DOI: 10.1016/j.softx.2017.03.001

SoftwareX 6 (2017) 81–84

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

iamxt: Max-tree toolbox for image processing and analysis

Roberto Souza [a,*], Letícia Rittner [a], Rubens Machado [b], Roberto Lotufo [a]

[a] School of Electrical and Computer Engineering, University of Campinas, Campinas, Brazil

[b] Center for Information Technology Renato Archer, Campinas, Brazil

ARTICLE INFO

Article history:
Received 14 August 2015
Received in revised form
1 September 2016
Accepted 14 March 2017

Keywords:
Max-tree
Component tree
Mathematical morphology

ABSTRACT

The iamxt is an array-based max-tree toolbox implemented in Python using the NumPy library for array processing. It has state of the art methods for building and processing the max-tree, and a large set of visualization tools that allow to view the tree and the contents of its nodes. The array-based programming style and max-tree representation used in the toolbox make it simple to use. The intended audience of this toolbox includes mathematical morphology students and researchers that want to develop research in the field and image processing researchers that need a toolbox simple to use and easy to integrate in their applications.

Code metadata

Current code version	v0.1
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-15-00049
Legal Code License	BSD 2-Clause License
Code versioning system used	git
Software code languages, tools, and services used	python, C++, OpenMP
Compilation requirements, operating environments & dependencies	NumPy, OpenCV, gvgen, dot
If available Link to developer documentation/manual	http://adessowiki.fee.unicamp.br/adesso/wiki/iamxt/view/
Support email for questions	iamxt@googlegroups.com, roberto.medeiros.souza@gmail.com

1. Motivation and significance

The max-tree [1] is a data structure that represents a gray-scale image through the hierarchical relationship of its connected components. A simple illustration of the max-tree is depicted in Fig. 1. Filtering an image using the max-tree consists in removing some of its nodes. It has been used for attribute filtering and interactive visualization [2], morphological filtering [3], feature extraction with Maximally Stable Extremal Regions (MSER) [4], interactive collection of training samples [5], image segmentation [6,7], among other applications. Despite its large applicability, the max-tree is still little known outside the mathematical morphology community.

The motivation for this work is to provide to the scientific community a max-tree toolbox that implements state of the art algorithms, a large set of connected filters [8], and disposes of many visualization tools to display the tree nodes and the connected components they represent. The toolbox is intended for developing applications, and teaching purposes.

There are a few public max-tree libraries. The SDC morphology toolbox for MATLAB and Python [9] has a limited number of max-tree processing functions, and the source code is not available. The PINK image processing library has the max-tree algorithm and Python bindings. The Milena implementation [10] in C++ also has Python bindings. It is not focused on max-trees, but it has many max-tree construction algorithms implemented. Also, it lacks the max-tree visualization features that our toolbox provides, which is very useful for understanding and designing methods. For instance, built on top of our toolbox Tavares et al. [11] developed an interactive visualization demo that displays not only the image, but its max-tree. This demo is very useful for understanding the max-tree. Westenberg et al. [2] provide a max-tree demo for interactive volume visualization with attribute filtering.

To the best of our knowledge, there is no max-tree code available that gathers so many features as our iamxt toolbox, and is written in a high level programming language, such as Python,

* Corresponding author.
E-mail address: roberto.medeiros.souza@gmail.com (R. Souza).

http://dx.doi.org/10.1016/j.softx.2017.03.001

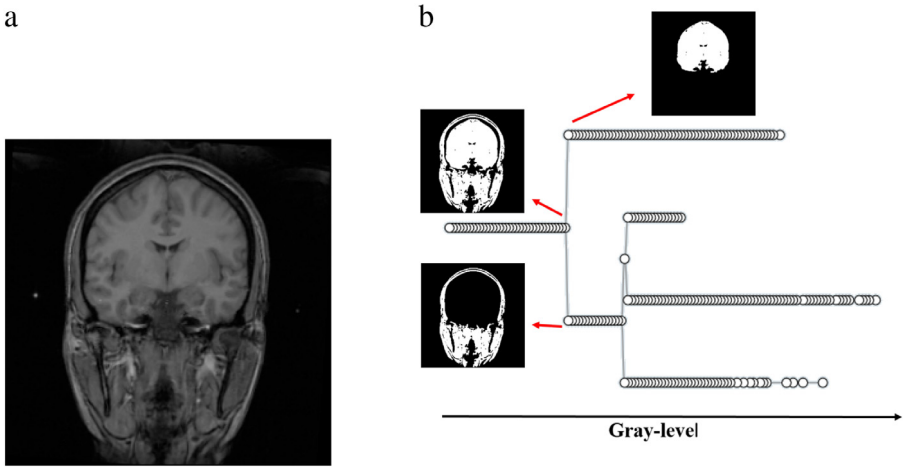

Fig. 1. (a) Brain image. (b) Max-tree illustration of (a). The arrows point to some of the connected components corresponding to the max-tree nodes.

using an array programming style. The toolbox is well documented and with many usage examples, making easier its use even for people with not much knowledge of mathematical morphology. Our iamxt toolbox can be employed in many applications and be used to answer many research questions like the presence of a common topology in brain magnetic resonance images of healthy subjects.

This paper is organized as follows: Section 2 describes the software implementation and its features. Section 3 illustrates some examples of the toolbox features. Section 4 describes the impact we expect iamxt will have in the scientific community. Section 5 concludes this paper.

2. Software description

In this section we describe the languages and libraries used to implement the iamxt toolbox. We also describe the structure of the toolbox and the algorithms implemented.

2.1. iamxt implementation

The iamxt toolbox was implemented in Python using the NumPy [12] library, which allows fast array processing. Our implementation uses the array programming style, which avoids using explicit loops as much as possible, but the portions of the code in which we were not able to avoid it were implemented in C++ and wrapped using SWIG [13]. The parallel loops where optimized using OpenMP [14]. The toolbox also depends on other libraries for rendering graphs. A complete list of the dependencies and instructions on how to install the toolbox are available at: https://github.com/rmsouza01/iamxt.

2.2. iamxt structure

The toolbox was structured using the object oriented programming paradigm. The iamxt is divided in two classes: the base class called *MorphTreeAlpha* and the derived class *MaxTreeAlpha*. The *MorphTreeAlpha* class has the methods for walking on the tree, such as retrieving nodes ancestors, descendants and children. It has methods for drawing the tree and parts of it, and it also has methods that are inherent to morphological trees other than the max-tree, such as the tree of shapes [15]. The *MaxTreeAlpha* class implements methods specific of max-trees.

The toolbox has a total of 30 methods: 4 for walking on the tree, 2 filtering algorithms, 7 max-tree filters, 4 for drawing the tree and parts of it, 9 for computing attributes from the max-tree nodes, 2 auxiliary methods, and 2 reconstruction methods: one for recovering the image corresponding to the max-tree and the other to recover the connected component corresponding to a node.

2.3. iamxt algorithms

The max-tree construction algorithm implemented in the toolbox is the one based on the union-find with level compression algorithm, which is described in [16]. The filtering algorithm implementation is the one proposed in [17], which uses a different array-based parent pointer max-tree representation, called node oriented max-tree, which is more memory efficient than the usual max-tree representation structures and easier for manipulating the max-tree nodes. Our toolbox uses this structure to represent the max-tree. The toolbox also implements many methods to extract attributes from the max-tree nodes and to filter the max-tree, such as extinction values [18], the hmax filter [1], the area-open [19], the vmax [18], the extinction filter [20], the maximal max-tree simplification (MMS) [21], among others that are detailed in the toolbox documentation.

3. Illustrative examples

3.1. Bounding-box filtering

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, the width is not between 7 and 17 pixels, and the rectangularity ratio is larger than 0.45. The license plate image and the result of the filtering procedure are depicted in Fig. 2. The code to perform that is shown in the Code Fragment 1, and the comments describe the code. This example shows some of the power of the max-tree. A simple filtering procedure simplified greatly the image leaving practically just the license plate characters.

Code Fragment 1: Bounding-box filter.

```
import iamxt
import numpy as np
#Structuring element, connectivity-8
Bc = np.ones((3,3),dtype = bool)
mxt = iamxt.MaxTreeAlpha(img,Bc) # max-tree construction

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

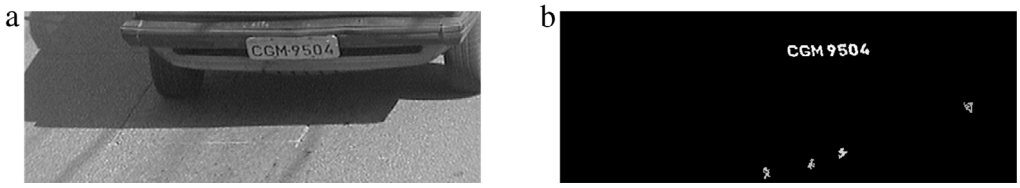

Fig. 2. (a) Original image, (b) after the bounding-box filter.

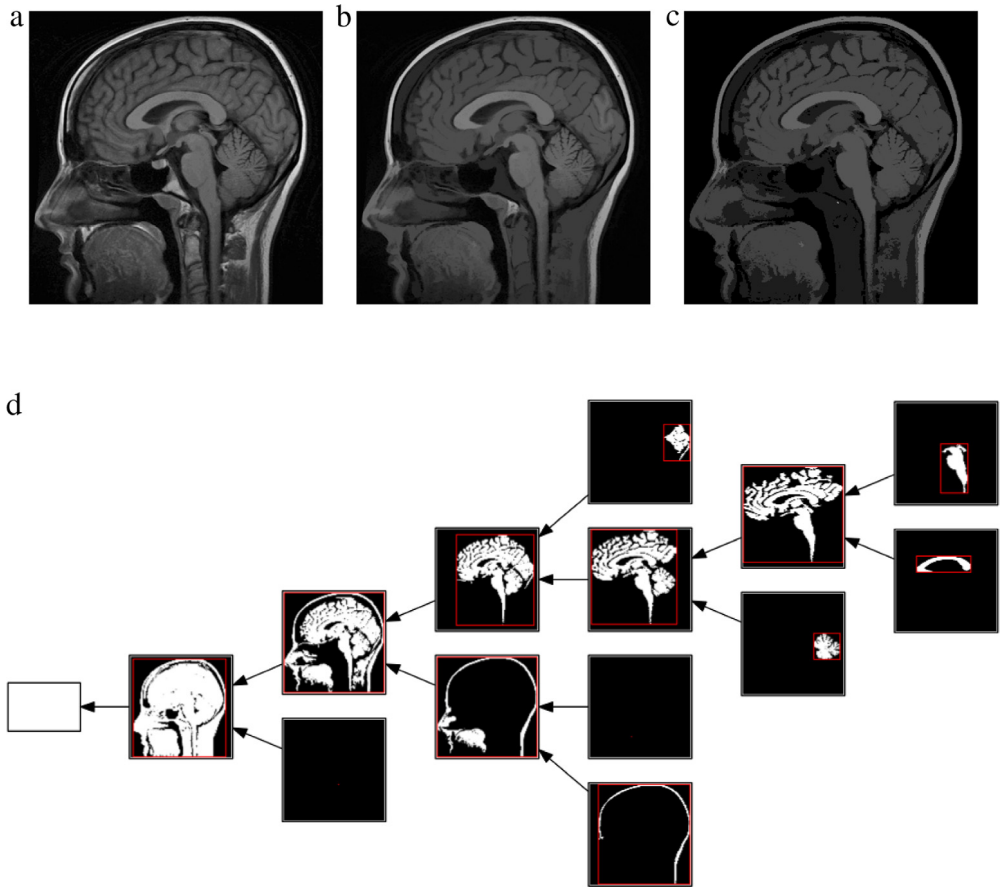

Fig. 3. (a) Original image, (b) after EF (c) after MMS. (d) Resulting max-tree.

```
11
   #Computing bounding-box lengths from the
13 #attributes stored in NA
   dx = mxt.node_array[7,:] - mxt.node_array[6,:]
15 dy = mxt.node_array[10,:] - mxt.node_array[9,:]
   area = mxt.node_array[3,:]
17 RR = 1.0*area/(dx*dy)

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

23 #Filtering
   mxt.contractDR(nodes)
25 img_filtered = mxt.getImage()
```

3.2. Maximal max-tree simplification methodology

In this example we illustrate the maximal max-tree simplification (MMS) methodology [21]. This methodology consists of setting the number of tree leaves using an extinction filter (EF) followed by the MMS filter. It guarantees that at the end of the procedure the number of max-tree nodes is bounded between the number of leaves plus one and two times the number of leaves. This methodology can be used for image segmentation and object recognition [21]. This methodology applied to a 2D brain MR image set to preserve 7 leaves is depicted in Fig. 3.

3.3. Processing time

A brief notion of the toolbox processing time is presented here, a more detailed analysis is presented in [17]. The experiments were performed on a 4-core virtual machine running in the Intel Xeon X5675 server with clock of 3.06 GHz. We chose the three 256×256 pixels sample images shown in Fig. 4. We measured the average times for building the max-tree using a 8-neighborhood, filtering the max-tree using an area-open filter set to remove nodes with area smaller than 500, and reconstructing the image. The images were interpolated up to 1024×1024 pixels, so we can have an idea of how the processing times evolve according to the images size. The average processing times are summarized in Table 1.

4. Impact

Although very powerful, mathematical morphology tools, such as the max-tree, are not very known/used outside the

Fig. 4. Sample images.

Table 1
Average max-tree construction, filtering and image restitution processing times in milliseconds.

Dimensions	Construction	Filtering	Restitution	Total
256 × 256	17.7	1.7	0.08	19.5
512 × 512	72.7	2.6	0.3	75.6
1024 × 1024	216.7	4.3	1.3	222.3

morphology community, one of the reasons for that may be the fact that mathematical morphology algorithms can be very complex to understand and implement. The iamxt is an educative toolbox, where users that are not specialists on max-trees and mathematical morphology can learn and develop applications, and mathematical morphology researchers can benefit from our toolbox. The impact of providing a simple to use, efficient, and easy to extend open-source code is that the number of max-tree users will grow, and they will extend our toolbox and use it to develop their applications.

Our toolbox has been used in our research [21,20,17,11]. Also, we have international collaborators that use our toolbox for remote sensing applications [22]. Also, iamxt is used to teach courses at graduate level at the University of Campinas.

5. Conclusions

We provided a max-tree toolbox implemented in Python and NumPy using the array-based programming style, where the explicit loops that were not suitable to be implemented using this style were optimized using C++ and OpenMP. State of the art algorithms available in the literature were implemented in the toolbox. That allied to the max-tree visualization routines that we provide and the fact that it was implemented in an easy and open-source programming language, such as Python/NumPy, makes our toolbox suitable for developing applications, and teaching purposes. In our next release of the toolbox we will incorporate the methodology that builds a max-tree from a another tree-based image representation proposed in [3].

Acknowledgments

The authors would like to thank FAPESP grants 2013/23514-0 and 2013/07559-3 and CNPq grant 311228/2014-3.

