Hands-on morphological processing using the max-tree

Roberto Souza
Department of Electrical and Software Engineering
Schulich School of Engineering
University of Calgary
Calgary, Canada
{roberto.medeirosdeso}@ucalgary.ca

I. General information

This tutorial will use the iamxt toolbox, which is a max-tree toolbox wrapped in Python with many methods for building, processing and visualizing the max-tree. It was developed by our research group and is available for download at https://github.com/rmsouza01/iamxt. The documentation and examples of the code is available at http://adessowiki.fee.unicamp.br/adesso/wiki/iamxt/view/.

The hands-on portion of the tutorial will be developed on Adessowiki [1], which is an on-line collaborative scientific programming platform that allows programming without installation of software and packages, using just a web browser. All the packages necessary for the development of this tutorial are already installed on Adessowiki, therefore students only need a computer with a web browser and internet connection to follow this tutorial. The only difference between programming in Python on adessowiki from programming in a personal computer are the functions for reading and displaying images. The code developed during the tutorial should be portable to a personal computer by changing these functions for other reading and displaying functions provided by packages like OpenCV [2].

II. Basic Image Processing Concepts

Definition 1. ***Grey-scale image:*** *It is a function $I(z) : E \to k$, $E \in \mathcal{N}^2$ and $k \in \mathcal{Z}$. z is an ordered pair (z_{lin}, z_{col}) that represents a pixel of the image, and $I(z)$ is its luminosity intensity, usually $I(z) \in [0, L-1]$, $L > 1$. If $L = 2$, the image is said to be a binary image, the pixels with intensity equal 1 compose the foreground and the pixels with intensity equal 0 compose the background of the binary image.*

Definition 2. ***Negative of an image:***

$$Neg(I)(z) = (L-1) - I(z), \quad \forall z \in E \tag{1}$$

Definition 3. ***Image thresholding:*** *The image thresholding is a procedure that given a threshold h, it transforms an image into a binary image. There are two cases the upper and the lower thresholding. They are expressed, respectively by the following equations:*

$$\mathcal{X}_h^{\geq}(z) = \begin{cases} 1 & \text{if } I(z) \geq h \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

$$\mathcal{X}_h^{<}(z) = \begin{cases} 1 & \text{if } I(z) < h \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

Definition 4. ***Neighborhood of a pixel:*** *The neighborhood of a pixel z corresponds to a set of coordinates $C(p) \subset E$ defined in relation to z in the image domain E. The most common pixel neighborhoods used are the neighborhoods $C4$ and $C8$ defined below:*

$$C4(z) = \{z + (-1, 0), z + (0, -1), z + (0, 1), z + (1, 0)\} \cap E \tag{4}$$

$$C8(z) = C4(z) \cup \{z + (-1, -1), z + (-1, 1), z + (1, -1), z + (1, 1)\} \cap E \tag{5}$$

The neighborhoods $C4$ and $C8$ of a pixel are illustrated in Fig. 1.

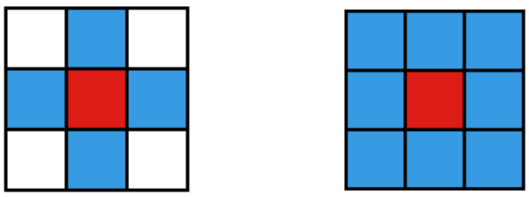


Fig. 1. Illustration of the neighborhoods $C4$ (a) and $C8$ (b) of a pixel. The blue pixels are the neighbors of the red pixels.

Definition 5. ***Connectivity:*** *An image can be seen as a graph, where the pixels correspond to the nodes. Two pixels are connected, i.e. there is an edge joining them, if they are neighbors, according to the neighborhood used, that share a common property that defines a component. The property may be color, brightness, range of brightness values, or anything else of interest.*

Definition 6. ***Adjacency:*** *Two pixels z_1 and z_2 are adjacent, if they are connected.*

Definition 7. ***Upper threshold set:*** *The upper threshold set is the set of connected components $\{C_{h,1}, C_{h,2}, ..., C_{h,n_{CC}}\}$ that compose the foreground of the binary image resulting of the upper threshold $\mathcal{X}_h^{\geq}(I)$. The similarity criterion to define the connectivity is that the pixels have the same grey-level.*

An important property of level sets is that for each connected component $C_{h+1,m}$ resulting of the upper threshold $\mathcal{X}_{h+1}^{\geq}(I)$ there is a connected component $C_{h,n}$ resulting of the upper threshold $\mathcal{X}_h^{\geq}(I)$, such that $C_{h+1,m} \subseteq C_{h,n}$.

Definition 8. ***Flat zone:*** *Flat zones are connected components in which the similarity criterion used to define connectivity is that the pixels have the same grey-level.*

Definition 9. ***Regional maximum:*** *A regional maximum of an image I is a flat zone M, if $I(z) > I(q)$, $\forall z \in M$ and q is any neighboring pixel of M.*

Definition 10. ***Partition*** *A partition $\mathcal{P}$ of a set X is a set of nonempty subsets of X such that every element x in X is in exactly one of these subsets.*

$$\mathcal{P}(X) = \{X_0, X_1, ...X_{n-1}\}, \quad X_i \cap X_j = \emptyset, \quad for \quad i \neq j \tag{6}$$

$$X = X_0 \cup X_1 ... \cup X_{n-1} \tag{7}$$

Definition 11. ***Grey-scale image ordering:*** *A grey-scale image f is said to be contained in a grey-scale image I of the same dimensions, $f \leq I$, if and only if:*

$$f[z] \leq I[z], \quad \forall z. \tag{8}$$

Definition 12. ***Anti-extensive filter:*** *An anti-extensive filter ψ is a filter such that:*

$$\psi(I) \leq I, \quad \forall I. \tag{9}$$

Definition 13. ***Connected filter [3]:*** *is a filter in which the partition composed of the input image flat zones is always finer than the partition of the filtered image flat zones. The mathematical definition of a connected filter ψ is given by:*

$$\mathcal{P}_I \subseteq \mathcal{P}_{\psi(I)}, \quad \forall I. \tag{10}$$

Connected filters do not create new contours in the image and they do not modify the position of the existing contours.

Definition 14. ***Extinction value [4]:*** *Consider M a regional maximum of an image I, and $\Psi = (\psi_\lambda)_\lambda$ is a family of decreasing connected anti-extensive transformations. The extinction value corresponding to M with respect to Ψ and denoted by $\varepsilon_\Psi(M)$ is the maximal λ value, such that M still is a regional maxima of $\psi_\lambda(I)$. This definition can be expressed through the following equation:*

$$\varepsilon_\Psi(M) = sup\{\lambda \geq 0 | \forall \mu \leq \lambda, M \subset Max(\psi_\mu(I))\}, \tag{11}$$

where $Max(\psi_\mu(I))$ is the set containing all the regional maxima of $\psi_\mu(I)$. Extinction values of regional minima can be defined similarly.

III. Max-tree and component tree

The component tree was proposed by Jones [5]. It is a data structure that represents an image through the hierarchical relationship of its connected components. It separates the image filtering procedure on three steps: tree construction, tree filtering, and image restitution. The max-tree was proposed by Salembier et al.[6]. It is a compact structure for the component tree representation. Most filters are designed thinking in terms of the component tree, but they are more efficiently implemented in the max-tree.

A. Component tree

The component tree was proposed as a structure for image representation that represents all connected components resulting of all possible thresholds of the image, and that provides an attribute signature as means of discriminating features in the image. It is efficient to implement connected anti-extensive filters, and by duality extensive filters.

The component tree, $\mathcal{T}_{CT}$, is a rooted tree in which each node stores a set of attributes. It is completely defined by its set of nodes $\mathcal{N}(\mathcal{T}_{CT})$:

$$\mathcal{N}(\mathcal{T}_{CT}) = \{0, 1, .., m-1\}, \tag{12}$$

and parents $\mathcal{P}(\mathcal{T}_{CT})$:

$$\mathcal{P}(\mathcal{T}_{CT}) = \{p_i | i \in (\mathcal{N} \setminus 0)\}, \tag{13}$$

where $\setminus$ is the set minus operand, and p_i is the parent of node i. Our convention is that node 0 is the root.

