SoftwareX 16 (2021) 100855

Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

AGAT: Building and evaluating binary partition trees for image segmentation

Jimmy Francky Randrianasoa [a], Camille Kurtz [b], Éric Desjardin [c], Nicolas Passat [c,*]

[a] EPITA Research and Development Laboratory (LRDE), France
[b] Université de Paris, LIPADE EA 2517, 75006 Paris, France
[c] Université de Reims Champagne Ardenne, CReSTIC EA 3804, 51100 Reims, France

ARTICLE INFO

Article history:
Received 21 July 2021
Received in revised form 16 October 2021
Accepted 18 October 2021

Keywords:
Binary partition tree
Hierarchical modeling
Image/graph processing
Quality evaluation

ABSTRACT

AGAT is a Java library dedicated to the construction, handling and evaluation of binary partition trees, a hierarchical data structure providing multiscale partitioning of images and, more generally, of valued graphs. On the one hand, this library offers functionalities to build binary partition trees in the usual way, but also to define multifeature trees, a novel and richer paradigm of binary partition trees built from multiple images and/or several criteria. On the other hand, it also allows one to manipulate the binary partition trees, for instance by defining/computing tree cuts that can be interpreted in particular as segmentations when dealing with image modeling. In addition, some evaluation tools are also provided, which allow one to evaluate the quality of different binary partition trees for such segmentation tasks. AGAT can be easily handled by various kinds of users (students, researchers, practitioners). It can be used as is for experimental purposes, but can also form a basis for the development of new methods and paradigms for construction, use and intensive evaluation of binary partition trees. Beyond the usual imaging applications, its underlying structure also allows for more general developments in graph-based analysis, leading to a wide range of potential applications in computer vision, image/data analysis and machine learning.

Code metadata

Current code version	v2.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-21-00135
Code Ocean compute capsule	NA
Legal Code License	Cecill-B
Code versioning system used	git
Software code languages	Java SE 8
Compilation requirements, operating environments & dependencies	JDK 1.8 (for Eclipse developers, .project files are also provided)
If available Link to developer documentation/manual	https://github.com/yonmi/AGAT2.0
Support email for questions	agat@univ-reims.fr

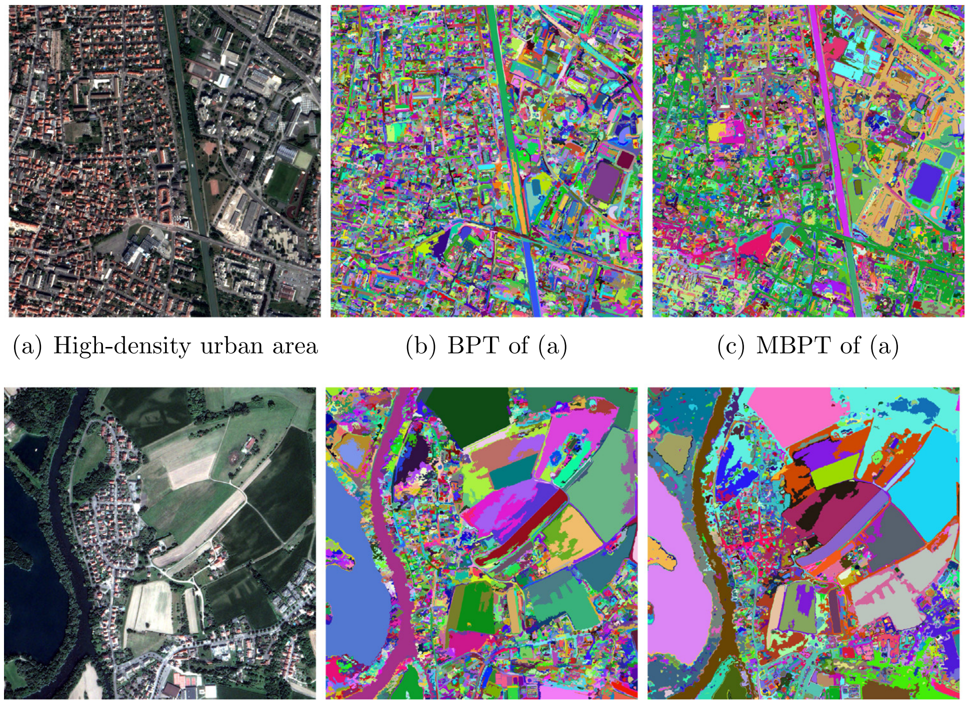
1. Introduction

Due to the rapid progress in the development of imaging sensors, the produced images are becoming increasingly complex, both in size and in semantics. This is the case for example in medical and biological imaging, remote sensing, material sciences, and more generally, in computer vision applications. In such domains, the data that are now handled require to be processed at various levels of detail, i.e. at various scales, in particular with the purpose of tackling computational and semantic analysis issues.

For tackling these issues, two main paradigms have been investigated over the last decades. On the one hand, the paradigm of multiscale analysis, that intrinsically relies on the underlying notion of scale space [1], consists in observing an image at “different distances”, then focusing on the details available at each distance. This led to various multiscale analysis approaches for image description (e.g. SIFT, pyramids [2]). On the other hand, the paradigm of image partitioning popularized under the terminology of “superpixels” consists in creating connected clusters

* Corresponding author.
E-mail address: nicolas.passat@univ-reims.fr (Nicolas Passat).

https://doi.org/10.1016/j.softx.2021.100855



(a) High-density urban area (b) BPT of (a) (c) MBPT of (a)

(d) Low-density urban area (e) BPT of (d) (f) MBPT of (d)

Fig. 1. (a, d) Two VHSR satellite images ($2\,000 \times 2\,000$ pixels) at a spatial resolution of 60 cm sensed by the Pléiades satellite and covering different areas. (b, e) Partitioning results from traditional binary partition trees computed from (a) and (d), respectively (23 500 and 5 000 regions, respectively), using one criterion: C_{color}. (c, f). Partitioning results from multifeature binary partition trees computed from (a) and (d), respectively (23 500 and 5 000 regions, respectively), using 4 criteria: C_{color}, C_{elong}, C_{ndvi}, C_{ndwi}.

of homogeneous pixels of certain size within an image, in order to reduce its space complexity without altering the carried visual information. Superpixels were then developed in many variants, mainly for pre-segmentation purposes (e.g. SLIC [3], waterpixels [4]).

At the convergence of these two paradigms, the notion of hierarchical image model was developed, in particular in the field of mathematical morphology, leading to a rich family of graph-based data structures, generally defined as trees (i.e. connected, acyclic graphs), designed for modeling images as hierarchies of partitions. Non-exhaustively, the most frequently used trees are the component-tree [5] (which models a gray-level image as the Hasse diagram of the binary connected components of all the threshold sets, with respect to the inclusion relation) and its multivalued variant [6]; the tree of shapes [7] (which is a self-dual variant of the component-tree, that gathers information obtained by thresholding the image in both top-down and bottom-up ways) and its multivalued variant [8]; the watershed tree [9] (that derives from hierarchical watersheds [10], and allows to model in a hierarchical way the saliency maps derived from the gradient of an image), the binary partition tree [11] (that we consider in this article), and some variants such as the α-tree [12] (that derives from the concept of constrained connectivity [13]). Many other hierarchical models (including not only trees but also more complex directed acyclic graph structures, e.g. asymmetric hierarchies, braids of partitions, component-graphs or component-hypertrees [14–17] that generalize/extend the tree structures beyond their usual topological and/or spectral hypotheses of definition) were provided. A whole discussion is beyond the scope of this article; the interested reader can refer to [18] for a recent survey.

The construction of the trees mentioned above is generally expressed as a graph partitioning problem. More generally, the induced methods lie in the same family as optimization methods on graphs, which are often involved in imaging problems, but can also tackle a wider family of problems, if the data to be processed are discrete and can be structured via a binary relation (e.g., in mesh-based applications, structured data processing, etc.). In particular, strong links exist between the concepts of hierarchical models, saliency maps and spanning trees in graphs [19].

Most of the hierarchical models (e.g. the component-tree or the tree of shapes) can be built from an image, without considering any additional information. Such trees can be seen as pure image modeling data structures, that embed an image into an alternative space, where it can be handled and modified thanks to image processing paradigms. By contrast, the binary partition tree (BPT, for brief) [11], is built from two kinds of information: (1) the intrinsic information carried by the input image (or, more generally, the input valued graph), and (2) an extrinsic—generally user-defined/application-based—information that determines which criteria should be considered for describing the input data in a multiscale way. This expert/domain-based information is crucial in certain application fields. In other words, the binary partition tree is not only an image-oriented but also a knowledge-based data structure. As a consequence, it can be relevantly involved in image analysis tasks that require the embedding of expert-defined priors and knowledge. For instance, the binary partition tree is quite popular in remote sensing applications [20–26], where the way to decompose an image depends on its content (e.g. an urban area vs. a wild forest zone) but also on the purpose of the analysis (e.g. classifying the buildings vs. observing pollution effects).

Various software programs and libraries are available for hierarchical image model handling. GraphBPT[1] [27] is especially designed for building and using binary partition trees. More general-purposed libraries, for instance scikit-image[2], Higra [28] or the

[1] https://github.com/ash-aldujaili/GraphBPT.

[2] https://scikit-image.org.

trees-lib library[3] deal with wider issues. In particular the last two mentioned allow for the construction of binary partition trees or α-trees, that can be seen as a variant of binary partition trees. Other libraries, namely Olena[4] or LibTIM[5] (used e.g. in [29]) are mainly geared towards the construction of the so-called component-trees and/or trees of shapes, but could probably support further extensions for handling binary partition trees.

In this article, we introduce AGAT, a Java library specifically dedicated to the binary partition tree. Similarly to previous libraries (e.g. GraphBPT) it proposes a construction framework of traditional binary partition trees. In addition, it also proposes a way of building a more general family of binary partition trees, the so-called multifeature binary partition trees [30]. From a structural point of view, these binary partition trees do not differ from the classical ones. They differ actually in the way they are built. Indeed, by contrast to the usual construction algorithm that relies on a single clustering criterion and a single image, the construction of the multifeature binary partition trees relies on the collaboration between various clustering criteria and/or allows to handle many images of a same scene.

Another contribution of AGAT compared to the already available libraries is the proposal of various tools for evaluating the quality of a binary partition tree (or equivalently, the quality of the meta-parametrization of its construction) with respect to object segmentation purposes. Indeed, evaluating the quality of a hierarchical image model is an important—but infrequently considered—topic as a prerequisite to its actual involvement for real applications [31–33].

2. Software description

In this section, we describe the structure of the AGAT library and the major functionalities implemented.

2.1. Software architecture

AGAT is composed of three modules (`Image`, `BinaryPartitionTree`, `TreeEvaluation`) required for the construction of binary partition trees from images, their handling and their evaluation. The three modules are coded in pure Java 8, taking advantage of the latest language innovations. They also contain some external Java libraries, encapsulated in the projects in the form of .jar files (mainly for input/output).

The `Image` module is independent. It contains basic tools for manipulating and processing raster images (e.g. equalization, conversion, channel management, inputs/outputs). Images are classically encoded via the `BufferedImage` class which is part of the Abstract Window Toolkit (AWT), a graphics library commonly used by the Java community.

The `BinaryPartitionTree` module is dependent on the `Image` module. It contains various functionalities for building binary partition trees in a usual way, but also to define multifeature trees from consensus of multiple images and/or multiple criteria. This module also allows to handle the trees, for instance by defining/computing tree cuts that can be interpreted in particular as segmentations when dealing with images. In AGAT, binary partition trees are modeled and analyzed through their hierarchical representations. They are encoded as trees, where each node is a region of the image support. The main data structures of the library are thus a graph class, implemented as an adjacency list (required for the construction step of the binary partition trees relying on a region adjacency graph), and a tree class, classically implemented with inheritance relationships. To enable the multiple images and/or multiple criteria paradigm, data structures based on ordered lists of valued edges have been also implemented. These data structures are a bit specific because they should allow to choose efficiently the next edges to be selected during the construction of the trees. Multiple index systems and optimized iterators have thus been implemented to make it possible to speed up their scans. In addition, when the number of criteria to be considered is important, the lists are kept sorted sporadically, after a fixed number of modifications, which approximates the expected solution but enables better scaling.

The `TreeEvaluation` module is dependent on the `BinaryPartitionTree` module. It provides various classes related to the quantitative evaluation of the quality of a binary partition tree (or its meta-parametrization) with respect to object segmentation purposes. Both intrinsic and extrinsic analyses can be carried out. For a given binary partition tree (an object built from the `BinaryPartitionTree` module), the user can provide examples of ground-truth (defined as binary regions of interest in the image support) and quality metrics. The main data structures of this module make it possible to manage both information on the ground-truth segments provided by the user (e.g. coordinates of the bounding box, semantic labels, etc.) but also sub-trees of interest on which the analysis is carried out (to make processing faster by restricting it spatially and hierarchically).

2.2. Software functionalities

AGAT proposes a large amount of algorithms for the construction, the handling and the evaluation of binary partition trees (complementary technical details regarding the main data structure architecture can be found in [35, Appendix B]):

- **BPT construction:**
  - mono-image/mono-criterion [11]: tree construction from pixels or flat zones or a given partition, various generic (e.g. color: RGB, LAB, geometric: elongation, smoothness) and thematic (e.g. NDVI, NDWI) criteria are available;
  - multi-image/multi-criteria [30]: efficient process to establish different kinds of consensus among the adjacency lists (e.g. majority vote, most frequent, etc.), visualization of the conflict between metrics.

  The construction criteria and the consensus policy are provided by the user as parameters of the (multifeature) binary partition tree construction process (an example can be found in the code snippet of Listing 1).
- **BPT handling:**
  - definition of tree cuts from partition cardinality: flat or fitting with an input mask of an object of interest;
  - backup and restore trees from .h5 files, which is a Hierarchical Data Format (HDF) designed to store and organize large amounts of data and ensures compatibility with external tools;
  - export trees to .xml and .dot files, which allows visualization with external tools.
- **BPT quality evaluation:**
  - management of multiple ground-truth examples per image with potentially different semantic labels;
  - extraction of sub-trees of interest to speed up the analysis by restricting it spatially and hierarchically;

[3] https://github.com/pbosilj/trees-lib.
[4] https://www.lrde.epita.fr/wiki/Olena.
[5] https://github.com/bnaegel/libtim.

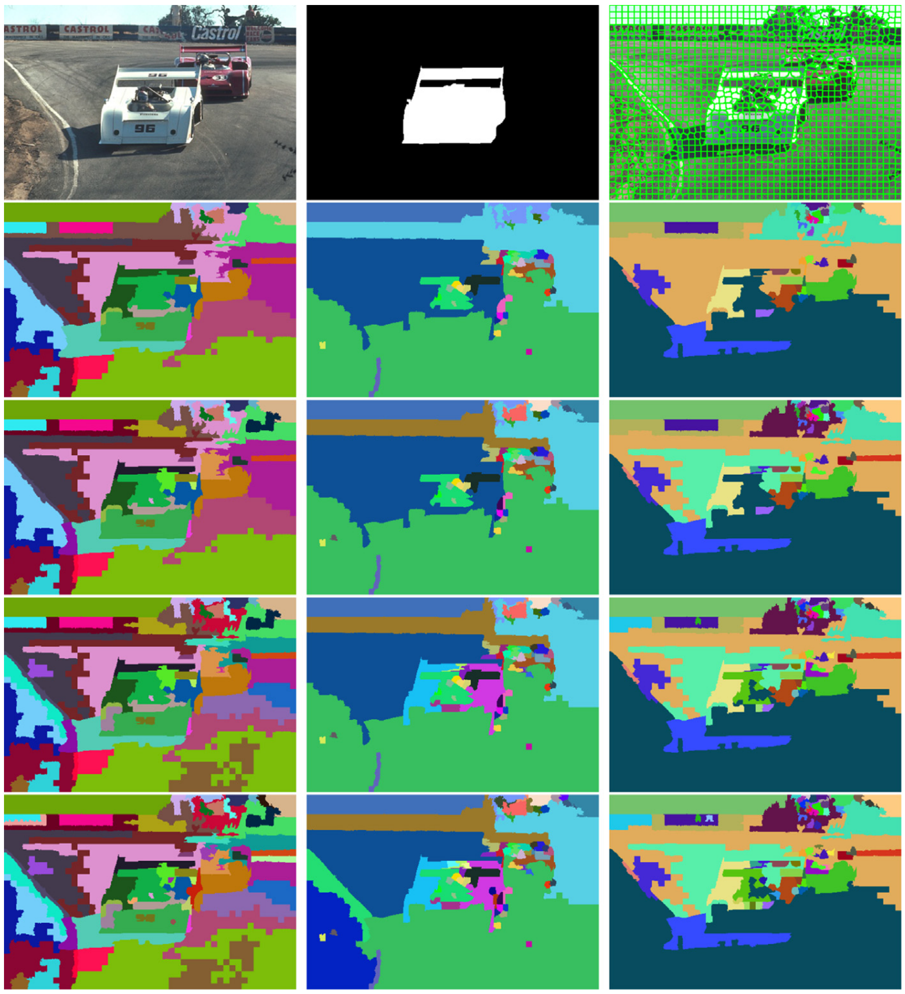


Fig. 2. Horizontal cuts of different binary partition trees on an image from the Grabcut dataset. First line: initial image, ground-truth and initial superpixel partition. First column: binary partition tree built with WSDM criterion. Second column: binary partition tree built with Min–Max criterion. Third column: binary partition tree built with MSE criterion. From second to fifth rows: horizontal cuts with 40, 50, 60, and 70 nodes. Each node is represented with a false color, for the sake of visualization.

– intrinsic analysis: combinatorial, quantitative, node assessment from the notion of pure and impure nodes matching with ground-truth examples [32,33];
– extrinsic analysis: home-made evaluation framework [31,33], implementation of other existing frameworks from the literature [34,36], evaluation metrics: F-measure/Dice, Jaccard index.

3. Illustrative examples

In this section, we present two illustrative examples of AGAT usage and performances. In the first example (Section 3.1), we show how AGAT can be used for building both traditional and multifeature binary partition trees, that can then be used for image partitioning. In the second example (Section 3.2), we show how AGAT can be employed for comparing the performances of various binary partition trees, in particular in the context of object segmentation. These illustrations were designed from the source codes and some codes snippets available in the provided GitHub repository.

3.1. Building a (traditional or multifeature) binary partition tree

The binary partition tree, such as defined in the pioneering article [11], was designed for hierarchically modeling one image with respect to one given criterion. Later on, the notion of multifeature binary partition tree, developed in [30], extended this initial paradigm, by allowing one to build a binary partition tree from one or many image(s) of the same scene with respect to one or many given criteria. The implementation of binary partition trees proposed in AGAT follows the latter paradigm of [30], and allows a fortiori to build traditional binary partition trees as defined in [11].

In this first illustrative example, we consider images in the context of remote sensing, and more precisely the analysis of very high spatial resolution (VHSR) satellite images, a domain where the concept of binary partition tree has been quite frequently and successfully involved over the last two decades.

The used dataset (courtesy LIVE, UMR CNRS 7263) was sensed over the town of Strasbourg (France) by the Pléiades satellite, in 2012. From this dataset, we sampled two VHSR images (2 000 × 2 000 pixels) representing:

- a complex high-density urban area (Fig. 1(a)) composed of different urban objects (e.g. individual houses, industrial buildings, parking lots, roads, shadows, water canals);
- a typical low-density urban area (Fig. 1(d)) composed of different geographical objects (e.g. crop fields, forests, bare soils, rivers).

These multispectral images are at a spatial resolution of 60 cm with 4 spectral bands (red, green, blue, near infrared).

Listing 1: Code snippet for building a binary partition tree using AGAT.

```
import java.awt.image.BufferedImage;
import java.util.Map.Entry;
import datastructure;
import metric.bricks.Metric.TypeOfMetric;
import multi.sequential.MBPT;
import multi.strategy.consensus.bricks.Consensus.ConsensusStrategy;
import ui.ImFrame;
import utils;
/**
 * Example to create a multi-feature binary partition tree using four criteria
 */
public class ExampleCreateAndCutMbpt {
   public static void main(String[] args) {

      BufferedImage image = ImTool.read("./dataset/VHSR-sample1.png");

      Tree tree = new MBPT(); //Create an empty tree
      ((MBPT)tree).registerImage(image); //Register the image(s)

      /* Choosing the consensus strategy to use */
      int consensusRange = 5; /* % defining the interval of the list to consider */
      int progressive = 1; /* interval defined proportionally to remaining number of adjacency links */
      ((MBPT) tree).setConsensusStrategy(ConsensusStrategy.SCORE_OF_RANK, consensusRange, progressive);

      /* Linking metrics to the image: four criteria are considered */
      ((MBPT)tree).linkMetricToAnImage(image, TypeOfMetric.RADIOMETRIC_MIN_MAX);
      ((MBPT)tree).linkMetricToAnImage(image, TypeOfMetric.NDVI);
      ((MBPT)tree).linkMetricToAnImage(image, TypeOfMetric.NDWI);
      ((MBPT)tree).linkMetricToAnImage(image, TypeOfMetric.SIMPLE_ELONGATION);

      tree.grow(); //Build the BPT

      /* Cutting */
      if(tree.hasEnded()) {
         int starting = 25, ending = 0, step = 5;
          CutResult cutResult = CutBPT.execute(tree, starting, ending, step);

          for(Entry<Integer, BufferedImage> entry: cutResult.regionImages.entrySet()) {
             int numberOfRegions = entry.getKey();
             BufferedImage partition = entry.getValue();
             ImTool.show(partition, ImFrame.IMAGE_DEFAULT_SIZE, numberOfRegions);
          }
      }
   }
}
```

We considered four criteria for building the binary partition trees, each one modeling either radiometric or geometrical information:

- C_{color}, defined as the increase of the range of the pixel intensity values for each radiometric band, induced by the putative fusion of incident regions;
- C_{ndvi}, that quantifies the difference of NDVI (Normalized Difference Vegetation Index, a standard indicator for the presence of green vegetation) between two adjacent regions;
- C_{ndwi}, that quantifies the difference of NDWI (Normalized Difference Water Index, a standard indicator for the presence of water) between two adjacent regions;
- C_{elong}, defined as the change of geometrical elongation, potentially induced by the fusion of two regions.

Fig. 1(b, e), illustrates the results of partitionings (induced by tree-cuts) obtained from traditional binary partition trees built by considering individually the first criterion. Fig. 1(c, f), illustrates the results of partitionings (induced by tree-cuts) obtained from multifeature binary partition trees built by considering collectively these four criteria.

A Java code snippet, presented in Listing 1, exemplifies how to obtain such results within AGAT. More extensive experiments related to the construction of various (multifeature) binary partition trees and the impact of these various kinds of trees in the context of remote sensing can be found in [30,32].

3.2. *Assessing/comparing the quality of various binary partition trees*

A wide literature has been devoted to segmentation based on binary partition trees. In this context, various criteria were investigated. The design of these criteria strongly influences the resulting trees and, equivalently, the research space for further segmentation, and thus the quality of the subsequent segmentation results. The literature dedicated to the evaluation of the quality of binary partition trees is not abundant. AGAT proposes (variants of) some of the most relevant approaches of the literature:

- an intrinsic quality analysis [33], that evaluates the relevance of a binary partition tree based on the combinatorial analysis of its nodes and their relationships with a binary ground-truth associated to the input image;
- an extrinsic quality analysis [33], that evaluates the ability of a binary partition tree to provide a cut that minimizes at best a given quality metric provided as hyperparameter with respect to the ground-truth associated to the input image;
- a framework adapted from [34] that provides the F-measure of the segmented object vs. the associated ground-truth, with respect to the size of inside/outside markers generated from the ground-truth associated to the input image;
- a framework adapted from [36] that aims at computing the best cuts composed of k nodes for each k in $[0, p]$ ($p > 0$), allowing to reconstruct the targeted segmentation.

In [33], extensive experiments were carried out with three commonly used datasets: Grabcut [37], Weizman [38] and VOC

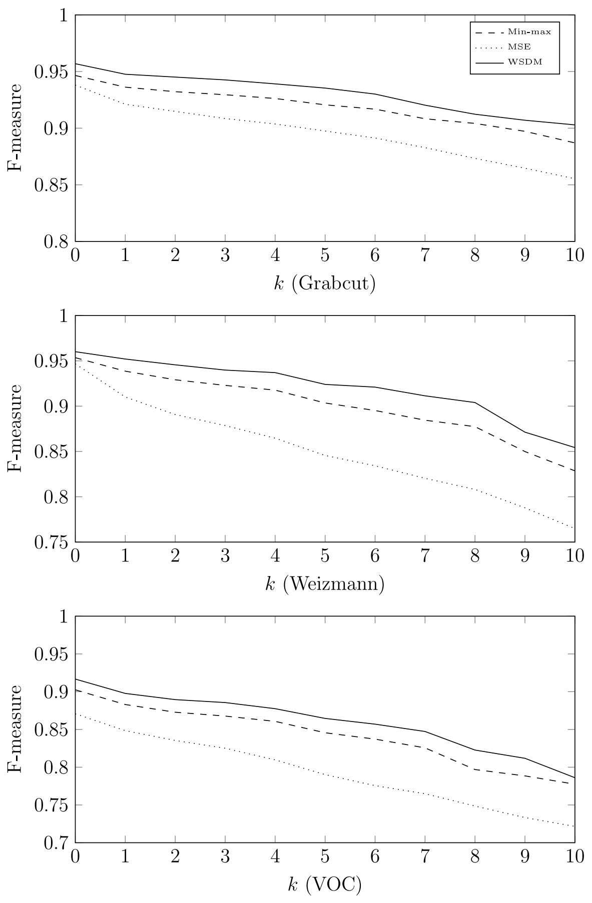


Fig. 3. F-measure of the segmentations obtained from various kinds of binary partition trees in the evaluation framework [34], with respect to the size $2k+1$ of the structuring elements used for erosion (background and foreground) of the ground-truth. Top: Grabcut. Middle: Weizmann. Bottom: VOC. Considering S as the segmentation result and G as the ground-truth, the F-measure/Dice score is defined as $2tp/(2tp+fp+fn)$ with $tp = |S \cap G|$, $fp = |S \setminus G|$ and $fn = |G \setminus S|$. (In this experiment, we kept the terminology of F-measure as used in the seminal paper [34].) The F-measure is related to the ratio of overlapping between two objects. It lies in [0, 1]; the closer to 1, the better the overlapping.

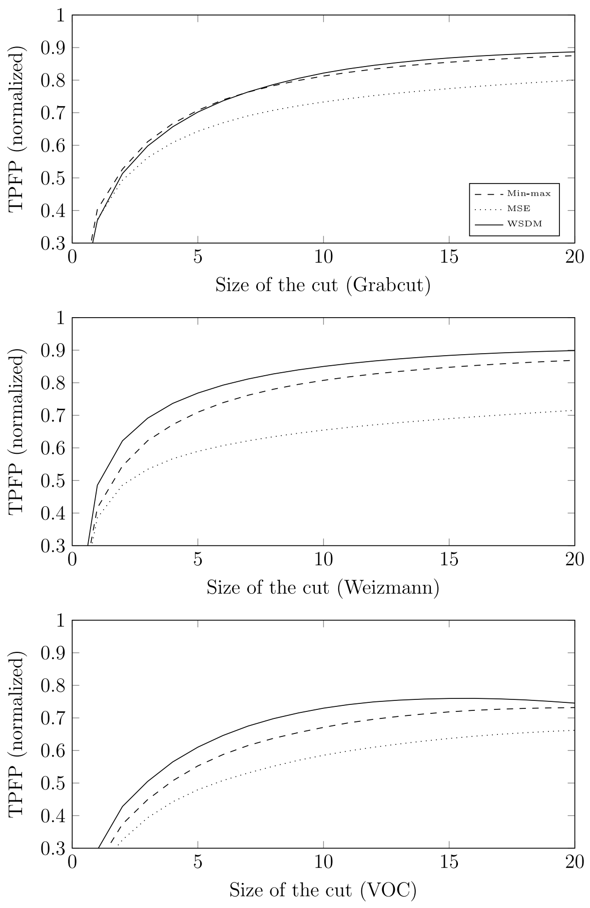


Fig. 4. Normalized (mean) value of the TPFP measure for the optimal cuts of a given size from various kinds of binary partition trees in the evaluation framework [36] Top: Grabcut. Middle: Weizmann. Bottom: VOC. Considering S as the segmentation result and G as the ground-truth, the (normalized) TPFP score is defined as $\frac{tp-fp}{|G|}$, with $tp = |S \cap G|$, $fp = |S \setminus G|$. The TPFP quantifies the trade-off between true and false positives. It lies in $(-\infty, 1]$; a 0 value means that there are as many true and false positives; the closer to 1, the better the result.

[39]. Examples of partitioning of an image from the Grabcut dataset are illustrated in Fig. 2. These partitions are obtained from three different binary partition trees, each one built with a given criterion, noted WSDM, Min–Max and MSE, respectively (see [33] for their formal definition).

Figs. 3–5 exemplify some metrics that can be computed from the various implemented evaluation frameworks, thus allowing to compare the relevance of distinct binary partition trees with respect to the considered data/ground-truth, here in the case of object segmentation. The results depicted in these figures quantitatively emphasize the superiority of one the three tested binary partition trees (namely the one built with WSDM criterion) over the other two ones. This is characterized by the fact that, in the three different experiments, WSDM leads to better values for the considered measures (the higher these values, the better the results). These results are qualitatively confirmed by the segmentations illustrated in Fig. 2.

A Java code snippet, presented in Listing 2, illustrates how to obtain such results within AGAT. The interested readers can find in [33] more extensive experiments related to the binary partition tree evaluation framework proposed in AGAT, applied in the context of natural images (Grabcut, VOC and Weizmann datasets).

4. Impact

There already exist libraries dedicated to hierarchical image models in general, and the binary partition trees in particular. However, AGAT is the first library that integrates the construction algorithms for traditional binary partition trees, but also for the recently introduced multifeature binary partition trees. This provides potential users with a unique opportunity to design and to experiment new and richer ways of building hierarchical models from complex/large (sets of) images. The success of the binary partition trees in the field of remote sensing over the last years is the proof of the relevance of this hierarchical model in the case of large and/or semantically complex images. Many domains involving such kinds of images (e.g. biological and (bio)medical imaging) could benefit from the opportunities offered by the binary partition trees, and then from the functionalities offered by AGAT.

Additionally, AGAT also embeds various tools for assessing the quality of binary partition trees. This quantitative evaluation framework can be very useful, for instance in the field of computer vision, where the partitioning of large/complex images is generally a prerequisite for high-level scene analysis tasks (e.g. object detection and recognition). In this regard, providing tools that can be involved in selection/learning of appropriate

Listing 2: Code snippet for evaluating a binary partition tree using AGAT.

```java
import java.awt.image.BufferedImage;
import java.io.PrintWriter;
import java.util;
import evaluation;
import standard.sequential.BPT;
import ui.ImFrame;
import utils.ImTool;
/**
 * Example to evaluate the quality of a BPT with intrinsic analysis
 */
public class ExampleBptIntrinsicAnalysis {
   public static void main(String[] args) {

      /* Create a tree */
      BufferedImage image = ImTool.read("./dataset/test_img.png");
      BufferedImage presegImg = ImTool.read("./dataset/test_img_slic.tif");
      BPT tree = new BPT(image);
      tree.setPreSegImage(presegImg);
      tree.grow();

      /* Extract the segments of reference from a ground truth image */
      String gtImgPath = "./dataset/test_img-gt.png";
      double alpha = 0.0;
      TreeMap<Integer, SegReference> segReferences = SegReference.extractSegmentsOfReference(gtImgPath, alpha, true);

      /* Visualizing the segments if wanted */
      SegReference.showBoundingBoxes(segReferences, image, ImFrame.IMAGE_STD_SIZE, "BB");

      /* Extract sub trees */
      STree extractedSubTrees[] = new STree[segReferences.size()];
      int gti = 0;
      for(Entry<Integer, SegReference> entry: segReferences.entrySet()){
         SegReference gt = entry.getValue(); // each segment of reference
         STree st = new STree(gti, tree, gt); st.index = gti++; // Create the subtree
         extractedSubTrees[st.index] = st;
      }

      /* Evaluate the sub trees */
      ArrayList<Eval> evalRes = new ArrayList<Eval>();
      for(int i = 0; i < extractedSubTrees.length; ++i) {
         STree st = extractedSubTrees[i];
         evalRes.add(st.eval(EvalType.INTRINSIC));
      }

      /* Print the intrinsic evaluations results */
      System.out.println("Intrinsic evaluation results: ");
      for(Eval res: evalRes) {
         res.printResults();
      }
   }
}
```

context-dependent meta-parameters for image decomposition is of great interest for the community. In particular, such frameworks proposed in combination with the binary partition tree construction algorithms could lead to consider the induced image partitioning results as a relevant alternative to the usual, non-hierarchical, image decompositions proposed by super-pixel paradigms.

Finally, it is worth mentioning that the binary partition trees, beyond their usefulness for image analysis task, are first of all a way of building partitions of graph-based structures, independently of their associated semantics. Based on this fact, AGAT may then be used for solving any graph partitioning problem provided that such partitionings are guided by one/many prior knowledge. This opens the way of the use of AGAT in a wide range of machine learning domains where the data are organized as graphs, i.e. with respect to usual binary relations.

5. Conclusions

We presented AGAT, a library dedicated to the construction, the handling and the evaluation of binary partition trees, which are tree data structures providing hierarchical partitioning of images and, more generally, valued graphs. AGAT contains also many standard and state-of-the-art algorithms in this domain. AGAT is the first library allowing for the construction of multi-feature binary partition trees, and it also gathers a large set of evaluation tools for traditional binary partition trees. From this point of view, it constitutes, to our knowledge, the most complete and up-to-date library dedicated to binary partition trees. It is composed of three self-contained Java modules. Code sources are available via GitHub and natively compatible for Linux, Mac, and Windows systems. They can be downloaded with a simple command: `git clone` https://github.com/yonmi/AGAT2.0.git.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the French *Agence Nationale de la Recherche, France* (Grants ANR-15-CE23-0009, ANR-17-CE23-0015 and ANR-18-CE23-0025) and by the American Memorial Hospital Foundation.

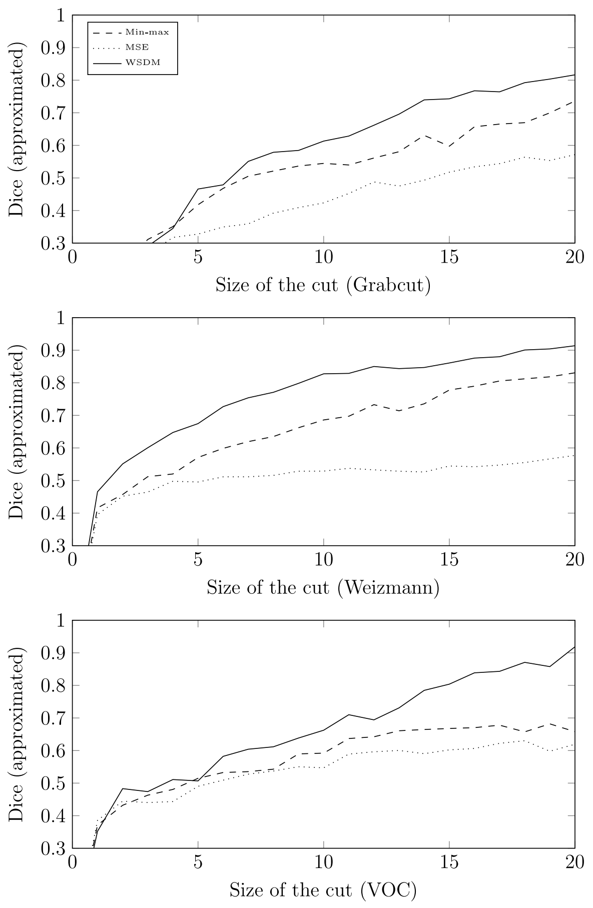


Fig. 5. Normalized (mean) value of the Dice measure for the (near-)optimal cuts of a given size from various kinds of binary partition trees. Top: Grabcut. Middle: Weizmann. Bottom: VOC. Considering S as the segmentation result and G as the ground-truth, the Dice score is defined as $2tp/(2tp+fp+fn)$ with $tp=|S\cap G|$, $fp=|S\setminus G|$ and $fn=|G\setminus S|$. The Dice score is related to the ratio of overlapping between two objects. It lies in [0, 1]; the closer to 1, the better the overlapping.

References

[1] Lindeberg T. Scale-space for discrete signals. IEEE Trans Pattern Anal Mach Intell 1990;12(3):234–54.
[2] Rey-Otero I, Delbracio M. Anatomy of the SIFT method. Imag Process Line 2014;4:370–96.
[3] Achanta R, Shaji A, Smith K, Lucchi A, Fua P, Süsstrunk S. SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Trans Pattern Anal Mach Intell 2012;34:2274–82.
[4] Cettour-Janet P, Cazorla C, Machairas V, Delannoy Q, Bednarek N, Rousseau F, et al. Watervoxels. Imag Process Line 2019;9:317–28.
[5] Salembier P, Oliveras A, Garrido L. Anti-extensive connected operators for image and sequence processing. IEEE Trans Image Process 1998;7:555–70.
[6] Kurtz C, Naegel B, Passat N. Connected filtering based on multivalued component-trees. IEEE Trans Image Process 2014;23(12):5152–64.
[7] Monasse P, Guichard F. Scale-space from a level lines tree. J Vis Commun Image Represent 2000;11:224–36.
[8] Carlinet E, Géraud T. MToS: A tree of shapes for multivariate images. IEEE Trans Image Process 2015;24(12):5330–42.
[9] Santana Maia D, Cousty J, Najman L, Perret B. Characterization of graph-based hierarchical watersheds: Theory and algorithms. J Math Imaging Vision 2020;62(5):627–58.
[10] Najman L, Schmitt M. Geodesic saliency of watershed contours and hierarchical segmentation. IEEE Trans Pattern Anal Mach Intell 1996;18:1163–73.
[11] Salembier P, Garrido L. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. IEEE Trans Image Process 2000;9:561–76.
[12] Havel J, Merciol F, Lefèvre S. Efficient tree construction for multi-scale image representation and processing. J Real-Time Image Process 2019;16(4):1129–46.
[13] Soille P. Constrained connectivity for hierarchical image decomposition and simplification. IEEE Trans Pattern Anal Mach Intell 2008;30(7):1132–45.
[14] Perret B, Cousty J, Tankyevych O, Talbot H, Passat N. Directed connected operators: Asymmetric hierarchies for image filtering and segmentation. IEEE Trans Pattern Anal Mach Intell 2015;37(6):1162–76.
[15] Tochon G, Dalla Mura M, Veganzones MA, Géraud T, Chanussot J. Braids of partitions for the hierarchical representation and segmentation of multimodal images. Pattern Recognit 2019;95:162–72.
[16] Passat N, Naegel B, Kurtz C. Component-graph construction. J Math Imaging Vision 2019;61(6):798–823.
[17] Morimitsu A, Passat N, Alves WAL, Hashimoto RF. Efficient component-hypertree construction based on hierarchy of partitions. Pattern Recognit Lett 2020;135:30–7.
[18] Bosilj P, Kijak E, Lefèvre S. Partition and inclusion hierarchies of images: A comprehensive survey. J Imaging 2018;4(2):33.
[19] Cousty J, Najman L, Kenmochi Y, Guimarães SJF. Hierarchical segmentations with graphs: Quasi-flat zones, minimum spanning trees, and saliency maps. J Math Imaging Vision 2018;60(4):479–502.
[20] Alonso-González A, López-Martínez C, Salembier P. Filtering and segmentation of polarimetric SAR data based on binary partition trees. IEEE Trans Geosci Remote Sens 2012;50(2):593–605.
[21] Valero S, Salembier P, Chanussot J. Hyperspectral image representation and processing with binary partition trees. IEEE Trans Image Process 2013;22(4):1430–43.
[22] Alonso-González A, Valero S, Chanussot J, López-Martínez C, Salembier P. Processing multidimensional SAR and hyperspectral images with binary partition tree. Proc IEEE 2013;101(3):723–47.
[23] Alonso-González A, López-Martínez C, Salembier P. PolSAR time series processing with binary partition trees. IEEE Trans Geosci Remote Sens 2014;52(6):3553–67.
[24] Veganzones MA, Tochon G, Dalla Mura M, Plaza AJ, Chanussot J. Hyperspectral image segmentation using a new spectral unmixing-based binary partition tree representation. IEEE Trans Image Process 2014;23(8):3574–89.
[25] Valero S, Salembier P, Chanussot J. Object recognition in hyperspectral images using binary partition tree representation. Pattern Recognit Lett 2015;56:45–51.
[26] Salembier P, Foucher S. Optimum graph cuts for pruning binary partition trees of polarimetric SAR images. IEEE Trans Geosci Remote Sens 2016;54(9):5493–502.
[27] Al-Dujaili A, Merciol F, Lefèvre S. GraphBPT: An efficient hierarchical data structure for image representation and probabilistic inference. In: International symposium on mathematical morphology, proceedings. Lecture Notes in Computer Science, Vol. 9082, Springer; 2015, p. 301–12.
[28] Perret B, Chierchia G, Cousty J, Guimarães S, Kenmochi Y, Najman L. Higra: Hierarchical graph analysis. SoftwareX 2019;10:100335.
[29] Naegel B, Passat N. Interactive segmentation based on component-trees. Imag Process Line 2014;4:89–97.
[30] Randrianasoa JF, Kurtz C, Desjardin E, Passat N. Binary partition tree construction from multiple features for image segmentation. Pattern Recognit 2018;84:237–50.
[31] Randrianasoa JF, Kurtz C, Desjardin É, Gançarski P, Passat N. Evaluating the quality of binary partition trees based on uncertain semantic ground-truth for image segmentation. In: International conference on image processing, proceedings. 2017. p. 3874–3878.
[32] Randrianasoa JF, Kurtz C, Desjardin É, Gançarski P, Passat N. Intrinsic quality analysis of binary partition trees. In: International conference on pattern recognition and artificial intelligence, proceedings. 2018. p. 114–119.
[33] Randrianasoa JF, Cettour-Janet P, Kurtz C, Desjardin E, Gançarski P, Bednarek N, et al. Supervised quality evaluation of binary partition trees for object segmentation. Pattern Recognit 2021;111:107667.
[34] Perret B, Cousty J, Rivera Ura JC, Guimarães SJF. Evaluation of morphological hierarchies for supervised segmentation. In: International symposium on mathematical morphology, proceedings. Lecture Notes in Computer Science, Vol. 9082, 2015, p. 39–50.
[35] Randrianasoa JF. Représentation d'images hiérarchique multi-critère. (hierarchical multi-feature image representation). (Ph.D. thesis), 2017, University of Reims Champagne-Ardenne, France.
[36] Pont-Tuset J, Marqués F. Upper-bound assessment of the spatial accuracy of hierarchical region-based image representations. In: International conference on acoustics, speech and signal processing, proceedings. 2012. p. 865–868.
[37] Blake A, Rother C, Brown MA, Pérez P, Torr PHS. Interactive image segmentation using an adaptive GMMRF model. In: European conference on computer vision, proceedings. Lecture Notes in Computer Science, Vol. 3021, 2004, p. 428–41.
[38] Alpert S, Galun M, Basri R, Brandt A. Image segmentation by probabilistic bottom-up aggregation and cue integration. In: Computer vision and pattern recognition. proceedings. 2007.
[39] Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A. The pascal visual object classes (VOC) challenge. Int J Comput Vis 2010;88:303–38.