References

[1] Salembier P, Oliveras A, Garrido L. Antiextensive connected operators for image and sequence processing. IEEE Trans Image Process 1998;7(4):555–70.
[2] Westenberg M, Roerdink J, Wilkinson M. Volumetric attribute filtering and interactive visualization using the max-tree representation. IEEE Trans Image Process 2007;16(12):2943–52.
[3] Xu Y, Géraud T, Najman L. Morphological filtering in shape spaces: Applications using tree-based image representations. In: 2012 21st International Conference on Pattern Recognition (ICPR). IEEE; 2012. p. 485–8.
[4] Matas J, Chum O, Urban M, Pajdla T. Robust wide-baseline stereo from maximally stable extremal regions. In: British machine vision conference. 2002. p. 384–93.
[5] Ouzounis G, Gueguen L. Interactive collection of training samples from the max-tree structure, in: 18th IEEE international conference on image processing; 2011, pp. 1449–1452.
[6] Jones R. Connected filtering and segmentation using component trees. Comput Vis Image Underst 1999;75(3):215–28.
[7] Naegel B, Passat N. Interactive segmentation based on component-trees. Image Processing On Line 2014;89–97.
[8] Salembier P, Serra J. Flat zones filtering, connected operators, and filters by reconstruction. IEEE Trans Image Process 1995;4(8):1153–60.
[9] Dougherty E, Lotufo R. Hands-on morphological image processing. SPIE tutorial texts in optical engineering, vol. TT59. SPIE Publications; 2003.
[10] Levillain R, Géraud T, Najman L. Milena: Write generic morphological algorithms once, run on many kinds of images. Mathematical morphology and its application to signal and image processing, vol. 5720. 2009. p. 295–306.
[11] Tavares L, Souza R, Rittner L, Machado R, Lotufo R. Interactive max-tree visualization tool for image processing and analysis, in: 2015 International conference on image processing theory, tools and applications; 2015. p. 119–24.
[12] van der Walt S, Colbert S, Varoquaux G. The numpy array: A structure for efficient numerical computation. Comput Sci Eng 2011;13(2):22–30.
[13] Beazley D. SWIG: An easy to use tool for integrating scripting languages with C and C++. In: Proceedings of the 4th conference on USENIX Tcl/Tk workshop, 1996-Volume 4, USENIX Association; 1996. p. 15.
[14] Dagum L, Menon R. OpenMP: An industry-standard API for shared-memory programming. IEEE Comput Sci Eng 1998;5(1):46–55.
[15] Monasse P, Guichard F. Fast computation of a contrast-invariant image representation. IEEE Trans Image Process 1998;9:860–72.
[16] Carlinet E, Geraud T. A comparative review of component tree computation algorithms. IEEE Trans Image Process 2014;23(9):3885–95.
[17] Souza R, Rittner L, Lotufo R, Machado R. An array-based node-oriented max-tree representation. In: 2015 IEEE International Conference on Image Processing (ICIP). IEEE; 2015. p. 3620–4.
[18] Vachier C. Extinction value: a new measurement of persistence. IEEE workshop on nonlinear signal and image processing, vol. I. 1995. p. 254–7.
[19] Vincent L. Morphological area openings and closings for grey-scale images. In: Toet A Y-LO, Foster D, Heijmans H, Meer P, editors. Shape in picture, vol. 126. Berlin, Heidelberg: Springer; 1994. p. 197–208.
[20] Souza R, Rittner L, Machado R, Lotufo R. A comparison between extinction filters and attribute filters. In: Mathematical morphology and its applications to signal and image processing. 2015. p. 63–74.
[21] Souza R, Rittner L, Machado R, Lotufo R. Maximal max-tree simplification. In: 2014 22nd International conference on pattern recognition; 2014, p. 3132–37.
[22] Ghamisi P, Souza R, Benediktsson J, Zhu X, Rittner L, Lotufo R. Extinction profiles for the classification of remote sensing data. IEEE Trans Geosci Remote Sens 2016;54(10):5631–45.