Suppose that the grey-levels of an image I are bounded between I_{min} and I_{max}. The upper threshold sets $\{C_{h,1}, C_{h,2}, ..., C_{h,n_{CC}}\}$ resulting of every upper threshold $\mathcal{X}_h^{\geq}(I)$ for h varying between I_{min} and I_{max} can be computed, and we know that threshold sets have the following property:

$$\forall C_{h+1,m}, \quad \exists C_{h,n} : C_{h+1,m} \subseteq C_{h,n}. \tag{14}$$

This hierarchy property allows a tree representation. For each connected component $C_{h,n}$ resulting of all threshold sets, there is a component tree node i with attributes $h_i = h$ and $C_i = C_{h,n}$. h_i and C_i are the basic attributes stored in the component tree nodes necessary to recover the image from the tree. Other attributes, such as height, area and volume can be computed and stored in the nodes during the tree construction or they can be computed just when required.

Node i is a parent of node j if C_i contains C_j, and $h_i = h_j - 1$. Note that the component tree may represent a pixel in more than one node, therefore it has some redundancy. The leaves of the component tree correspond to regional maxima in the image and the root represents the whole domain of the image.

The restitution of the image corresponding to a given component tree, is given by the following equation:

$$I(z) = \max_{\forall i} \{h_i | z \in C_i\}. \tag{15}$$

B. Max-tree

The max-tree can be seen as a compact structure for the component tree representation. Every operation that can be done in the component tree can be done in the max-tree and vice versa.

The max-tree, $\mathcal{T}_{MT}$, is a rooted tree in which each node stores a set of attributes. It is completely defined by its set of nodes $\mathcal{N}(\mathcal{T}_{MT})$:

$$\mathcal{N}(\mathcal{T}_{MT}) = \{0, 1, .., m' - 1\}, \tag{16}$$

and parents $\mathcal{P}(\mathcal{T}_{MT})$:

$$\mathcal{P}(\mathcal{T}_{MT}) = \{p_i | i \in (\mathcal{N} \setminus 0)\}. \tag{17}$$

$\widehat{C_{h,n}}$ is defined by:

$$\widehat{C_{h,n}} = \{z \in C_{h,n} | I(z) = h\}, \tag{18}$$

it represents a set of flat zones [7]. For each non-empty $\widehat{C_{h,n}}$, there is a max-tree node i with $h_i = h$ and $\widehat{C_i} = \widehat{C_{h,n}}$. The pixels represented by $\widehat{C_i}$ and their level h_i are the minimum set of attributes that a node i has to store in order to be able to recover the image from the max-tree. Note that storing $\widehat{C_i}$ is much more memory efficient than storing C_i, there is no redundancy in the pixels stored. The connected component C_i can be recovered by:

$$C_i = \widehat{C_i} \cup \bigcup_{\forall j \in D(i)} \widehat{C_j}, \tag{19}$$

where $D(i)$ is the set containing all the descendants of i. Node i is a parent of node j if C_i is the smallest connected component that contains C_j.

The number of connected components $nlevels$ that a max-tree node i represents is given by:

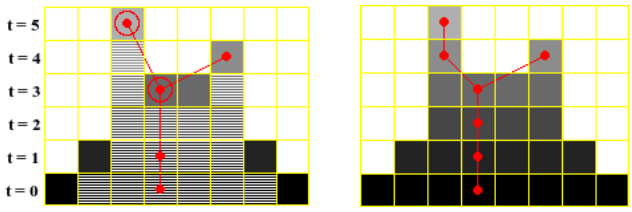
$$nlevels(i) = h_i - h_{p_i}. \tag{20}$$

If $nlevels(i) > 1$, this node is a **composite node**, i.e. a node that represents a connected component that remained unchanged for a sequence of threshold values. This notion of composite node is useful to implement the max-tree based algorithms.

The restitution of the image corresponding to a given max-tree, is an inverse mapping of the pixels stored in the nodes to the image coordinates system, and is expressed by the following equation:

$$I(z) = \{h_i | z \in \widehat{C_i}\}. \tag{21}$$

The max-tree and the component tree corresponding to the 1D image $I = [0, 1, 5, 3, 3, 4, 1, 0]$ is illustrated in Fig. 2. The composite nodes are represented by a double circles. In this example, there are two composite nodes both with $nlevels = 2$. The filled pixels in each connected component represent the pixels each max-tree node stores ($\widehat{C_i}$), and its union with the dashed pixels in the same connected component (C_i) are the result of applying equation 19 to this node. The number of component tree nodes is equal to the sum of the attribute $nlevels$ of each max-tree node. Note that there is no redundancy in the pixels stored by each max-tree node.



(a) Max-tree (b) Component tree

Fig. 2. (a) Max-tree and (b) component tree corresponding to the 1D image, $I = [0, 1, 5, 3, 3, 4, 1, 0]$.

Filtering the max-tree consists in contracting some of its nodes. There are two possible cases: a full node contraction and a partial composite node contraction. In the first case, the node is contracted with its parent, in the second case we contract a composite node by decreasing its $nlevels$ attribute. Both contractions increase the attribute $nlevels$ of the children of the contracted node.

The full node contraction procedure of a node i with its parent p_i using the direct rule is given by:

$$p_j = p_i, \quad \forall j \in children(i), \tag{22}$$

$$\widehat{C_{p_i}} = \widehat{C_{p_i}} \cup \widehat{C_i}, \tag{23}$$

$$\mathcal{N} = \mathcal{N} \setminus i, \tag{24}$$

where $children(i)$ is the set containing all the children of node i.

The partial composite node contraction of a node i by x is given by:

$$h_i = h_i - x, \quad 1 \leq x \leq nlevels(i) - 1. \tag{25}$$

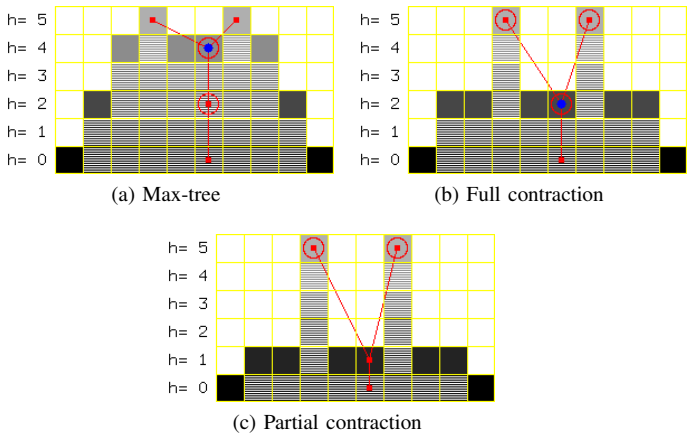
The parameter x defines how many levels the composite node will contract.

Fig. 3 illustrates the filtering procedure in the max-tree of the 1D image $I = [0, 2, 4, 5, 4, 4, 5, 4, 2, 0]$. The blue node in Fig. 3(a) will be fully contracted, and the result is shown in Fig. 3(b). Fig. 3(c) shows the partial contraction result of the blue node in Fig. 3(b) by $x = 1$. Note the increase of the attribute $nlevels$ of the children of the nodes being contracted.

It is important to mention that contracting a node of the max-tree is a connected filter, and after choosing which nodes will be fully contracted, partially contracted and the parameter x, the order of the contractions is irrelevant, the result will always be the same.

IV. Max-tree signature analysis

One important concept related to the max-tree is the attribute signature proposed by Jones [5]. The max-tree signature consists in analyzing an attribute variation starting at a leaf node and going towards the root, composite nodes must be considered in the analysis. Signature analysis is a powerful tool to inspect how the connected components evolve when you go



(a) Max-tree (b) Full contraction

(c) Partial contraction

Fig. 3. (a) max-tree, the node in blue will be fully contracted. (b) result of the full contraction. (c) result of partial contraction of the blue node in (b).

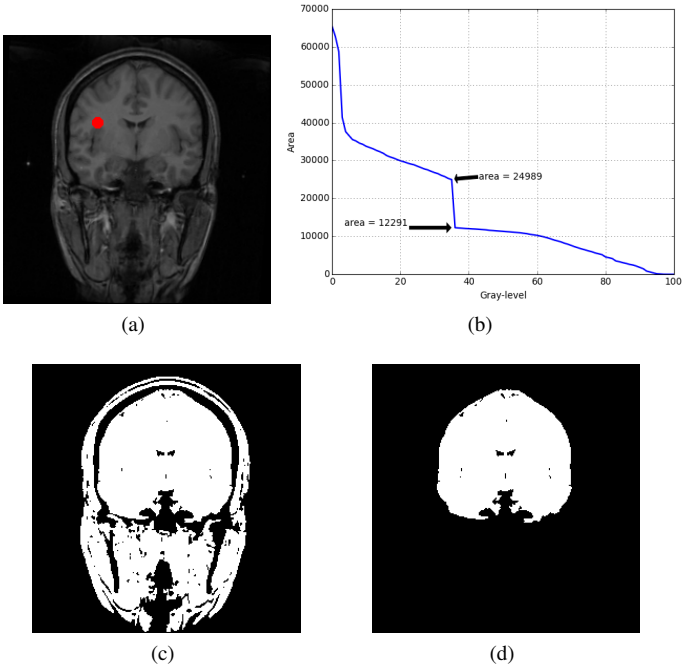
through many threshold values. For instance, when there is a sudden change in the area signature value, it may represent a connected component that split in two or more significant connected components. A connected component whose area remains practically unchanged for a sequence of thresholds is a maximally stable extremal region (MSER)[8]. An example using a brain image showing where the brain and the scalp split is displayed in Fig. 4. The CC immediately before the sudden change in the signature (Fig. 4(c)) has an area of 24989 pixels and it represents the brain and the scalp connected, while the CC immediately after the sudden change (Fig. 4(d)) has an area of 12291 pixels and it represents the brain after disconnecting from the scalp.

V. Max-tree array-based representation

The data structure used to represent the max-tree we are going to use was proposed in [9]. It is composed of two arrays: node index (NI) and node array (NA). NI is an array with the same shape of the original image, and its elements point to the max-tree nodes they belong. NA is an array that is organized such that its sequential scan is a tree traversal from the root to its leaves. This organization avoids the requirement of the array S. It is an array $M \times nattr$, where M is the number of max-tree nodes and $nattr$ is the number of attributes it stores. Actually, each column of NA can be seen as a different array each one storing different attributes of the nodes. In order to be able to recover the image from this representation, NA has to store at least the parent relationship (par), and the grey-level of each node (h). The illustration of the node oriented max-tree is depicted in Fig. 5. In this example, NA also stores the area of each connected component (column a of NA).

VI. iamxt toolbox

The iamxt toolbox uses the NA/NI max-tree representation. It accepts both 2D and 3D images as input of either 8-bit or 16-bit unsigned integers data type. Each column of NA represents a node and the lines represent the attributes of that node. By default our toolbox computes the area, number of children,



(a) (b)

(c) (d)

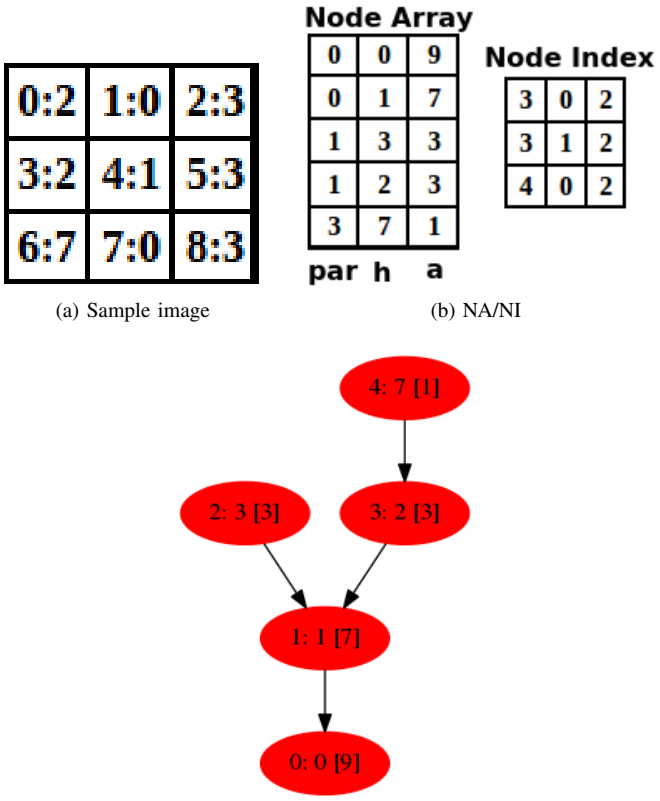
Fig. 4. (a) Original image. The red dot corresponds to a regional maximum (max-tree leaf). (b) Area signature. (c) Node reconstruction before the sudden drop in the area signature value and (d) node reconstruction after.

bounding-box coordinates, the sum of the coordinates in each axis for every max-tree node, and a seed for reconstruction of the CC corresponding to the node. The meanings of each line of NA are summarized in Table I.

TABLE I
Average max-tree construction, filtering and image restitution processing times in milliseconds.

Line index	Attribute	Description
0	par	Parent
1	nchild	Number of children
2	level	Gray-level
3	area	Node area
4	seed	Seed for node reconstruction
5	sumx	Sum of x coordinates
6	xmin	Minimum x coordinate
7	xmax	Maximum x coordinate
8	sumy	Sum of y coordinates
9	ymin	Minimum y coordinate
10	ymax	Maximum y coordinate
11	sumz	Sum of z coordinates
12	zmin	Minimum z coordinate
13	zmax	Maximum z coordinate

The Code Fragment 1 illustrates the syntax used to build the max-tree of a gray-scale image using connectivity C8. The NumPy [10] and the iamxt toolboxes are the basic toolboxes used in this tutorial. From now on,we will omit the imports and the max-tree construction lines of code from the code



(a) Sample image (b) NA/NI

(c) Node oriented max-tree.

Fig. 5. (a) Sample image. (b) NA/NI max-tree representation of the sample image , and (c) the node oriented max-tree illustration (node ID: h [area]).

fragments illustrated in this tutorial.

Code Fragment 1. Max-tree construction example..

```
import iamxt
import numpy as np

# Adessowiki image reading function
img = adreadgray(<image path>)

#Structuring element conectivity C8
Bc = np.ones((3,3), dtype = bool)

# Max-tree construction
mxt = iamxt.MaxTreeAlpha(img,Bc)
```

VII. Extraction of attributes from the max-tree nodes

One important feature of the max-tree is that it allows the efficient extraction of size and shape attributes from its nodes. We list some of the attributes that can be extracted:

1) Area (volume): number of pixels (voxels) that compose a CC.
2) Aspect ratio: length of the smaller bounding-box side divided by the length of the largest bounding-box side.
3) Rectangularity ratio: area (volume) of the CC divided by its bounding-box area (volume).
4) Perimeter (surface area): Number of pixels (voxels) that compose the boundary of the CC.
5) Average: average gray-level of the image in the region corresponding to the node's CC.
6) Standard-deviation: standard-deviation of the gray-levels of the image in the region corresponding to the node's CC.
7) Centroid: centroid of the node's CC.
8) Height: is a contrast attribute given by the following equation:

