



**HAL**  
open science

## AGAT: Building and evaluating binary partition trees for image segmentation

Jimmy Francky Randrianasoa, Camille Kurtz, Eric Desjardin, Nicolas Passat

► **To cite this version:**

Jimmy Francky Randrianasoa, Camille Kurtz, Eric Desjardin, Nicolas Passat. AGAT: Building and evaluating binary partition trees for image segmentation. *SoftwareX*, 2021, 16, pp.100855. 10.1016/j.softx.2021.100855 . hal-03384164

**HAL Id: hal-03384164**

**<https://hal.science/hal-03384164v1>**

Submitted on 14 Nov 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# AGAT: Building and Evaluating Binary Partition Trees for Image Segmentation

Jimmy Francky Randrianasoa<sup>a</sup>, Camille Kurtz<sup>b</sup>, Éric Desjardin<sup>c</sup>,  
Nicolas Passat<sup>c</sup>

<sup>a</sup>*EPITA Research and Development Laboratory (LRDE), France*

<sup>b</sup>*Université de Paris, LIPADE EA 2517, 75006 Paris, France*

<sup>c</sup>*Université de Reims Champagne Ardenne, CReSTIC EA 3804, 51100 Reims, France*

---

## 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.

*Keywords:* binary partition tree, hierarchical modeling, image / graph processing, quality evaluation

---

## Current code version

Nr.	Code metadata description	Please fill in this column
C1	Current code version	v2.0
C2	Permanent link to code/repository used for this code version	<a href="https://github.com/yonmi/AGAT2.0">https://github.com/yonmi/AGAT2.0</a>
C3	Code Ocean compute capsule	NA
C4	Legal Code License	Cecill-B
C5	Code versioning system used	git
C6	Software code languages	Java SE 8
C7	Compilation requirements, operating environments & dependencies	JDK 1.8 (for Eclipse developers, .project files are also provided)
C8	If available Link to developer documentation/manual	<a href="https://github.com/yonmi/AGAT2.0">https://github.com/yonmi/AGAT2.0</a>
C9	Support email for questions	<a href="mailto:agat@univ-reims.fr">agat@univ-reims.fr</a>

Table 1: AGAT code metadata.

### 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 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

26 trees are the component-tree [5] (which models a grey-level image as the  
27 Hasse diagram of the binary connected components of all the threshold sets,  
28 with respect to the inclusion relation) and its multivalued variant [6]; the tree  
29 of shapes [7] (which is a self-dual variant of the component-tree, that gath-  
30 ers information obtained by thresholding the image in both top-down and  
31 bottom-up ways) and its multivalued variant [8]; the watershed tree [9] (that  
32 derives from hierarchical watersheds [10], and allows to model in a hierarchi-  
33 cal way the saliency maps derived from the gradient of an image), the binary  
34 partition tree [11] (that we consider in this article), and some variants such  
35 as the  $\alpha$ -tree [12] (that derives from the concept of constrained connectivity  
36 [13]). Many other hierarchical models (including not only trees but also more  
37 complex directed acyclic graph structures, e.g. asymmetric hierarchies, braids  
38 of partitions, component-graphs or component-hypertrees [14, 15, 16, 17]  
39 that generalize / extend the tree structures beyond their usual topological  
40 and/or spectral hypotheses of definition) were provided. A whole discussion  
41 is beyond the scope of this article; the interested reader can refer to [18] for  
42 a recent survey.

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

51 Most of the hierarchical models (e.g. the component-tree or the tree of  
52 shapes) can be built from an image, without considering any additional in-  
53 formation. Such trees can be seen as pure image modeling data structures,  
54 that embed an image into an alternative space, where it can be handled and  
55 modified thanks to image processing paradigms. By contrast, the binary  
56 partition tree (BPT, for brief) [11], is built from two kinds of information:  
57 (1) the intrinsic information carried by the input image (or, more gener-  
58 ally, the input valued graph), and (2) an extrinsic—generally user-defined  
59 / application-based—information that determines which criteria should be  
60 considered for describing the input data in a multiscale way. This expert /  
61 domain-based information is crucial in certain application fields. In other  
62 words, the binary partition tree is not only an image-oriented but also a  
63 knowledge-based data structure. As a consequence, it can be relevantly in-  
64 volved in image analysis tasks that require the embedding of expert-defined  
65 priors and knowledge. For instance, the binary partition tree is quite popu-  
66 lar in remote sensing applications [20, 21, 22, 23, 24, 25, 26], where the way

67 to decompose an image depends on its content (e.g. an urban area vs. a  
68 wild forest zone) but also on the purpose of the analysis (e.g. classifying the  
69 buildings vs. observing pollution effects).

70 Various software programs and libraries are available for hierarchical im-  
71 age model handling. GraphBPT<sup>1</sup> [27] is especially designed for building and  
72 using binary partition trees. More general-purposed libraries, for instance  
73 scikit-image<sup>2</sup>, Higrā [28] or the trees-lib library<sup>3</sup> deal with wider issues. In  
74 particular the last two mentioned allow for the construction of binary parti-  
75 tion trees or  $\alpha$ -trees, that can be seen as a variant of binary partition trees.  
76 Other libraries, namely Olena<sup>4</sup> or LibTIM<sup>5</sup> (used e.g. in [29]) are mainly  
77 geared towards the construction of the so-called component-trees and / or  
78 trees of shapes, but could probably support further extensions for handling  
79 binary partition trees.

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

91 Another contribution of AGAT compared to the already available libraries  
92 is the proposal of various tools for evaluating the quality of a binary parti-  
93 tion tree (or equivalently, the quality of the meta-parametrization of its con-  
94 struction) with respect to object segmentation purposes. Indeed, evaluating  
95 the quality of a hierarchical image model is an important—but infrequently  
96 considered—topic as a prerequisite to its actual involvement for real appli-  
97 cations [31, 32, 33].

---

<sup>1</sup><https://github.com/ash-aldujaili/GraphBPT>

<sup>2</sup><https://scikit-image.org>

<sup>3</sup><https://github.com/pbosilj/trees-lib>

<sup>4</sup><https://www.lrde.epita.fr/wiki/Olena>

<sup>5</sup><https://github.com/bnaegel/libtim>

## 98 2. Software description

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

### 101 2.1. Software architecture

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

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

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

134 The `TreeEvaluation` module is dependent on the `BinaryPartitionTree`  
135 module. It provides various classes related to the quantitative evaluation of

136 the quality of a binary partition tree (or its meta-parametrization) with re-  
137 spect to object segmentation purposes. Both intrinsic and extrinsic analyses  
138 can be carried out. For a given binary partition tree (an object built from the  
139 `BinaryPartitionTree` module), the user can provide examples of ground-  
140 truth (defined as binary regions of interest in the image support) and quality  
141 metrics. The main data structures of this module make it possible to manage  
142 both information on the ground-truth segments provided by the user (e.g.  
143 coordinates of the bounding box, semantic labels, etc.) but also sub-trees of  
144 interest on which the analysis is carried out (to make processing faster by  
145 restricting it spatially and hierarchically).

## 146 *2.2. Software functionalities*

147 AGAT proposes a large amount of algorithms for the construction, the  
148 handling and the evaluation of binary partition trees (complementary tech-  
149 nical details regarding the main data structure architecture can be found in  
150 [34, Appendix B]):

### 151 • **BPT construction:**

- 152 – mono-image / mono-criterion [11]: tree construction from pix-  
153 els or flat zones or a given partition, various generic (e.g. color:  
154 RGB, LAB, geometric: elongation, smoothness) and thematic  
155 (e.g. NDVI, NDWI) criteria are available;
- 156 – multi-image / multi-criteria [30]: efficient process to establish dif-  
157 ferent kinds of consensus among the adjacency lists (e.g. majority  
158 vote, most frequent, etc.), visualization of the conflict between  
159 metrics.

160 The construction criteria and the consensus policy are provided by the  
161 user as parameters of the (multifeature) binary partition tree construc-  
162 tion process (an example can be found in the code snippet of Listing 1).

### 163 • **BPT handling:**

- 164 – definition of tree cuts from partition cardinality: flat or fitting  
165 with an input mask of an object of interest;
- 166 – backup and restore trees from .h5 files, which is a Hierarchical  
167 Data Format (HDF) designed to store and organize large amounts  
168 of data and ensures compatibility with external tools;
- 169 – export trees to .xml and .dot files, which allows visualization with  
170 external tools.

- 171 • **BPT quality evaluation:**
  - 172 – management of multiple ground-truth examples per image with
  - 173 potentially different semantic labels;
  - 174 – extraction of sub-trees of interest to speed up the analysis by re-
  - 175 stricting it spatially and hierarchically;
  - 176 – intrinsic analysis: combinatorial, quantitative, node assessment
  - 177 from the notion of pure and impure nodes matching with ground-
  - 178 truth examples [32, 33];
  - 179 – extrinsic analysis: home-made evaluation framework [31, 33], im-
  - 180 plementation of other existing frameworks from the literature
  - 181 [35, 36], evaluation metrics: F-measure / Dice, Jaccard index.

### 182 3. Illustrative examples

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

#### 191 3.1. Building a (traditional or multifeature) binary partition tree

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

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

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

```
204 import java.awt.image.BufferedImage;  
205 import java.util.Map.Entry;
```



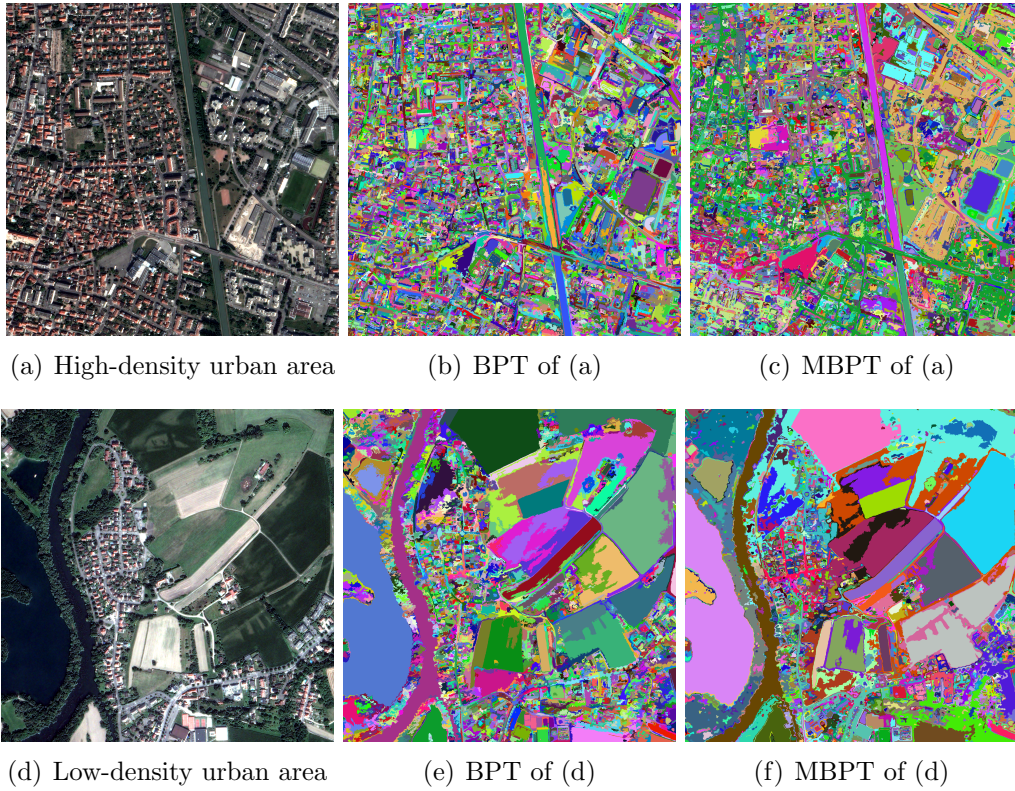


Figure 1: (a, d) Two VHSR satellite images ( $2000 \times 2000$  pixels) at a spatial resolution of 60 cm sensed by the PLÉIADES satellite and covering different areas. (b, e) Partitioning results from traditional binary partition trees computed from (a) and (d), respectively (23 500 and 5 000 regions, respectively), using one criterion:  $C_{colour}$ . (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:  $C_{colour}$ ,  $C_{elong}$ ,  $C_{ndvi}$ ,  $C_{ndwi}$ .

```

206 import datastructure;
207 import metric.bricks.Metric.TypeOfMetric;
208 import multi.sequential.MBPT;
209 import multi.strategy.consensus.bricks.Consensus.ConsensusStrategy;
210 import ui.ImFrame;
211 import utils;
212 /**
213  * Example to create a multi-feature binary partition tree using four criteria
214  */
215 public class ExampleCreateAndCutMbpt {
216     public static void main(String[] args) {
217
218         BufferedImage image = ImTool.read("../dataset/VHSR-sample1.png");
219
220         Tree tree = new MBPT(); //Create an empty tree
221         ((MBPT)tree).registerImage(image); //Register the image(s)
222

```

```

223     /* Choosing the consensus strategy to use */
224     int consensusRange = 5; /* % defining the interval of the list to consider */
225     int progressive = 1; /* interval defined proportionally to remaining number of
226        adjacency links */
227     ((MBPT) tree).setConsensusStrategy(ConsensusStrategy.SCORE_OF_RANK, consensusRange,
228        progressive);
229
230     /* Linking metrics to the image: four criteria are considered */
231     ((MBPT)tree).linkMetricToAnImage(image, TypeOfMetric.RADIOMETRIC_MIN_MAX);
232     ((MBPT)tree).linkMetricToAnImage(image, TypeOfMetric.NDVI);
233     ((MBPT)tree).linkMetricToAnImage(image, TypeOfMetric.NDWI);
234     ((MBPT)tree).linkMetricToAnImage(image, TypeOfMetric.SIMPLE_ELONGATION);
235
236     tree.grow(); //Build the BPT
237
238     /* Cutting */
239     if(tree.hasEnded()) {
240         int starting = 25, ending = 0, step = 5;
241         CutResult cutResult = CutBPT.execute(tree, starting, ending, step);
242
243         for(Entry<Integer, BufferedImage> entry: cutResult.regionImages.entrySet()) {
244             int numberOfRegions = entry.getKey();
245             BufferedImage partition = entry.getValue();
246             ImTool.show(partition, ImFrame.IMAGE_DEFAULT_SIZE, numberOfRegions);
247         }
248     }
249 }
250 }

```

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

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

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

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

- 263 •  $C_{colour}$ , defined as the increase of the range of the pixel intensity values  
264 for each radiometric band, induced by the putative fusion of incident  
265 regions;
- 266 •  $C_{ndvi}$ , that quantifies the difference of NDVI (Normalized Difference  
267 Vegetation Index, a standard indicator for the presence of green vege-  
268 tation) between two adjacent regions;

- 269 •  $C_{ndwi}$ , that quantifies the difference of NDWI (Normalized Difference  
270 Water Index, a standard indicator for the presence of water) between  
271 two adjacent regions;
- 272 •  $C_{elong}$ , defined as the change of geometrical elongation, potentially in-  
273 duced by the fusion of two regions.

274 Figure 1(b, e), illustrates the results of partitionings (induced by tree-cuts)  
275 obtained from traditional binary partition trees built by considering individ-  
276 ually the first criterion. Figure 1(c, f), illustrates the results of partitionings  
277 (induced by tree-cuts) obtained from multifeature binary partition trees built  
278 by considering collectively these four criteria.

279 A Java code snippet, presented in Listing 1, exemplifies how to obtain  
280 such results within AGAT. More extensive experiments related to the con-  
281 struction of various (multifeature) binary partition trees and the impact of  
282 these various kinds of trees in the context of remote sensing can be found in  
283 [32, 30].

### 284 3.2. *Assessing / comparing the quality of various binary partition trees*

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

- 292 • an intrinsic quality analysis [33], that evaluates the relevance of a binary  
293 partition tree based on the combinatorial analysis of its nodes and their  
294 relationships with a binary ground-truth associated to the input image;
- 295 • an extrinsic quality analysis [33], that evaluates the ability of a binary  
296 partition tree to provide a cut that minimizes at best a given quality  
297 metric provided as hyperparameter with respect to the ground-truth  
298 associated to the input image;
- 299 • a framework adapted from [35] that provides the F-measure of the seg-  
300 mented object vs. the associated ground-truth, with respect to the size  
301 of inside / outside markers generated from the ground-truth associated  
302 to the input image;
- 303 • a framework adapted from [36] that aims at computing the best cuts  
304 composed of  $k$  nodes for each  $k$  in  $[0, p]$  ( $p > 0$ ), allowing to reconstruct  
305 the targeted segmentation.

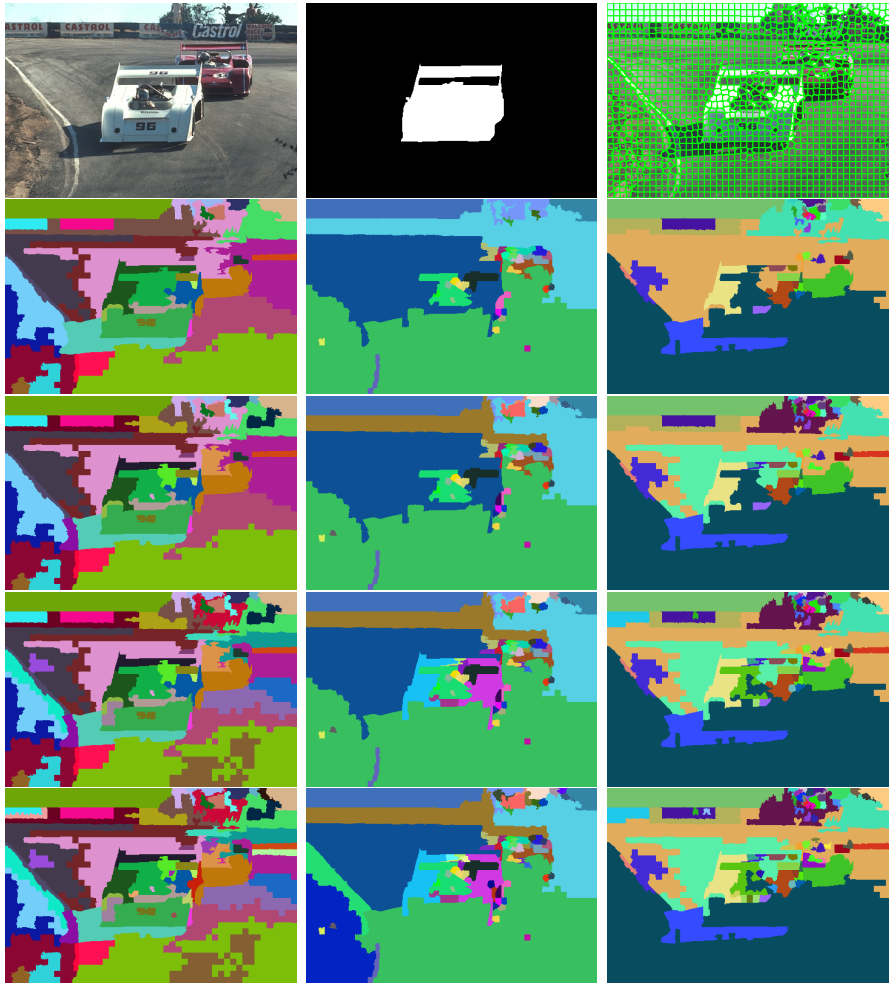


Figure 2: Horizontal cuts of different binary partition trees on an image from the Grabcut dataset. First line: initial image, ground-truth and initial superpixel partition. First column: binary partition tree built with WSDM criterion. Second column: binary partition tree built with Min-Max criterion. 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 colour, for the sake of visualization.

306 In [33], extensive experiments were carried out with three commonly used  
 307 datasets: Grabcut [37], Weizman [38] and VOC [39]. Examples of partition-  
 308 ing of an image from the Grabcut dataset are illustrated in Figure 2. These  
 309 partitions are obtained from three different binary partition trees, each one  
 310 built with a given criterion, noted WSDM, Min-Max and MSE, respectively  
 311 (see [33] for their formal definition).

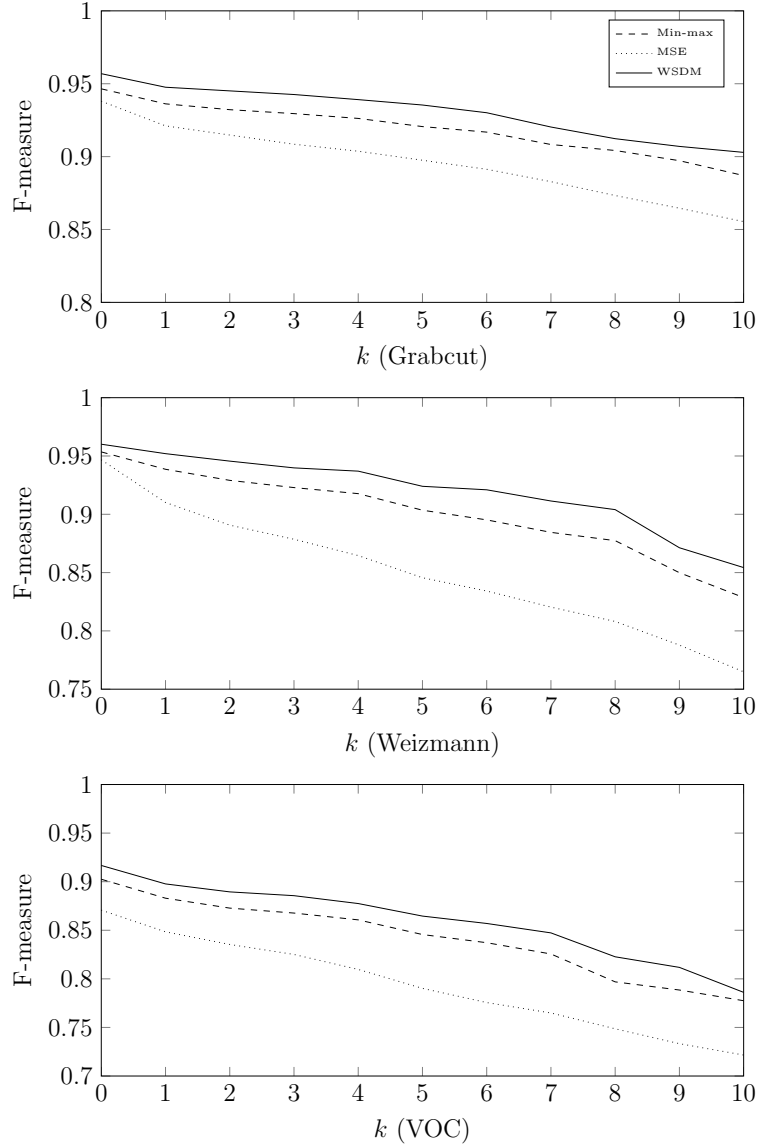


Figure 3: F-measure of the segmentations obtained from various kinds of binary partition trees in the evaluation framework [35], with respect to the size  $2k + 1$  of the structuring elements used for erosion (background and foreground) of the ground-truth. Top: Grabcut. Middle: Weizmann. Bottom: VOC. 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 [35].) 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.

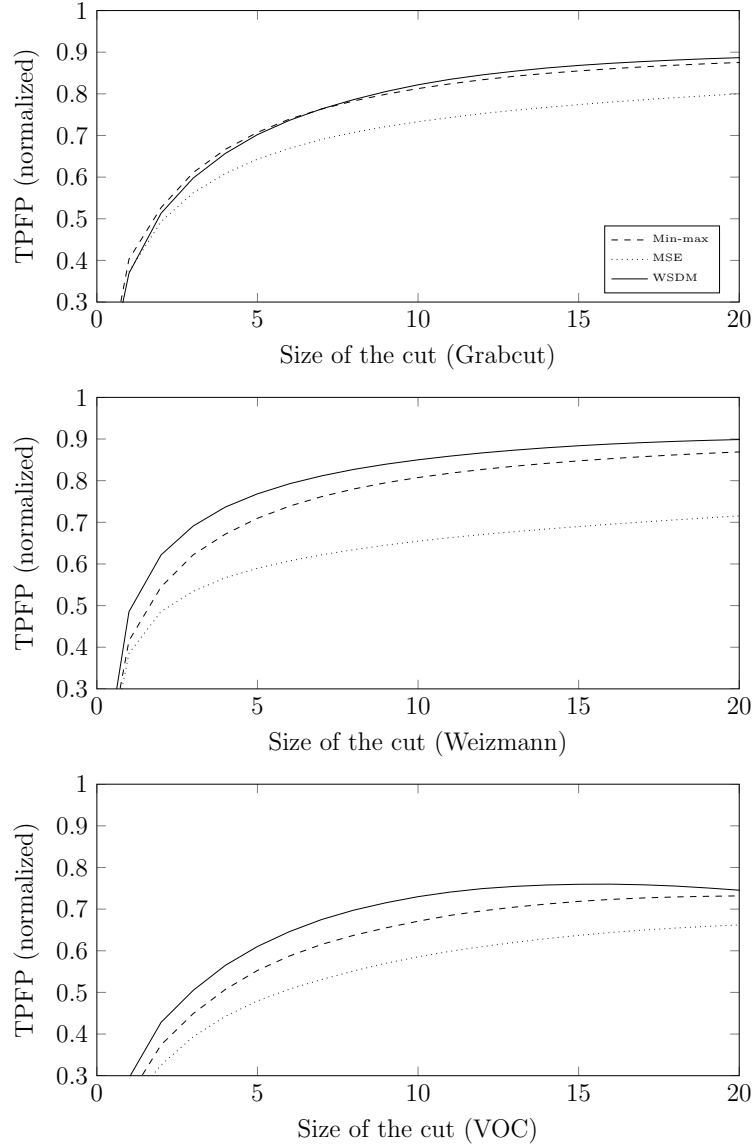


Figure 4: Normalized (mean) value of the TPFPP measure for the optimal cuts of a given size from various kinds of binary partition trees in the evaluation framework [36] Top: Grabcut. Middle: Weizmann. Bottom: VOC. Considering  $S$  as the segmentation result and  $G$  as the ground-truth, the (normalized) TPFPP score is defined as  $\frac{tp-fp}{|G|}$ , with  $tp = |S \cap G|$ ,  $fp = |S \setminus G|$ . The TPFPP 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.

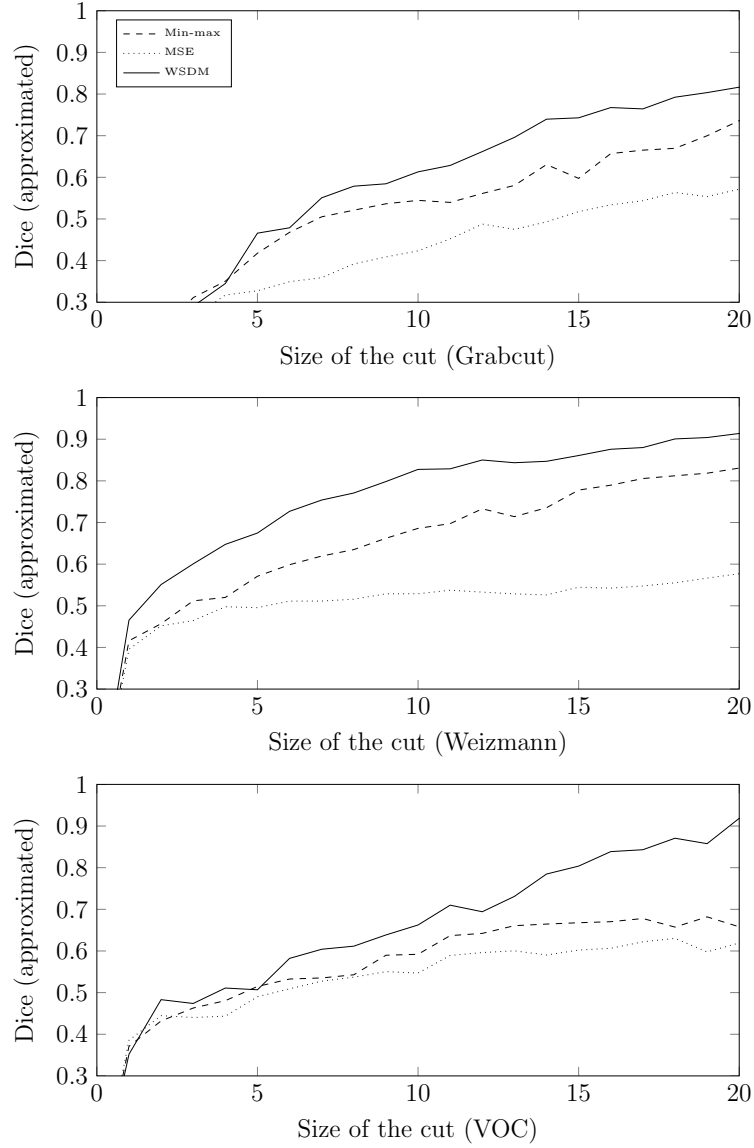


Figure 5: Normalized (mean) value of the Dice measure for the (near-)optimal cuts of a given size various kinds of binary partition trees. Top: Grabcut. Middle: Weizmann. Bottom: VOC. Considering  $S$  as the segmentation result and  $G$  as the ground-truth, the Dice score is defined as  $2tp/(2tp+fp+fn)$  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.

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

```

312 import java.awt.image.BufferedImage;
313 import java.io.PrintWriter;
314 import java.util;
315 import evaluation;
316 import standard.sequential.BPT;
317 import ui.ImFrame;
318 import utils.ImTool;
319 /**
320  * Example to evaluate the quality of a BPT with intrinsic analysis
321  */
322 public class ExampleBptIntrinsicAnalysis {
323     public static void main(String[] args) {
324
325         /* Create a tree */
326         BufferedImage image = ImTool.read("./dataset/test_img.png");
327         BufferedImage presegImg = ImTool.read("./dataset/test_img_slic.tif");
328         BPT tree = new BPT(image);
329         tree.setPreSegImage(presegImg);
330         tree.grow();
331
332         /* Extract the segments of reference from a ground truth image */
333         String gtImgPath = "./dataset/test_img-gt.png";
334         double alpha = 0.0;
335         TreeMap<Integer, SegReference> segReferences =
336             SegReference.extractSegmentsOfReference(gtImgPath, alpha, true);
337
338         /* Visualizing the segments if wanted */
339         SegReference.showBoundingBoxes(segReferences, image, ImFrame.IMAGE_STD_SIZE, "BB");
340
341         /* Extract sub trees */
342         STree extractedSubTrees[] = new STree[segReferences.size()];
343         int gti = 0;
344         for(Entry<Integer, SegReference> entry: segReferences.entrySet()){
345             SegReference gt = entry.getValue(); // each segment of reference
346             STree st = new STree(gti, tree, gt); st.index = gti++; // Create the subtree
347             extractedSubTrees[st.index] = st;
348         }
349
350         /* Evaluate the sub trees */
351         ArrayList<Eval> evalRes = new ArrayList<Eval>();
352         for(int i = 0; i < extractedSubTrees.length; ++i) {
353             STree st = extractedSubTrees[i];
354             evalRes.add(st.eval(EvalType.INTRINSIC));
355         }
356
357         /* Print the intrinsic evaluations results */
358         System.out.println("Intrinsic evaluation results: ");
359         for(Eval res: evalRes) {
360             res.printResults();
361         }
362     }
363 }

```

364 Figures 3–5 exemplify some metrics that can be computed from the var-  
365 ious implemented evaluation frameworks, thus allowing to compare the rele-  
366 vance of distinct binary partition trees with respect to the considered data /  
367 ground-truth, here in the case of object segmentation. The results depicted in  
368 these figures quantitatively emphasize the superiority of one the three tested



369 binary partition trees (namely the one built with WSDM criterion) over the  
370 other two ones. This is characterized by the fact that, in the three different  
371 experiments, WSDM leads to better values for the considered measures (the  
372 higher these values, the better the results). These results are qualitatively  
373 confirmed by the segmentations illustrated in Figure 2.

374 A Java code snippet, presented in Listing 2, illustrates how to obtain  
375 such results within AGAT. The interested readers can find more extensive  
376 experiments related to such natural images in [33]. In particular, more exten-  
377 sive experiments related to the binary partition tree evaluation framework  
378 proposed in AGAT, applied in the context of natural images (Grabcut, VOC  
379 and Weizmann datasets) are provided in this reference.

#### 380 4. Impact

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

393 Additionally, AGAT also embeds various tools for assessing the quality of  
394 binary partition trees. This quantitative evaluation framework can be very  
395 useful, for instance in the field of computer vision, where the partitioning of  
396 large / complex images is generally a prerequisite for high-level scene anal-  
397 ysis tasks (e.g. object detection and recognition). In this regard, providing  
398 tools that can be involved in selection / learning of appropriate context-  
399 dependent meta-parameters for image decomposition is of great interest for  
400 the community. In particular, such frameworks proposed in combination  
401 with the binary partition tree construction algorithms could lead to consider  
402 the induced image partitioning results as a relevant alternative to the usual,  
403 non-hierarchical, image decompositions proposed by super-pixel paradigms.

404 Finally, it is worth mentioning that the binary partition trees, beyond  
405 their usefulness for image analysis task, are first of all a way of building  
406 partitions of graph-based structures, independently of their associated se-  
407 mantics. Based on this fact, AGAT may then be used for solving any graph

408 partitioning problem provided that such partitionings are guided by one /  
409 many prior knowledge. This opens the way of the use of AGAT in a wide  
410 range of machine learning domains where the data are organized as graphs,  
411 i.e. with respect to usual binary relations.

## 412 **5. Conclusions**

413 We presented AGAT, a library dedicated to the construction, the han-  
414 dling and the evaluation of binary partition trees, which are tree data struc-  
415 tures providing hierarchical partitioning of images and, more generally, val-  
416 ued graphs. AGAT contains also many standard and state-of-the-art algo-  
417 rithms in this domain. AGAT is the first library allowing for the construction  
418 of multifeature binary partition trees, and it also gathers a large set of eval-  
419 uation tools for traditional binary partition trees. From this point of view,  
420 it constitutes, to our knowledge, the most complete and up-to-date library  
421 dedicated to binary partition trees. It is composed of three self-contained  
422 Java modules. Code sources are available via GitHub and natively compati-  
423 ble for Linux, Mac, and Windows systems. They can be downloaded with a  
424 simple command: `git clone https://github.com/yonmi/AGAT2.0.git`.

## 425 **Conflict of Interest**

426 We wish to confirm that there are no known conflicts of interest associated  
427 with this publication and there has been no significant financial support for  
428 this work that could have influenced its outcome.

## 429 **Acknowledgements**

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

## 433 **References**

- 434 [1] T. Lindeberg, Scale-space for discrete signals, *IEEE Transactions on*  
435 *Pattern Analysis and Machine Intelligence* 12 (3) (1990) 234–254.
- 436 [2] I. Rey-Otero, M. Delbracio, Anatomy of the SIFT method, *Image Pro-*  
437 *cessing On Line* 4 (2014) 370–396.

- 438 [3] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Ssstrunk, SLIC  
439 superpixels compared to state-of-the-art superpixel methods, *IEEE*  
440 *Transactions on Pattern Analysis and Machine Intelligence* 34 (2012)  
441 2274–2282.
- 442 [4] P. Cettour-Janet, C. Cazorla, V. Machairas, Q. Delannoy, N. Bednarek,  
443 F. Rousseau, E. Decencire, N. Passat, *Watervoxels*, *Image Processing*  
444 *On Line* 9 (2019) 317–328.
- 445 [5] P. Salembier, A. Oliveras, L. Garrido, Anti-extensive connected oper-  
446 ators for image and sequence processing, *IEEE Transactions on Image*  
447 *Processing* 7 (1998) 555–570.
- 448 [6] C. Kurtz, B. Naegel, N. Passat, Connected filtering based on multivalued  
449 component-trees, *IEEE Transactions on Image Processing* 23 (12) (2014)  
450 5152–5164.
- 451 [7] P. Monasse, F. Guichard, Scale-space from a level lines tree, *Journal of*  
452 *Visual Communication and Image Representation* 11 (2000) 224–236.
- 453 [8] E. Carlinet, T. Graud, MToS: A tree of shapes for multivariate images,  
454 *IEEE Transactions on Image Processing* 24 (12) (2015) 5330–5342.
- 455 [9] D. Santana Maia, J. Cousty, L. Najman, B. Perret, Characterization of  
456 graph-based hierarchical watersheds: Theory and algorithms, *Journal of*  
457 *Mathematical Imaging and Vision* 62 (5) (2020) 627–658.
- 458 [10] L. Najman, M. Schmitt, Geodesic saliency of watershed contours and  
459 hierarchical segmentation, *IEEE Transactions on Pattern Analysis and*  
460 *Machine Intelligence* 18 (1996) 1163–1173.
- 461 [11] P. Salembier, L. Garrido, Binary partition tree as an efficient repre-  
462 sentation for image processing, segmentation, and information retrieval,  
463 *IEEE Transactions on Image Processing* 9 (2000) 561–576.
- 464 [12] J. Havel, F. Merciol, S. Lefvre, Efficient tree construction for multi-  
465 scale image representation and processing, *Journal of Real-Time Image*  
466 *Processing* 16 (4) (2019) 1129–1146.
- 467 [13] P. Soille, Constrained connectivity for hierarchical image decomposition  
468 and simplification, *IEEE Transactions on Pattern Analysis and Machine*  
469 *Intelligence* 30 (7) (2008) 1132–1145.

- 470 [14] B. Perret, J. Cousty, O. Tankyevych, H. Talbot, N. Passat, Directed  
471 connected operators: Asymmetric hierarchies for image filtering and  
472 segmentation, *IEEE Transactions on Pattern Analysis and Machine In-*  
473 *telligence* 37 (6) (2015) 1162–1176.
- 474 [15] G. Tochon, M. Dalla Mura, M. A. Veganzones, T. Géraud, J. Chanussot,  
475 Braids of partitions for the hierarchical representation and segmentation  
476 of multimodal images, *Pattern Recognition* 95 (2019) 162–172.
- 477 [16] N. Passat, B. Naegel, C. Kurtz, Component-graph construction, *Journal*  
478 *of Mathematical Imaging and Vision* 61 (6) (2019) 798–823.
- 479 [17] A. Morimitsu, N. Passat, W. A. L. Alves, R. F. Hashimoto, Efficient  
480 component-hypertree construction based on hierarchy of partitions, *Pat-*  
481 *tern Recognition Letters* 135 (2020) 30–37.
- 482 [18] P. Bosilj, E. Kijak, S. Lefèvre, Partition and inclusion hierarchies of  
483 images: A comprehensive survey, *Journal of Imaging* 4 (2) (2018) 33.
- 484 [19] J. Cousty, L. Najman, Y. Kenmochi, S. J. F. Guimarães, Hierarchical  
485 segmentations with graphs: Quasi-flat zones, minimum spanning trees,  
486 and saliency maps, *Journal of Mathematical Imaging and Vision* 60 (4)  
487 (2018) 479–502.
- 488 [20] A. Alonso-González, C. López-Martínez, P. Salembier, Filtering and seg-  
489 mentation of polarimetric SAR data based on binary partition trees,  
490 *IEEE Transactions on Geoscience and Remote Sensing* 50 (2) (2012)  
491 593–605.
- 492 [21] S. Valero, P. Salembier, J. Chanussot, Hyperspectral image representa-  
493 tion and processing with binary partition trees, *IEEE Transactions on*  
494 *Image Processing* 22 (4) (2013) 1430–1443.
- 495 [22] A. Alonso-González, S. Valero, J. Chanussot, C. López-Martínez,  
496 P. Salembier, Processing multidimensional SAR and hyperspectral im-  
497 ages with binary partition tree, *Proceedings of the IEEE* 101 (3) (2013)  
498 723–747.
- 499 [23] A. Alonso-González, C. López-Martínez, P. