)$: expanding the contractions by ψ_c on $\psi_p(X)$.

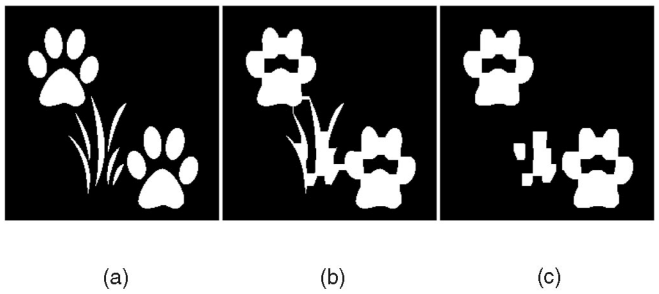


Fig. 5. Partitioning Clustered Sets: (a) original image X, (b) the expanded set by ψ_c, and (c) $\psi_{pc}(X)$: the contraction by ψ_p on the expanded set $\psi_c(X)$.

It is evident that, if ψ is a strong clustering (extensive), the situation in which an element $x \in X \setminus \psi(X)$ cannot occur, therefore (13) simplifies to (4). If ψ is a contraction (antiextensive), then for $x \in X \cap \psi(X)$, the corresponding term in (13) simplifies to $\Gamma_x(\psi(X))$ due to the antiextensivity of ψ and, hence, we obtain (8).

Merging the two cases of clustering and partitioning as operations in a single expression unifies implementations of second-generation connectivity like in [15]. The connectivity opening of (13) also allows a number of other structural operators to be used, but because they violate the properties described in Section 2.2, they prevent obtaining a valid connectivity class.

Typical examples are the *alternating-sequential filters* or *AS-filters* [1], [22], [23] which are excluded since they are neither extensive nor antiextensive operators. AS-filters modify the original image by introducing regions that appear as the result of local clustering or partitioning. They are defined as either:

$$\psi_{cp} = \psi_c \psi_p \tag{14}$$

or

$$\psi_{pc} = \psi_p \psi_c, \tag{15}$$

where ψ_c and ψ_p are closings and openings, respectively, by a connected structuring element. Examples of each case are illustrated in Fig. 4 and Fig. 5. When clustering contracted sets, first the input image is contracted to disconnect objects from thin elongated structures like the grid in Fig. 4a which is no longer present in the second. If the resulting objects are to be treated as groupings a further clustering operator is applied which merges the neighboring connected components to the desired clusters as Fig. 4c. By contrast, when partitioning expanded sets, we aim at disconnecting the expanded objects as in Fig. 5b from unwanted narrow structures by applying a contracting operator. The grid as in Fig. 3 represents a background object.

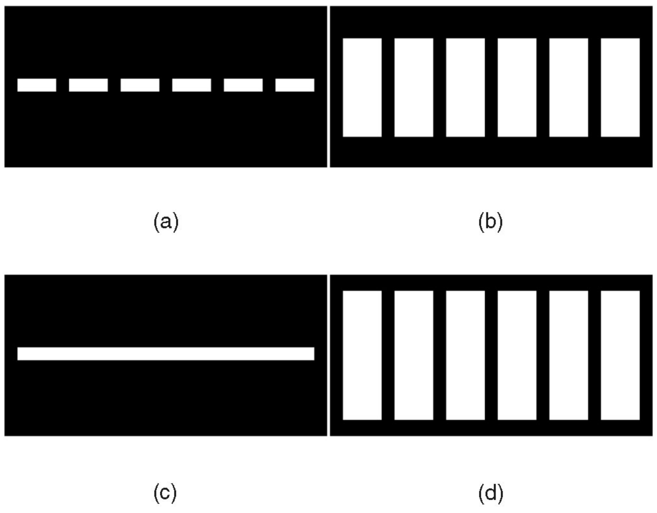


Fig. 6. An example of an extensive and nonincreasing operator (adaptive Minkowski addition) that violates increasingness of the connected openings; (a) original image X, (b) image $Y \supseteq X$, (c) $\psi(X)$, and (d) $\psi(Y)$. Note that the connected components are expanded along the direction of elongation.

An example of operators violating the increasingness property are directional Minkowski additions which perform a maximum operation along the direction of elongation using adaptive structural elements. They are nonincreasing extensive operators; hence, for $X \subseteq Y$, we may obtain $\psi(X) \not\subseteq \psi(Y)$ (see Fig. 6). For an element x with $x \in X \cap \psi(X)$, $x \notin Y \cap \psi(Y)$, therefore $\Gamma_x^{\psi}(X) \not\subseteq \Gamma_x^{\psi}(Y)$. This implies that Γ_x^{ψ} is nonincreasing, hence, not an algebraic opening.

A typical case of an increasing, antiextensive, and nonidempotent operator is that of erosions. Erosions violate the idempotence of the contraction-based connectivity opening which requires that

$$\Gamma_s^{\psi}(\Gamma_x^{\psi}(X)) = \Gamma_x^{\psi}(X). \tag{16}$$

For the case that $x \in X \cap \psi(X)$ according to (13) this is:

$$\Gamma_x^{\psi}(\Gamma_x^{\psi}(X)) = \Gamma_x^{\psi}(\Gamma_x(\psi(X)) \cap X). \tag{17}$$

This is equivalent to $\Gamma_x^{\psi}(\Gamma_x(\psi(X)))$ due to the antiextensivity of ψ and can be written as:

$$\Gamma_x^{\psi}(\Gamma_x^{\psi}(X)) = \Gamma_x(\psi(\Gamma_x(\psi(X)))) \neq \Gamma_x(\psi(X)), \tag{18}$$

i.e., erosions are not valid operators.

All three cases demonstrate incompatibility with the existing second-generation connectivity framework by violating the properties of the connectivity opening Γ_x^{ψ} through the properties of ψ. In Section 6.2, we show practical examples of the first two cases.

4 Mask-Based Second-Generation Connectivity

4.1 Mask-Based Connectivity

The limitations on the nature of the second-generation connectivity to clustering or partitioning only can be eliminated if the associated family of connectivity openings is no longer dependent on a structural operator. For this purpose we present an alternative scheme in which we associate the connectivity openings to the resulting image of some arbitrary transformation on X. We call this the *connectivity mask* and since we initially make no assumptions as to how it is created, we denote it as a generic set $M \subseteq E$. The key notion is that we only apply the modifying operator ψ to image X once to obtain mask M, whereas the operator-based framework applies ψ each time a filter is applied which uses connectivity based on $\mathcal{C}^{\psi}$. Thus, if we were to compute a granulometry based on attribute filters [8] within this framework, mask M would be precomputed and all subsequent openings performed using the same M. Based on M, the connectivity openings $\{\Gamma_x^M \mid x \in E\}$ "mask" the desired members of $\mathcal{C}$ to the child class $\mathcal{C}^M$. Apart from the family of the singleton sets and the empty set which are essential in the definition of a connectivity class, the members of $\mathcal{C}^M$ can be summarized to all subsets A of the universal set E that are included in some grain of M, denoted as $\Gamma_x(M)$. This is formalized accordingly:

Definition 7. *Let $\mathcal{C} \subseteq \mathcal{P}(E)$ be a connectivity class and $M \subseteq E$ a connectivity mask for an image X. The* mask-based *second-generation connectivity class $\mathcal{C}^M$ is given by:*

$$\mathcal{C}^M = \{\emptyset\} \cup \mathcal{S} \cup \{A \subseteq E \mid \exists\, x \in E : A \subseteq \Gamma_x(M)\}. \tag{19}$$

Inspired by (13), we propose an association of $\mathcal{C}^M$ with a family of connectivity openings $\{\Gamma_x^M \mid x \in E\}$ as follows:

Proposition 1. *Let $\mathcal{C}$ be a connectivity class in $\mathcal{P}(E)$ and $X, M \subseteq \mathcal{C}$ be the original image and the connectivity mask, respectively.*

1. *Then, the operator*

$$\Gamma_x^M(X) = \begin{cases} \Gamma_x(M) \cap X & \text{if } x \in X \cap M & (20a) \\ \{x\} & \text{if } x \in X \setminus M & (20b) \\ \emptyset & \text{otherwise} & (20c) \end{cases}$$

extracting subsets of X found within the grains of M is a connectivity opening and

2. *the family $\{\Gamma_x^M \mid x \in E\}$ is associated to $\mathcal{C}^M$.*

Proof.

1. To prove this proposition, we must show that $\Gamma_x^M(X)$ meets the requirements of Theorem 1. First, we show that it is an algebraic opening, i.e., it is an antiextensive, increasing, and idempotent operator.

   Antiextensivity is trivial since for all three cases of (20) $\Gamma_x^M(X) \subseteq X$. Increasingness requires if $X \subseteq Y \Rightarrow \Gamma_x^M(X) \subseteq \Gamma_x^M(Y)$. We can identify two important cases: 1) $x \notin X$ and 2) $x \in X$. In the first case, $\Gamma_x^M X = \emptyset$ so whichever case of (20) holds for $\Gamma_x^M(Y)$, we have $\Gamma_x^M(X) \subseteq \Gamma_x^M(Y)$. In the second case, $x \in Y$ because $X \subseteq Y$. Again, we identify two cases: 1) $x \in M$ and 2) $x \notin M$ corresponding to (20a) and (20b), respectively. If $x \in M$, we have

$$\Gamma_x^M(X) = \Gamma_x(M) \cap X \tag{21}$$

and

$$\Gamma_x^M(Y) = \Gamma_x(M) \cap Y. \tag{22}$$

Obviously, if $X \subseteq Y$, then $\Gamma_x(M) \cap X \subseteq \Gamma_x(M) \cap Y$ and, therefore, we have $\Gamma_x^M(X) \subseteq \Gamma_x^M(Y)$. Finally, if $x \notin M$, we have

$$\Gamma_x^M(X) = \Gamma_x^M(Y) = \{x\}, \tag{23}$$

so that $\Gamma_x^M(X) \subseteq \Gamma_x^M(Y)$ holds in all three cases of (20). For idempotence, we require that

$$\Gamma_x^M(\Gamma_x^M(X)) = \Gamma_x^M(X). \tag{24}$$

Again, we treat the three cases of (20) separately. The simplest is the case (20c), in which $\Gamma_x^M(X) = \emptyset$. Because of antiextensivity $\Gamma_x^M(\emptyset) = \emptyset$, so in this case (24) holds. In the case of (20b), we have $\Gamma_x^M(X) = \{x\}$. Obviously, $x \in \{x\} \setminus M$, so that in $\Gamma_x^M(\{x\})$, the case of (20b) applies again, and (24) holds. Finally, if $x \in X \cap M$, we have

$$\Gamma_x^M(X) = \Gamma_x(M) \cap X. \tag{25}$$

In this case $x \in \Gamma_x^M(X) \cap M$, so (20a) applies:

$$\begin{aligned} \Gamma_x^M(\Gamma_x^M(X)) &= \Gamma_x(M) \cap \Gamma_x^M(X) \\ &= \Gamma_x(M) \cap \Gamma_x(M) \cap X \\ &= \Gamma_x(M) \cap X = \Gamma_x^M(X) \end{aligned} \tag{26}$$

and, therefore, idempotence holds in all three cases. Note that no restriction is placed on M, i.e., $\Gamma_x^M(X)$ is an algebraic opening for any $M \subseteq E$.

The second requirement of Theorem 1 states that $\Gamma_x^M(\{x\}) = \{x\}$, $\forall x \in E$. In the case where $x \in X \cap M$, $\Gamma_x^M(\{x\}) = \Gamma_x(\{x\}) \cap M = \{x\}$ for whatever M. Similarly, if $x = \{x\} \setminus M$, $\Gamma_x^M(\{x\}) = \{x\}$ from (20b). The third requirement states that $\forall X \subseteq E$, and $\forall x \in E$, if $x \notin X \Rightarrow \Gamma_x(X) = \emptyset$. From (20c), we see that, if $x \notin X \Rightarrow x \in \emptyset$, therefore $\Gamma_x^M(X) = \emptyset$.

To prove the fourth requirement of Theorem 1, we require that for any x, $y \in E$, the connected components returned by the connectivity opening which are marked by x and y are either equal or disjoint, i.e., $\Gamma_x^M(X) \cap \Gamma_y^M(X) = \emptyset$ or $\Gamma_x^M(X) = \Gamma_y^M(X)$. We identify four cases:

   1. If $x, y \in X \cap M$, then $\Gamma_x^M(X)$ and $\Gamma_y^M(X)$ are equal or disjoint, because $\Gamma_x(M)$ and $\Gamma_y(M)$ are equal or disjoint by the definition of connectivity openings.
   2. If $x, y \in X \setminus M$, then $\Gamma_x^M(X) = \{x\}$ and $\Gamma_y^M(X) = \{y\}$, which are equal or disjoint.
   3. If either x or $y \notin X$, then $\Gamma_x^M(X) \cap \Gamma_y^M(X) = \emptyset$.
   4. If $x \in X \cap M$ and $y \in X \setminus M$, then $\Gamma_x^M(X) \cap \Gamma_y^M(X) = \emptyset$ because $(X \cap M) \cap (X \setminus M) = \emptyset$, i.e., the two connected components are defined over disjoint partitions of X.

2. So far, we showed that $\Gamma_x^M(X)$ satisfies Theorem 1, hence the family $\{\Gamma_x^M \mid x \in E\}$ is associated with a connectivity class. Now, we verify that this is $\mathcal{C}^M$ according to Definition 7.

A connectivity class is equivalent to the union over all $x \in E$ of the invariance domains of the associated connectivity openings [6], [12]. The invariance domain of Γ_x^M contains besides the empty set, all connected sets in $\mathcal{C}^M$ that contain x, i.e.,

$$\begin{aligned}\mathrm{Inv}(\Gamma_x^M) = \{\emptyset\} &\cup \{\{x\}\} \\ &\cup \{A \\ &\subseteq E \mid x \in A, A \subseteq \Gamma_x(M)\},\end{aligned} \tag{27}$$

for each $x \in E$. By $\{\{x\}\}$ we denote the set containing the singleton set $\{x\}$, to distinguish the two. The first term is trivial since the empty set is invariant to every Γ_x^M, i.e., $\Gamma_x^M(\emptyset) = \emptyset$. The second is included because Γ_x^M is a connectivity opening so that $\Gamma_x^M(\{x\}) = \{x\}$ for all $x \in E$. The last term states that $\Gamma_x^M(A) = A$ if $x \in A$ and $A \subseteq \Gamma_x(M)$, which follows from (20a). This readily shows that $\mathrm{Inv}(\Gamma_x^M) \subseteq \mathcal{C}^M$, from (19) and, therefore,

$$\bigcup_{x\in E} \mathrm{Inv}(\Gamma_x^M) \subseteq \mathcal{C}^M. \tag{28}$$

We will now show that

$$\mathcal{C}^M \subseteq \bigcup_{x\in E} \mathrm{Inv}(\Gamma_x^M), \tag{29}$$

by proving that any element of $\mathcal{C}^M$ is included in the right-hand side of (29). Obviously, this holds for $\emptyset$. We now verify that for any nonempty element $C \in \mathcal{C}^M$, $C \in \mathrm{Inv}(\Gamma_x^M)$ for all $x \in C$. If C is a singleton, this is obvious because Γ_x^M is a connectivity opening (Property 2, Theorem 1). Otherwise, C is subset of a grain of M, i.e., for any $x \in C$ we have $C \subseteq \Gamma_x(M)$, and $C \in \mathrm{Inv}(\Gamma_x^M)$ for all $x \in C$. Thus, every element of $\mathcal{C}^M$ is contained in the union of the invariance domains of operators from the family $\{\Gamma_x^M \mid x \in E\}$. Therefore, (29) is true and, with (28), we have

$$\bigcup_{x\in E} \mathrm{Inv}(\Gamma_x^M) = \mathcal{C}^M. \tag{30}$$

□

The connectivity opening of (20) can be used for both clustering or partitioning by generating the connectivity mask with an appropriate operator ψ from X. From (13) and (20), it is obvious that connectivity openings $\Gamma_x^\psi(X) = \Gamma_x^M(X)$, if $M = \psi(X)$ for any $x \in E$ and any $X \in \mathcal{P}(E)$, and that the resulting attribute filters will yield the same result. However, the connectivity classes are not the same. Consequently, the connectivity class $\mathcal{C}^M$ cannot be expressed explicitly as a superset or subset of the original $\mathcal{C}$.

4.2 Gray-Scale Mask-Based Attribute Filters

Attribute openings as mentioned earlier apply a trivial opening Γ_Λ on the output of a binary connectivity opening Γ_x. To associate these operators with the connected sets of a second-generation connectivity class, we replace Γ_x with the corresponding connectivity opening [15], which in the case of mask-based connectivity is Γ_x^M. The increasingness of Γ_Λ and Γ_x^M makes it possible to extend Γ_M^Λ directly to gray scale by the principle of *threshold superposition* [24]. Superimposing threshold sets requires their hierarchical nesting along the gray scale. Given that a gray-scale image f can be decomposed to a set of binary images $T_h(f)$ resulting from thresholding f at all levels $h \in [0, N-1]$, i.e.,

$$T_h(f) = \{x \in E \mid f(x) \geq h\}, \tag{31}$$

the nesting of $T_h(f)$ within $T_k(f)$ is trivial for any $h \geq k$. In operator-based second-generation connectivity with ψ being any of the increasing structural operators described in Section 2.2, the nesting of $\psi(T_h(f)) \subseteq \psi(T_k(f))$ is obvious since the connectivity class $\mathcal{C}^\psi$ applies to all threshold sets [25]. In mask-based connectivity, however, this is not the case; a different mask applies to every threshold set and, therefore, a different connectivity class.

Let $m = \phi(f)$ be a gray-scale connectivity mask where ϕ is an arbitrary operator, with $M_h = T_h(m)$ denoting each threshold set of m. Obviously, $M_h \subseteq M_k$ for any $h \geq k$ (even if ϕ, is nonincreasing). At each level h, the family of connectivity openings given in Proposition 1 yields a set of connected components according to $\mathcal{C}^{M_h}$. Because any connected component P_h^i of $T_h(f)$ must remain connected at lower gray levels (see [25] for details), we require that $\mathcal{C}^{M_h} \subseteq \mathcal{C}^{M_k}$ for any $h \geq k$. Since $M_h \subseteq M_k$, all we need to prove is the following proposition:

Proposition 2. *For any two mask-based connectivity classes $\mathcal{C}^M$ and $\mathcal{C}^L$ associated to connectivity masks $M \subseteq L \subseteq E$, the following property holds:*

$$\mathcal{C}^M \subseteq \mathcal{C}^L. \tag{32}$$

Proof. Comparing two mask-based connectivity classes as given in (19) is by looking at the nesting of the grains. For two connectivity masks such that $M \subseteq L$, the nesting implies that:

$$\forall\, C_i^M\ \exists\, j\ :\ C_i^M \subseteq C_j^L \tag{33}$$

in which C_i^M and C_j^L are connected components of M and L, respectively. Therefore, for any set $A \subseteq E$:

$$A \subseteq C_i^M\ \Rightarrow\ A \subseteq C_j^L. \tag{34}$$

Therefore, if $A \in \mathcal{C}^M$, this implies $A \in \mathcal{C}^L$, i.e., $\mathcal{C}^M \subseteq \mathcal{C}^L$.□

Superimposing the outputs of the filtered threshold sets can be summarized to:

Definition 8. *For a mapping $f : E \to \mathbb{R}$, the gray-scale mask-based attribute opening $\gamma_m^\Lambda(f)$ is given by:*

$$(\gamma_m^\Lambda(f))(x) = \sup\{h \mid x \in \Gamma_\Lambda(\Gamma_x^{T_h(m)}(T_h(f)))\}. \tag{35}$$

Thus, the mask-based attribute opening of a gray-scale image assigns each point of the original image the highest threshold at which it still belongs to a connected foreground component. Similarly, using a nonincreasing criterion Λ, we can define the mask-based attribute thinnings.

5 Computing Second-Generation Attribute Filters

5.1 The Max-Tree Algorithm

The Max-Tree is a hierarchical image representation algorithm introduced by Salembier et al. [16] in the context of antiextensive attribute filtering. The tree structure encodes the set-theoretical notion of connectivity and its gray-scale extension within the nesting properties of the level components which are represented by nodes. It resembles to a certain extent the *Component Tree* by Jones [26] and its derivative *Gray-scale Component Tree* by Braga-Neto and Goutsias [25] where the nodes of the first correspond to level-sets while those of the second to gray-scale images. Connectivity at multiple scales is modeled by the *Connectivity Tree* of Tzafestas and Maragos [27], but can only handle binary images. Our work on second-generation connectivity representation and filtering is based on the Max-Tree primarily due to the algorithm's ability to handle nonincreasing attributes on gray-scale images at rather low computational time.

The Max-Tree nodes correspond to connected components or sets of flat zones and there exists a unique mapping to *peak components*. A peak component P_h at level h is a connected component of the thresholded image $T_h(f)$ and a *regional maximum* M_h at level h is a level component no members of which have neighbors of intensity larger that h. The regional maxima in this case correspond to the leaves of the tree. Each tree node C_h^k (k is the node index) corresponding to a certain peak component contains only those pixels in P_h^k which have gray-level h. In addition each node except for the root, points toward its parent $C_{h'}^{k'}$ with $h' < h$. The root node is defined at the minimum level $h_{\min}$ and represents the set of pixels belonging to the background.

The attributes of the connected components are computed during the construction phase of the tree and stored within the corresponding node structure. The attributes can be either increasing or nonincreasing such as area/volume or shape descriptors such as moment invariants, respectively. In both cases, the peak component k at level h inherits the attribute data of all the peak components $P_{h'}^k$ connected to C_h^k at levels $h' > h$. Thus, computing an attribute filter reduces to removing all nodes with attribute value smaller than a given threshold λ from the tree. Note that the node filtering is a separate stage from the computation of attributes and connected component analysis [16] and consumes only a short fraction of the total computation time.

5.2 The Dual-Input Max-Tree Algorithm

The Dual-Input Max-Tree algorithm presented in this section operates like the conventional Max-Tree [28] only it requires two input images; the original image X and the connectivity mask M according to Proposition 1. The tree is constructed in a recursive manner from data retrieved from a set of hierarchical first in-first out (FIFO) queues. The queues are allocated at initialization in the form of a static array called $HQueue$ segmented to a number of entries equal to the number of gray levels in the connectivity mask. Data are accessed and stored in each queue entry by the flooding function (Fig. 7) which reassigns priority pixels to the Max-Tree structure and stores new pixels retrieved from the neighborhood of the one under study, to the appropriate queue entries.

The Max-Tree structure consists of nodes corresponding to pixels of a given peak component P_h^k at level h. Each node is characterized by its level h and index k and contains information about its parent node id, the node status, and the attribute value (note that the tree structure is shaped by the histogram of the original image).

The two structures are managed with the aid of three arrays; the $STATUS[p]$, the $NumberOfNodes[h]$, and the $NodeAtLevel[h]$. $STATUS$ is an array of image size that keeps track of the pixel status. A pixel p can either be $NotAnalyzed$, $InTheQueue$, or already assigned to node k at level h. In this case, $STATUS[p] = k$. The $NumberOfNodes$ is an array that stores the number of nodes created until that moment at level h. Last, $NodeAtLevel$ is a Boolean array that flags the presence of a node still being flooded at level h.

5.2.1 Initialization

During initialization, the status of all image pixels is set to $NotAnalyzed$. Similarly, the $NumberOfNodes$ is set to zero, while $NodeAtLevel$ is set to $FALSE$ for each gray level. The histograms of both images are then computed to shape the $HQueue$ and Max-Tree structures accordingly. The first pixel with the lowest intensity $h_{\min}$ in M, retrieved during the histogram computation, is placed in the corresponding queue while the three arrays are updated. This pixel defines the root node and is passed on to the main routine (flood) as the starting element.

5.2.2 The Flooding Function

The flooding routine is a recursive function involved in the construction phase of the Max-Tree. It is initiated by accessing the first root pixel from the queue at level $h_{\min}$ and proceeds with flooding nodes along the different root paths that emerge during this process. The pseudocode in Fig. 7 describes in detail the steps involved. ORI and P_ORI are two image-size arrays containing the pixel intensities in the original image and connectivity mask, respectively.

Calling this function for a given level h, it first initializes an attribute variable `attr` which is updated for every pixel at level h in both ORI and P_ORI. An inspection on pixel availability for the given queue entry at level h proceeds by retrieving the first available pixel and continues with flooding, otherwise it skips flooding and finalizes the current node. If a pixel is available, its status is updated to the current node index for the level at $ORI[p]$ and its coordinates are computed. The process until this instance is identical to the conventional Max-Tree algorithm. The Dual Input Max-Tree upon retrieving a pixel inspects for an intensity mismatch between the ORI and P_ORI entries. If $ORI[p] < P\_ORI[p]$, where p is the pixel under study, the connectivity mask involves local clustering, while if $ORI[p] > P\_ORI[p]$, it involves local partitioning.

The first case implies that p is a background pixel in the original image therefore it is regarded as connected to the current active node at level $ORI[p]$ through the connected component at level $P\_ORI[p]$, i.e., it defines a peak component at level $ORI[p]$ to which p in the modified image is connected. $NodeAtLevel[ORI[p]]$ is set and a subroutine inspects for a node allocated for that level. If not found, a node is created and its attribute is initialized otherwise we simply update the existing node attribute.

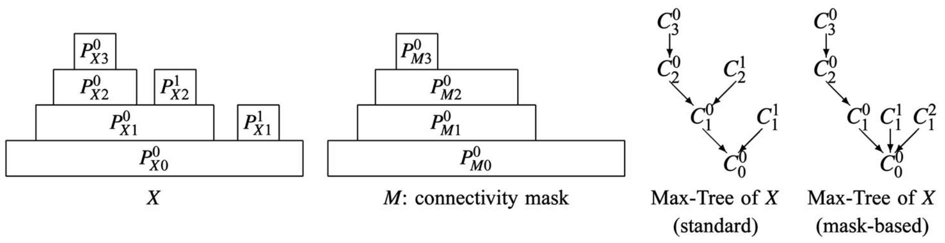
In the second case where partitioning is involved, we have $P\_ORI[p] < ORI[p]$. Pixel p is therefore part of a discarded component according to Definition 1 and, consequently, is treated as a singleton. Singletons define a node of unit attribute at level $ORI[p]$ hence upon detection the node must be finalized before retrieving the next priority pixel from the corresponding queue at level $P\_ORI[p]$. This involves setting the node status to the node index at level $ORI[p]$ and

```
flood(h, thisAttribute) {                           /* Flooding function at level h                      */
  attr = InitializeAttribute()                      /* Initialize attr                                   */
  if(thisAttribute)                                 /* Accounts for child attributes                     */
     MergeAuxData(attr,thisAttribute)

  while (not HQueue-empty(h))                       /* First step: propagation                           */
  { p = HQueue-first(h)                             /* Retrieve priority pixel                           */
    STATUS[p] = NumberOfNodes[ORI[p]]               /* STATUS = the node index                           */
    x = x_coord_of_p                                /* Retrieve x, y, z coordinates of p                 */
    y = y_coord_of_p
    z = z_coord_of_p

    if(ORI[p]!=h){                                  /* Detect intensity mismatch                         */
        NodeAtLevel[ORI[p]]=TRUE                    /* Same for both cases                               */
        idx = NodeOffsetAtLevel[ORI[p]]             /* Get the parent node offset                        */
        if(Tree[idx]==NULL)                         /* Check if node exists, if not create               */
        { Tree[idx] = malloc(...)
          Tree[idx]->Attribute = NewAuxData(x,y,z)
        } else
          AddAuxData(Tree[idx]->Attribute, x,y,z)
        if(ORI[p]>h){                               /* Local partitioning                                */
          Tree[idx]->Parent = NodeOffsetAtLevel[h]  /* Finalize Singleton node                           */
          Tree[idx]->Status = Finalized
          Tree[idx]->Level  = ORI[p]
          NumberOfNodes[ORI[p]] += 1;
          NodeAtLevel[ORI[p]] = FALSE
          AddAuxData(attr,x,y,z) }                  /* Update the attribute at the current node          */
    } else
       AddAuxData(attr,x,y,z)                       /* If pixel intensity is the same in both images */
    for (every neighbor q of p)                     /* Process the neighbors                             */
    {   if (STATUS[q] == "NotAnalyzed")
        {   HQueue-add(P_ORI[q],q)                  /* Add in the queue                                  */
            STATUS[q] = "InTheQueue"
            NodeAtLevel[P_ORI[q]] = TRUE            /* Confirm node existence                            */
            if (P_ORI[q] > P_ORI[p])                /* Check for child nodes                             */
            {   m = P_ORI[q]
                child_attribute = NULL
                do{                                 /* Recursive child flood                             */
                    m = flood(m,child_attribute)
                } while (m != h)
                MergeAuxData(attr,child_attribute
  }}}}

  NumberOfNodes = NumberOfNodes[h] + 1              /* Update the node index                             */
  m = h-1                                           /* 2nd step: defines father                          */
  while ((m >= 0) and (NodeAtLevel[m] = FALSE)) m = m-1;
  if (m >= 0){                                      /* Node parent is not the background                 */
     idx = NodeOffsetAtLevel[h] -1                  /* Check if node exists and create if not            */
     if(Tree[idx]==NULL) { ... }                    /* (as in the ORI[p]!=h case)                        */
     Tree[idx]->Parent = NodeOffsetAtLevel[m]       /* Compute the parent node                           */
  } else {                                          /* Node parent is the background                     */
    idx = NodeOffsetAtLevel[h]                      /* Check if node exists and create if not            */
    if(Tree[idx]==NULL) { ... }                     /* (as in the ORI[p]!=h case)                        */
    Tree[idx]->Parent = idx                         /* Compute the parent node                           */
  }
  MergeAuxData(Tree[idx]->Attribute, attr)          /* Merge node attributes                             */
  Tree[idx]->Status = Finalized                     /* Finalize node                                     */
  Tree[idx]->Level = h
  NodeAtLevel[h] = FALSE
  thisAttribute = Tree[idx]->Attribute              /* Set 'thisAttribute' for the parent node           */
  return (m) }
```

Fig. 7. The flooding function of the Dual Input Max-Tree algorithm adopted for attribute openings and thinnings. The parameters `h` and `m` are the current and child node gray levels while `attr` is a attribute count at level `h` within the same connected component. The parameter `thisAttribute` is used to pass child attributes to parent nodes.

detecting the parent node id. The attribute is initialized and upon completion $NodeAtLevel[ORI[p]]$ is set to $FALSE$ indicating that this node is finalized. Note that since the singleton nodes are not flooded, they do not return their attributes to the their parent node by default, hence `attr` at the current level must be updated separately.

Both cases are demonstrated in Fig. 8 using a 1D example. The first two diagrams illustrate the nesting of peak components in the original image X and the connectivity mask M. Note that M was chosen such that P^0_{X2} and P^1_{X2} from X are clustered to a single peak component while P^1_{X1} vanishes. This results in replacing C^0_2 and C^1_2 with a single node in which the attributes of the two components are merged and splitting the node C^1_1 to a number of singleton nodes equal to the number of pixels in P^1_{X1} (illustrated in the last diagram).



P^0_{X3}
P^0_{X2}
P^1_{X2}
P^0_{X1}
P^1_{X1}
P^0_{X0}
X
P^0_{M3}
P^0_{M2}
P^0_{M1}
P^0_{M0}
M: connectivity mask
C^0_3 C^0_2 C^1_2 C^0_1 C^1_1 C^0_0
Max-Tree of X (standard)
C^0_3 C^0_2 C^0_1 C^1_1 C^2_1 C^0_0
Max-Tree of X (mask-based)

Fig. 8. Dual Input Max-Tree: The attributes of C^0_2 and C^1_2 are merged to C^0_2 since all pixels at level $h = 2$ are clustered to a single peak component. Furthermore C^1_1 breaks up to a number of singleton nodes equal to the number of pixels in P^1_{X1}.

If there is no mismatch between ORI and P_ORI we simply update `attr` and proceed with inspecting the neighborhood of p. The number of neighbors depending on the foreground connectivity is stored temporarily in a dynamically allocated array from which we retrieve them sequentially and inspect their status. If the status of a neighboring pixel q is set to $NotAnalyzed$, its placed in the appropriate queue entry at level $P\_ORI[q]$ and its $STATUS$ and $NodeAtLevel$ attributes are updated accordingly. This process terminates there unless the pixel q is at a higher level from p. In that case, flooding halts at level $P\_ORI[p]$ and a recursive call to the same function initiates flooding at level $P\_ORI[q]$. This is repeated until reaching the regional maximum along the given root path. Once a node is flooded, there are no more pixels in the queue for the given level therefore the algorithm proceeds with parent detection and node finalization. Note that as opposed to the conventional Max-Tree, the node attribute is merged with `attr` since the last one is updated by pixels at the same level or by singletons at higher levels while the node attribute is possibly updated by local clusters already flooded at higher levels in P_ORI. A parameter `thisattr` is also updated with the overall node attribute and returns to the calling function (usually the flood of the parent). The Max-Tree structure is completed when all nodes are finalized.

5.2.3 Masks by Operators with Nonflat Structuring Elements

Connectivity masks generated by operators with nonflat structuring elements often introduce new gray levels or remove existing ones. The Dual-Input Max-Tree algorithm structure is based on the histogram of the original image and nodes generated on gray levels that are not present in ORI overlap with others creating a memory conflict. To counter this, on the initialization of the Max-Tree structure we allocate a total of twice the image size entries for Max-Tree nodes and segment the structure based on the sum of the maximum number of pixels in ORI and P_ORI for each level.

In the case where gray levels are removed, a further routine is required to handle a possible intensity mismatch between $h_{\min}$ in ORI and in P_ORI. If $h_{\min}$ in P_ORI is smaller than that in ORI, no action is required since all nodes will be finalized during the flooding procedure. If, however, the opposite is true, i.e., $h_{\min}$ in P_ORI is higher than in ORI, then the flood function will stop when it reaches $h_{\min}$ in P_ORI on all nodes at gray levels below it will remain nonfinalized. Furthermore, the structure will not have a root node. To counter this when reaching the only node at $h_{\min}$ in P_ORI, we reduce by one the updated node counter since no other nodes can be found at this intensity. A postprocess flag which is set if this mismatch occurs during the computation of the image histograms, triggers an additional routine that follows the tree flooding. Starting from $h_{\min}$ in P_ORI to $h_{\min}$ in ORI, the only one node that can exist per level in this margin is detected, its attribute measure is updated, the parent node is set and it is finalized in the same way as in the flood function.

5.2.4 Attribute Management

Attributes are managed by the use of four different functions; $InitializeAttribute()$, $NewAuxData()$, $AddAuxData()$, and $MergeAuxData()$. Our implementation demonstrates two types of attribute filtering, area/volume openings and elongation filtering based on moment invariants. To handle both, we use a structure called $InertiaData$ made of the area/volume count and four sums namely `SumX`, `SumY`, `SumZ`, and `SumSquares`. For 3D data sets, the shape filter is described in Section 2.3 and in [21].

$InitializeAttribute()$ simply allocates a structure of size $InertiaData$ and initializes all members to zero. $NewAuxData()$ does similar but initializes area/volume to 1 and sets the four sums to the given coordinates. $AddAuxData()$ updates the area count by 1 and adds the x, y, z coordinates to the corresponding sums. `SumSquares` is updated by the sum of the squared x, y, z coordinates. Last, $MergeAuxData()$ merges two structures of size $InertiaData$, the first corresponding to the parent data and the second to the child data and sums the individual members accordingly. With the aid of these four functions, the algorithm allows a number of other attributes to be computed this way.

5.3 Filtering and Image Restitution

The construction phase of the Dual-Input Max-Tree algorithm returns the same type of tree structure with the conventional Max-Tree. Routines for attribute filtering therefore do not differ between the implementations.

Filtering the Max-Tree constitutes a separate stage and involves visiting all nodes of the tree maximally twice. The node attributes are compared against a threshold λ and if the criterion as in (9) is not met the parent pointers of children of C^k_h are updated to point at the oldest "surviving" ancestor of C^k_h. The comparison is repeated until the criterion is satisfied. This is described as the *Direct Rule* [16] and has no further effect on the descendants of the filtered node. In contrast to this, the *Subtractive Rule* from [10], [20], [21], classified as a nonpruning strategy lowers in gray value all the descendants by the same amount as C^k_h itself. Other filtering rules are described by Salembier et al. [16], [29].

The node attributes are computed upon visiting each node from the attribute data stored in the node structure during the construction phase of the tree. This is realized by a routine implementing the $I/V^{5/3}$ term explained in Section 2.3.

The output image Out is generated by visiting all pixels p, retrieving their node ids from $ORI[p]$ and $STATUS[p]$ and assigning the output gray level of that node to $Out[p]$.

6 Experiments and Discussion

The Dual-Input Max-Tree algorithm has been employed for area openings in [15] and in this paper an extension is presented to handle more complicated attributes such as the elongation measure discussed in Section 2.3 both in 2D and 3D. Furthermore, the new update supports connectivity masks generated by nonflat structural operators. The Dual-Input Max-Tree algorithm is derived from the conventional Max-Tree, therefore it shares a number of characteristics concerning its performance which are discussed in depth in [16], [28]. If the same image is used twice in the Dual-Input Max-Tree algorithm, i.e., if $M = X$, it simply returns the Max-Tree of the original image.

In this section, we first demonstrate the new features of the algorithm on a 3D biomedical data set and compare the result with that obtained using the same filter based on ordinary connectivity. The second part demonstrates the filtering improvements using various operators that were previously excluded due to constraints imposed by the earlier formulation of the second-generation connectivity framework. The performance of the algorithm is evaluated by experiments on the 3D data set by measuring the CPU times for multiple runs and comparing it against the conventional Max-Tree. Dependencies of the algorithm are also discussed. All experiments were carried out on a 2.8 GHz Pentium 4 CPU with 1 GB DDR memory. Our implementation was written in ANSI-C and the code is available upon request.

6.1 Three-Dimensional Biomedical Data Sets

This first experiment shows the applicability of the Dual-Input Max-Tree algorithm to the case of operator-based second-generation connectivity, in the case of a nonincreasing attribute. Max-Trees have been employed for volume filtering and 3D filament enhancement [21] of biomedical data sets. In this section, we demonstrate a similar application with second-generation connected volumes. The algorithm uses the nonincreasing 3D shape filter based on the elongation measure of the filamentous structures (discussed in Section 2.3). All the illustrations are isosurface projections.

The data set shown in Fig. 9a is a $256 \times 342 \times 243$, 8-bit confocal microscopy volume of a pyramidal neuron. The noise density is relatively low, but the filamentous structures (the dendrites in this case) are fragmented at low levels. Filtering using ordinary connectivity consequently removes noise together with a considerable fraction of the dendrites. If the volume is clustered, however, nearby fragments are connected into a single entity with overall elongation greater than the threshold λ and, hence, are retained. The top right image shows the result of the shape filter based on clustering connectivity and using a cubic SE of size $5 \times 5 \times 5$. The two images of the bottom row show two different views of the difference volume between the filtering methods. Timings for this data sets were were 3.498 s for tree construction and 0.089 s for tree filtering using the conventional Max-Tree and 3.849 s and 0.061 s using the Dual-Input Max-Tree algorithm, respectively.

6.2 Images of Proteins

In this section, we demonstrate the use of AS filters and directional Minkowski additions with adaptive structuring elements for generating connectivity masks. All cases are compared with attribute thinnings based on clustering connectivity since the objective is to extract filamentous structures with disconnected members from the noisy background.

The first case is illustrated in Fig. 10. Generating a connectivity mask with either dilations or closings proves insufficient since both operators merge neighboring particles to the targeted object. The third image from the left - top row shows the result of elongation filtering using a mask (middle image) by a closing with a square SE of size 5×5 and $\lambda = 3$. We arrive at these settings since with any larger SE we cluster too much noise together with the protein chain and with any larger λ further parts of the chain are removed. If, however, after a closing, we perform an opening with an SE of equal size all the thin bridges introduced by the closing are removed. Applying a further closing by an SE of 13×13 clusters all desired members of the chain to large elongated fragments which are retained after filtering. The specific operator is both increasing and idempotent, but neither extensive nor anti-extensive. The filtering improvement is evident in the second image of the bottom row and to highlight the difference between the two types of connectivity masks we compute the difference image and enhance it with contrast stretching. Concluding it can be seen that to avoid merging background noise, ordinary clustering-based masks are limited and, therefore, cannot capture the entire chain thus elongation filters with high attribute threshold flatten certain chain regions by removing higher-intensity components. Had we used a sequence of attribute filters, with alternating clustering-based and contraction-based connectivities, we would not obtain the same result. This is because the partitioning operator may remove small objects which are irretrievably lost to any consecutive clustering operation. We perform the clustering/partitioning sequence, and only after pixels have been grouped properly, decide what the attributes of these groups are. Any interim filtering could distort or even completely destroy a group we wish to retain.

The second case illustrated in Fig. 11 is handled with directional Minkowski additions. To generate the connectivity mask, we first employ the Gabor wavelet-based method of [30] to compute the predominant orientation along which to perform the addition. The kernel used is of a fixed size and we use 18 angle steps of 10 degrees each. The convolution of the resulting wavelets with the input image is only used to compute the direction with the maximum filter response and no intensity modification takes place. Weak responses are ignored by thresholding the output while in cases where there is more than one response we handle each orientation separately. For each response above the threshold, we perform a Minkowski addition using a line SE along the given orientation. Our implementation is based on the original algorithm by Soille et al. [31]. To counter thin line fragments of high elongation emerging at the background, we compute an additional structural opening which for a square SE of size 3×3 yields a connectivity mask as seen in the first image of the bottom row in Fig. 11. The filtered output with an elongation threshold $\lambda = 6$ is given in the next image and the difference between this method and the equivalent result using the optimal clustering-based connectivity mask is shown in the last image after a log enhancement. The filtered outputs of both methods retain certain background elements

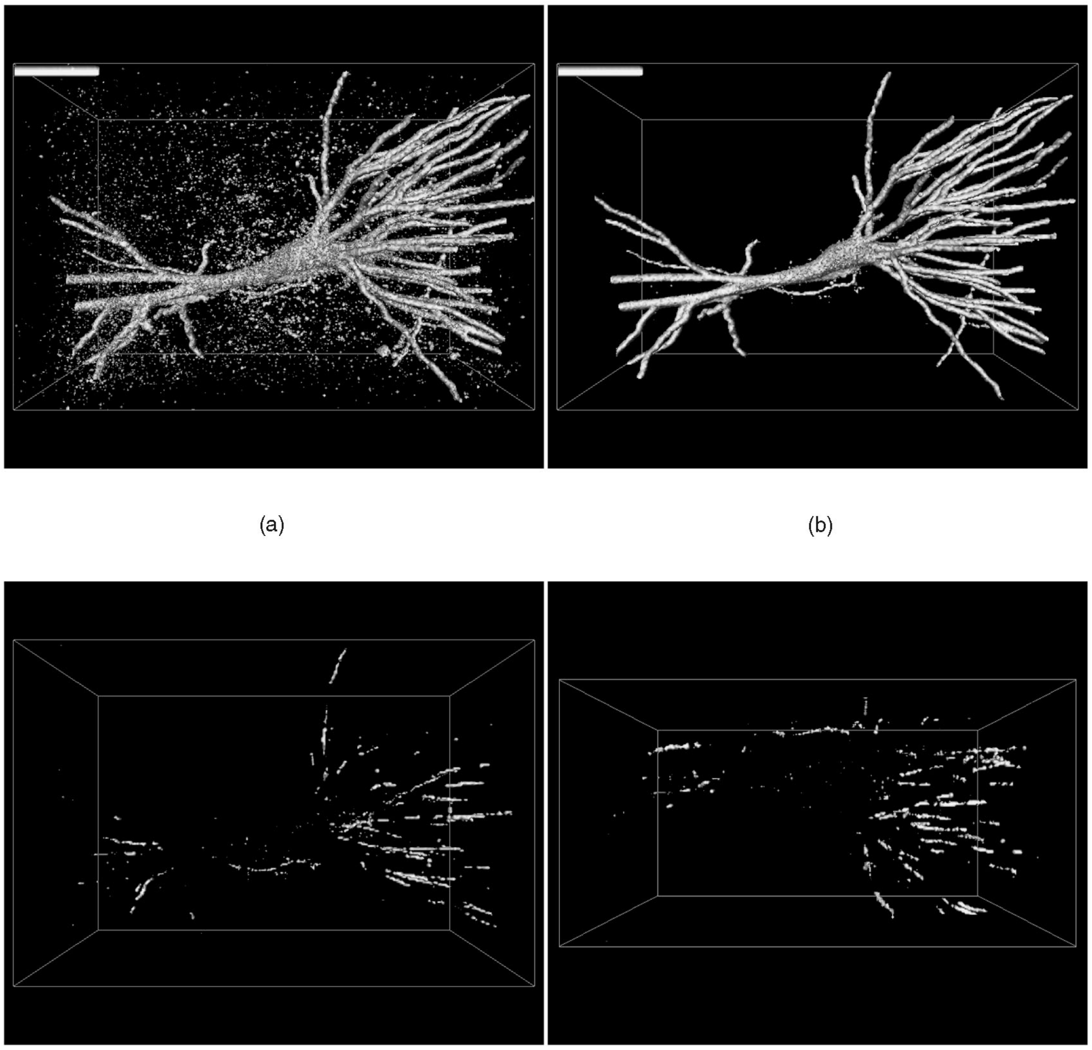

(a) (b) (c) (d)

Fig. 9. Top row: Isosurface projection of the neuron at (a) level 1 and (b) elongation filtering using Dual-Input Max-Tree algorithm (isolevel 1). Bottom row: difference volume between the Dual-Input and conventional Max-Tree outputs—$x - y$ and $y - z$ views (isolevel 1).

which can be later removed with an area opening. The advantage of the idempotent, nonincreasing, and neither extensive nor antiextensive operator presented as opposed to an ordinary closing is that it merges elements of the chain only and we can use large enough kernels to create a single object. By contrast, using a structural closing we face limitations similar to those mentioned in the previous example and consequently the protein chain remains fragmented. This results in severe flattening by the elongation filter which is illustrated in the difference image.

The images used in this section are courtesy of the Institute for Molecular Virology—University of Wisconsin, Madison, and can be obtained from the online Electron Micrograph Library at http://www.biochem.wisc.edu/inman/empics/index.html.

6.3 Computational Complexity

The computational complexity of our implementation has a strong dependency on the image content and on the size of the structuring element used to create the connectivity mask. The content obviously affects the size of the tree, the number of recursions and, therefore, the time complexity.

The size of the SE affects the timing depending on the type of connectivity mask. For cases where M is generated by an extensive operator from X, the greater the size of the SE, the lower the number of the connected components. Therefore, building the Max-Tree should consume less time as the SE size increases. This however is not the always the case with dynamically allocated attributes since the greater the number of mismatches between voxels in the two volumes the larger the number of searches for nodes at the parent level in ORI. With attributes represented by a scalar variable instead this is not true since no parent nodes need to be detected and allocated before finalizing the one being flooded.

If M is generated by an antiextensive operator from X, then the time overhead rises as the size of the SE increases since there are more singletons generated and, hence, more nodes that need to processed. Dynamically allocated attributes contribute an additional delay as discussed above.

For cases where M results from a neither extensive nor antiextensive operator from X or is independent of X, the performance of the algorithm depends strongly on the content of M. Means of evaluating it is by studying the frequency of

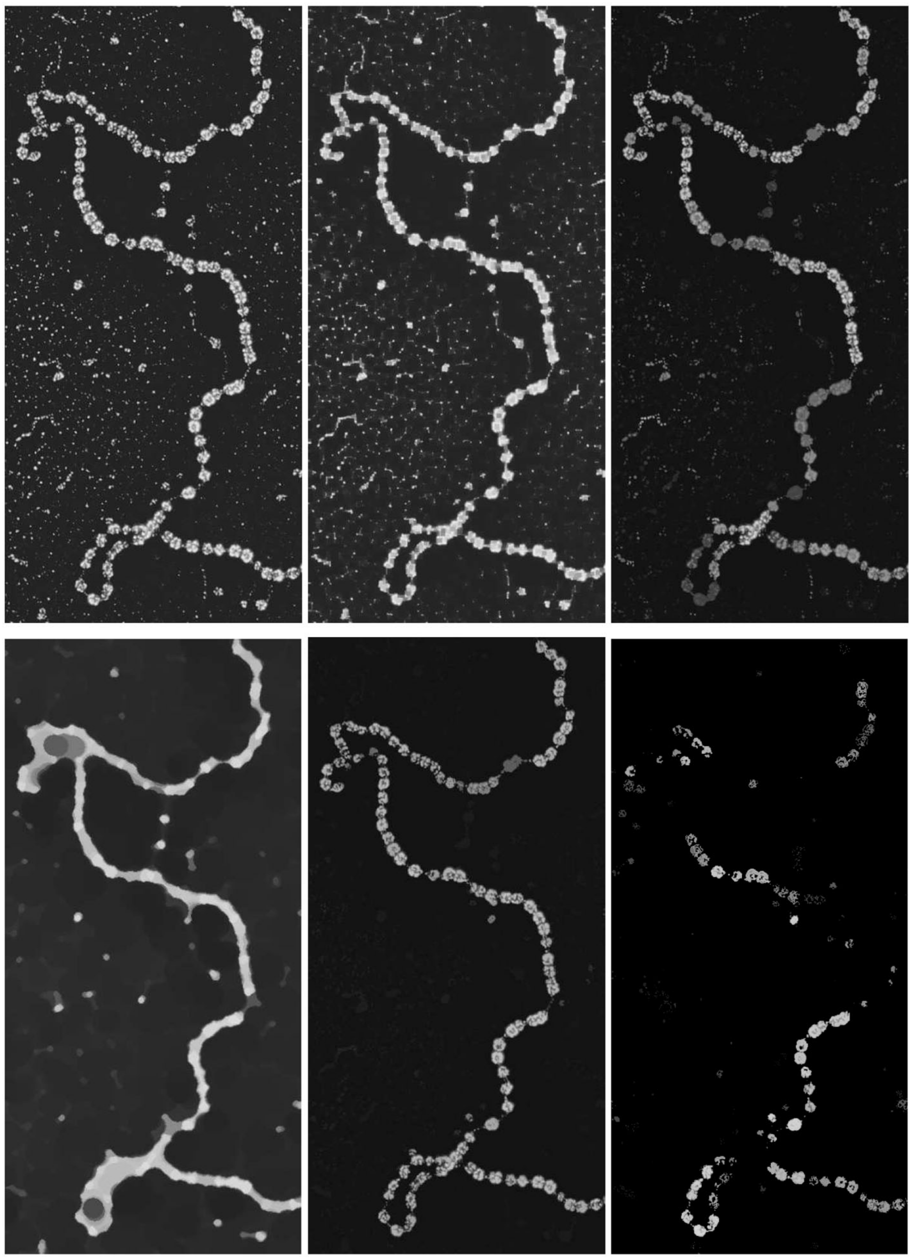


Fig. 10. Top row: The original image, the mask by a closing with an SE of size 5×5, and the filtered output with $\lambda = 3$. Bottom row: The mask by an ASF followed by an additional closing, the filtered output with $\lambda = 4$, and the difference image after contrast enhancement.

occurrence of regions of M that appear as the result of an either extensive or antiextensive operator with respect to X.

The filtering stage contributes a fixed time overhead which varies with the image/volume size. In all cases of M, filtering needs to access each pixel at most twice, therefore, the search depth along different root-paths is compensated by reducing the number of remaining pixel visits.

7 Conclusions and Further Work

In this paper, we presented an extension of the theory of second-generation connectivity. The connectivity opening introduced for this purpose is associated with connectivity masks rather than structural operators eliminating this way dependencies on their properties. This allows for images to be connected in any arbitrary fashion according to the connectivity mask and poses no restriction as to how the mask should be created. The main advantage is that any operator can be used to derive a second-generation connectivity class as opposed to the previous framework that was restricted to certain dilations, closings and openings only. Indeed, we could even use images of the same scene taken in different frequency bands (e.g., optical/IR combinations), or using different imaging modalities (optical/range imaging or registered CT/MRI pairs) to act as connectivity masks. When using a single filtering step, using a mask derived from the image by an arbitrary operator, the distinction between mask-based and operator-based connectivity may not seem very great. When trying to compute, e.g., granulometries using multiple filtering steps, the distinction is far more obvious. The difference is whether one considers mask generation and attribute filtering as a single operator (as in the operator-based case), or as two distinct steps (as in the mask-based case). If we use operators which do not meet the requirements

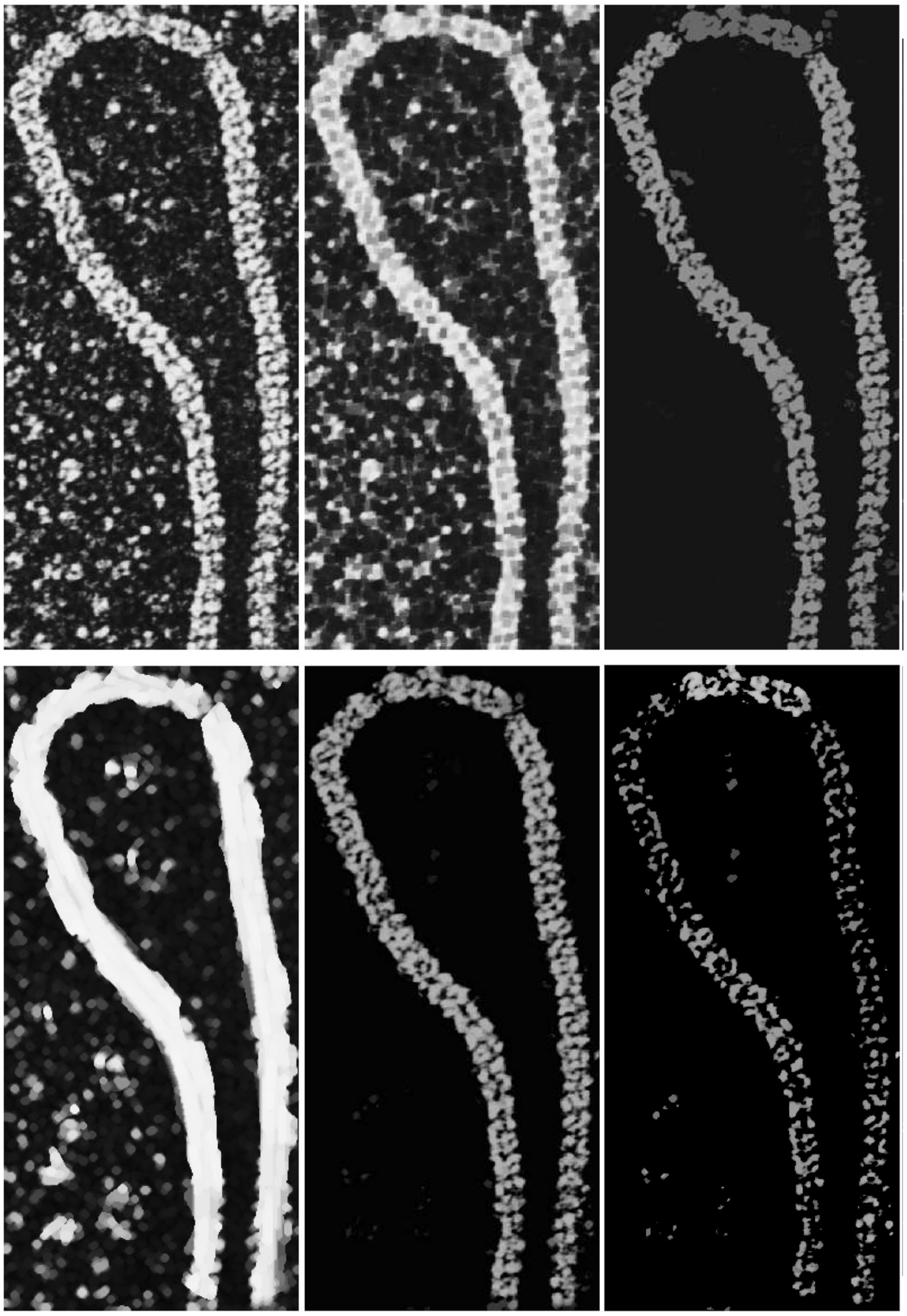


Fig. 11. Top row: The original image, the mask by a closing with an SE of size 5×5, and the filtered output with $\lambda = 6$. Bottom row: The mask described in the text, the filtered output with $\lambda = 6$, and the difference image after contrast enhancement.

imposed by the operator-based framework, but insist on interpreting mask-generation as part of the resulting filter, the question arises what the properties of such filters are. While our theory allows the use of any operator for mask generation, it is not concerned with the properties of the resulting combined mask-generation/filtering operator. By studying these combinations, we may find operators, beyond the known examples, which actually yield an operator-based second-generation connectivity, which is neither clustering nor partitioning. This is the topic of future work.

This theoretical work is complemented by and indeed validates, an efficient algorithm for attribute filtering using mask-based connectivity, referred to as the Dual-Input Max-Tree algorithm, which is demonstrated on both 2D and 3D data sets. The algorithm is an extension of the conventional Max-Tree [16]. The current version supports connectivity masks generated with both flat and nonflat structuring elements and provides the functionality for a wider range of attributes to be computed.

Potential applications of this work include filtering and segmentation of data sets characterized by thin elongated structures (like the neuron demonstrated in the previous section), connected component analysis and processing of second-generation connected sets and flexible attribute management depending on the image context. The relatively low-computational requirement in 2D examples makes it possible to use the presented algorithm also in real-time applications such as motion detection/analysis, tracking, and decision making tasks.

Future work on this area involves deriving connected operators that can counter the oversegmentation effect in the case of partitioning [32], [33], [34], with extensions to gray scale as well as algorithmic methods for their efficient computation. The issue of noise clustering is also being investigated and currently we are working on attribute-based clustering techniques that will allow the algorithm to cluster only objects of similar structural characteristics.

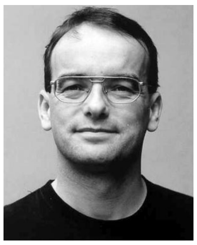
Acknowledgments

This work is funded by The Netherlands Organization for Scientific Research under project number 612.065.202. A special thanks to E.R. Urbach and A.C. Jalba for code sharing and development.

References

[1] *Image Analysis and Mathematical Morphology. II: Theoretical Advances,* J. Serra, ed. Academic Press, 1988.
[2] P. Salembier and J. Serra, "Flat Zones Filtering, Connected Operators, and Filters by Reconstruction," *IEEE Trans. Image Processing,* vol. 4, pp. 1153-1160, 1995.
[3] R. Diestel, *Graph Theory.* Springer-Verlag, 1997.
[4] T.Y. Kong and A. Rosenfeld, "Digital Topology: Introduction and Survey," *Computer Vision, Graphics, and Image Processing,* vol. 48, pp. 357-393, 1989.
[5] H.J.A.M. Heijmans, *Morphological Image Operators.* Academic Press, 1994.
[6] H.J.A.M. Heijmans, "Connected Morphological Operators for Binary Images," *Computer Vision and Image Understanding,* vol. 73, pp. 99-120, 1999.
[7] L. Vincent, "Morphological Grayscale Reconstruction in Image Analysis: Application and Efficient Algorithm," *IEEE Trans. Image Processing,* vol. 2, pp. 176-201, 1993.
[8] E.J. Breen and R. Jones, "Attribute Openings, Thinnings and Granulometries," *Computer Vision and Image Understanding,* vol. 64, no. 3, pp. 377-389, 1996.
[9] V. Caselles and P. Monasse, "Grain Filters," *J. Math. Imaging and Vision,* vol. 17, pp. 249-270, 2002.
[10] E.R. Urbach and M.H.F. Wilkinson, "Shape-Only Granulometries and Grey-Scale Shape Filters," *Proc. Sixth Int'l Symp. Math. Morphology,* H. Talbot and R. Beare, eds., pp. 305-314, 2002.
[11] U. Braga-Neto and J. Goutsias, "A Theoretical Tour of Connectivity in Image Processing and Analysis," *J. Math. Imaging and Vision,* vol. 19, pp. 5-31, 2003.
[12] C. Ronse, "Set-Theoretical Algebraic Approaches to Connectivity in Continuous or Digital Spaces," *J. Math. Imaging and Vision,* vol. 8, pp. 41-58, 1998.
[13] J. Serra, "Connectivity on Complete Lattices," *J. Math. Imaging and Vision,* vol. 9, pp. 231-251, 1998.
[14] U. Braga-Neto and J. Goutsias, "Connectivity on Complete Lattices: New Results," *Computer Vision and Image Understanding,* vol. 85, pp. 22-53, 2002.
[15] G.K. Ouzounis and M.H.F. Wilkinson, "Second-Order Connected Attribute Filters Using Max-Trees," *Proc. Seventh Int'l Symp. Math. Morphology,* C. Ronse, L. Najman, and E. Decencière, eds., pp. 65-74, 2005.
[16] P. Salembier, A. Oliveras, and L. Garrido, "Anti-Extensive Connected Operators for Image and Sequence Processing," *IEEE Trans. Image Processing,* vol. 7, pp. 555-570, 1998.
[17] J. Serra, "Connections for Sets and Functions," *Fundamenta Informaticae,* vol. 41, pp. 147-186, 2000.
[18] C. Ronse, "Openings: Main Properties, and How to Construct Them," Philips Research Laboratory Brussels, unpublished manuscript, 1990.
[19] L. Vincent, "Morphological Area Openings and Closings for Greyscale Images," *Proc. NATO Shape in Picture Workshop,* pp. 197-208, Sept. 1992.
[20] E.R. Urbach, J.B.T.M. Roerdink, and M.H.F. Wilkinson, "Connected Shape-Size Pattern Spectra for Rotation and Scale-Invariant Classification of Gray-Scale Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 2, pp. 272-285, Feb. 2007.
[21] M.H.F. Wilkinson and M.A. Westenberg, "Shape Preserving Filament Enhancement Filtering," *Proc. Fourth Int'l Conf. Medical Image Computing and Computer-Assisted Intervention,* W.J. Niessen and M.A. Viergever, eds., pp. 770-777, 2001.
[22] H.J.A.M. Heijmans, "Morphological Filters," *Proc. Summer School Morphological Image and Signal Processing,* 1995.
[23] H.J.A.M. Heijmans, "Composing Morphological Filters," *IEEE Trans. Image Processing,* vol. 6, no. 5, pp. 713-723, 1997.
[24] P. Maragos and R.D. Ziff, "Threshold Superposition in Morphological Image Analysis Systems," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 12, no. 5, pp. 498-504, May 1990.
[25] U. Braga-Neto and J. Goutsias, "Grayscale Level Connectivity: Theory and Applications," *IEEE Trans. Image Processing,* vol. 13, no. 12, pp. 1567-1580, 2004.
[26] R. Jones, "Connected Filtering and Segmentation Using Component Trees," *Computer Vision and Image Understanding,* vol. 75, pp. 215-228, 1999.
[27] C. Tzafestas and P. Maragos, "Shape Connectivity: Multiscale Analysis and Application to Generalized Granulometries," *J. Math. Imaging and Vision,* vol. 17, pp. 109-129, 2002.
[28] A. Meijster and M.H.F. Wilkinson, "A Comparison of Algorithms for Connected Set Openings and Closings," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 4, pp. 484-494, Apr. 2002.
[29] P. Salembier, "Binary Partition Tree as an Efficient Representation for Image Processing, Segmentation and Information Retrieval," *IEEE Trans. Image Processing,* vol. 9, no. 4, pp. 561-576, 2000.
[30] S.E. Grigorescu, N. Petkov, and P. Kruizinga, "Comparison of Texture Features Based on Gabor Filters," *IEEE Trans. Image Processing,* vol. 11, no. 10, pp. 1160-1167, 2002.
[31] P. Soille, E. Breen, and R. Jones, "Recursive Implementation of Erosions and Dilations Along Discrete Lines at Arbitrary Angles," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 5, pp. 562-567, May 1996.
[32] G.K. Ouzounis and M.H.F. Wilkinson, "Countering Oversegmentation in Partitioning-Based Connectivities," *Proc. Int'l Conf. Image Processing,* vol. III, pp. 844-847, 2005.
[33] M.H.F. Wilkinson, "Attribute-Space Connected Filters," *Proc. Seventh Int'l Symp. Math. Morphology,* C. Ronse, L. Najman, and E. Decencière, eds., pp. 85-94, 2005.
[34] M.H.F. Wilkinson, "Attribute-Space Connectivity and Connected Filters," *Image and Vision Computing,* vol. 25, no. 4, pp. 426-435, Apr. 2007.

Georgios K. Ouzounis received the MPhil degree in computer science from the Department of Computing and Electrical Engineering, Heriot Watt University, Edinburgh, Scotland in 2001. He is now working at the Institute of Mathematics and Computing Science, University of Groningen (RUG), The Netherlands on the GC-MORSE project toward the PhD degree. The prime area of his research is connectivity generalizations, multiscale connectivity, and medical image analysis in the field of mathematical morphology.



Michael H.F. Wilkinson received the MSc degree in astronomy from the Kapteyn Laboratory, University of Groningen (RUG) in 1993, after which he worked on image analysis of intestinal bacteria at the Department of Medical Microbiology, RUG. This work formed the basis of his PhD at the Institute of Mathematics and Computing Science (IWI), RUG, in 1995. He was appointed as researcher at the Centre for High Performance Computing (also RUG) working on simulating the intestinal microbial ecosystem on parallel computers. During that time, he edited the book *Digital Image Analysis of Microbes* (John Wiley, 1998) together with Frits Schut. After this, he worked as a researcher at the IWI on image analysis of diatoms. He is currently assistant professor at the IWI. He is a senior member of the IEEE and the IEEE Computer Society.