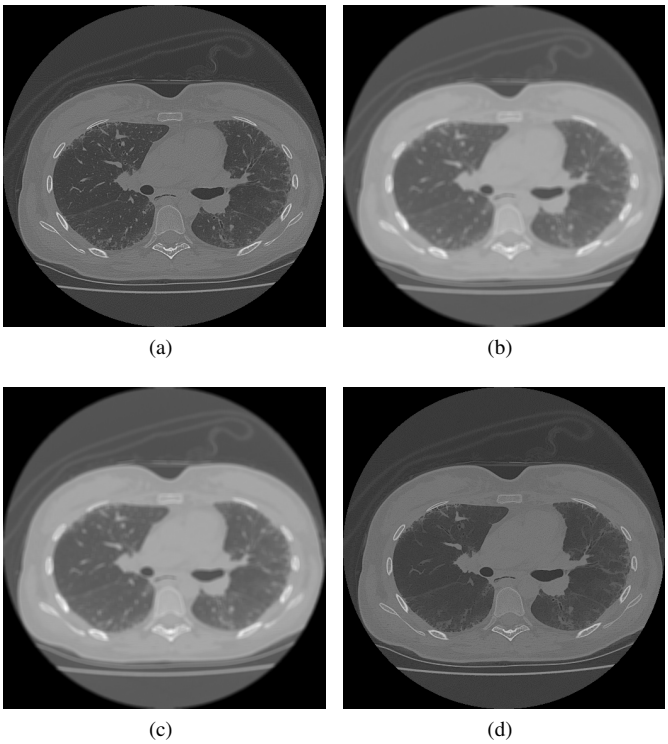
$$height(i) = \max_{\forall k \in descendants(i)} \{h_k - h_i\}. \quad (26)$$

VIII. Size and shape filtering

Image filters implemented on the max-tree are connected filters. The main advantage of these filters is that they do not blur the image. The advantage of implementing these filters in the max-tree structure is that for a sequence of filtering steps, the processing time is faster than using the non max-tree based implementations.

A. Area-open filter vs mean and median filters

A comparison between the mean, median, and the area-open filter implemented on the max-tree applied to a lung computer tomography (CT) image is depicted in Fig. 6. It is visible that the mean and median filters cause considerable blurring to the image, while the area-open implemented on the max-tree filters the image without producing that undesirable effect.



(a) (b)

(c) (d)

Fig. 6. (a) Lung CT image. (b) Mean filter and (c) median filter using a 7×7 window. (d) Area-open removing structures with area smaller than 25.

The Code Fragment 2 illustrates how to perform the area-open filter and display the filtered image on the Adessowiki environment.

Code Fragment 2. Area-open filter and image restitution using the max-tree.

```
# Area-open filter
mxt.areaOpen(50)
#Image restitution
area_open = mxt.getImage()
#Image display in adessowiki
adshow(area_open, 'Area-open - 25')
```

B. Bounding-box filter

In this example, we show how to use the max-tree for license plate location using a bounding-box filter. In this case we filter all the max-tree nodes whose bounding-box height is not between 13 and 25 pixels, and the width is not between 7 and 17 pixels, and the rectangularity ratio is greater than 0.45. The license plate image and the result of the filtering procedure are depicted in Fig. 7. This example shows that a filtering step greatly simplified the image leaving practically just the license plate characters.

(a)

(b)

Fig. 7. (a) Original image, (b) after the bounding-box filter.

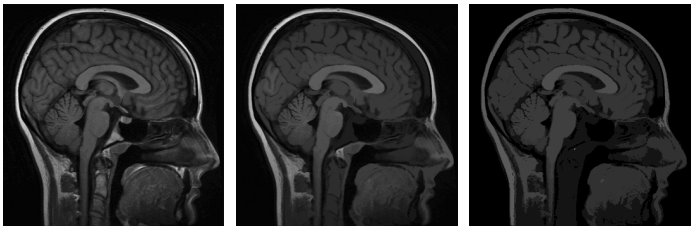
We do not have a filter that implements this filter, so with have to code it ourselves. The Code Fragment 3 illustrates the code to perform the bounding-box filtering.

Code Fragment 3. Bounding-box filter.

```
#Size and shape thresholds
Wmin,Wmax = 7,17
Hmin,Hmax = 13,25
rr = 0.45

#Computing bounding-box lengths from the
#attributes stored in NA
dx = mxt.node_array[7,:] - mxt.node_array[6,:]
dy = mxt.node_array[10,:] - mxt.node_array[9,:]
area = mxt.node_array[3,:]
RR = 1.0*area/(dx*dy)

#Selecting nodes that fit the criteria
nodes = (dx>Hmin) & (dx<Hmax) & (dy > Wmin) &
        (dy < Wmax) & (RR > rr)

#Filtering
mxt.contractDR(nodes)
```

C. Maximal Max-tree Simplification

In order to simplify the max-tree and still preserve its topology, the maximal max-tree simplification (MMS) methodology [11] may be employed. It consists in applying an extinction filter followed by the maximal max-tree simplification filter. This methodology guarantees that at the end the number of max-tree nodes is bounded between the number of tree leaves plus one and two times the number of leaves. The application of this methodology to a sagittal brain MR slice is illustrated in Fig. 8. The original max-tree has 14312 nodes, after the area extinction filter set to preserve 8 leaves, the tree has 614 nodes. After the MMS filter, the resulting max-tree has only 16 nodes and many relevant structures like scalp, brain stem, cerebellum, and the corpus callosum are still present in the image as can be seen in Fig. 9. This methodology is very useful for simplifying the max-tree, so its nodes can be used in association with machine learning techniques faster than using the entire set of nodes of the original max-tree.



(a) (b) (c)

Fig. 8. (a) Original image. (b) Result of the area extinction filter set to preserve 8 leaves. (c) Result of the MMS filter.

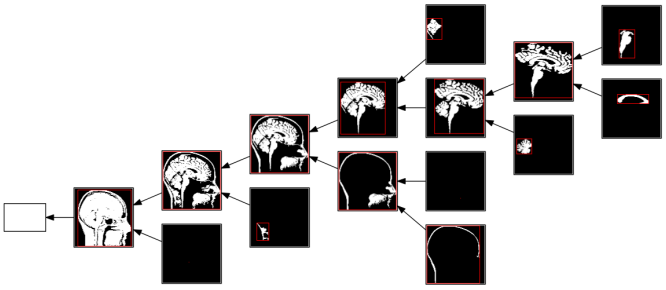


Fig. 9. Max-tree after the MMS methodology.

The Code Fragment 4 depicts the code to apply the MMS methodology.

Code Fragment 4. MMS methodology code.

```
#number of leaves to preserve
n = 8
# Height extinction values computation
ext = mxt.computeExtinctionValues(mxt.computeHeight(),
      "height")
#Extinction filter
mxt.extinctionFilter(ext,n)
#MMS filter
mxt.mmsT()
#Displaying the resulting max-tree graph
mmgraphviz(mxt.generateCCGraph(parent_scale = True))
```

REFERENCES

[1] R. A. Lotufo, R. C. Machado, A. Körbes, and R. G. Ramos, "Adessowiki on-line collaborative scientific programming platform," in *WikiSym '09: Proceedings of the 5th International Symposium on Wikis and Open Collaboration*, New York, NY, USA, 2009, pp. 1–6.

[2] G. Bradski and A. Kaehler, "Opencv," *Dr. Dobbs journal of software tools*, vol. 3, 2000.

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

[4] C. Vachier, "Extinction value: a new measurement of persistence," in *IEEE Workshop on Nonlinear Signal and Image Processing*, vol. I, 1995, pp. 254–257.

[5] R. Jones, "Connected filtering and segmentation using component trees," *Computer Vision and Image Understanding*, vol. 75, no. 3, pp. 215–228, 1999.

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

[7] J. C. Serra and P. Salembier, "Connected operators and pyramids," in *Image algebra and morphological image processing IV*, vol. 2030. International Society for Optics and Photonics, 1993, pp. 65–76.

[8] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," in *British Machine Vision Conference*, 2002, pp. 384–393.

[9] R. Souza, L. Rittner, R. Lotufo, and R. Machado, "An array-based node-oriented max-tree representation," in *2015 IEEE International Conference on Image Processing (ICIP)*, Sept 2015, pp. 3620–3624.

[10] S. van der Walt, S. Colbert, and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science Engineering*, vol. 13, no. 2, pp. 22–30, March 2011.

[11] R. Souza, L. Rittner, R. Machado, and R. Lotufo, "Maximal max-tree simplification," in *22nd International Conference on Pattern Recognition*, Aug 2014, pp. 3132–3137.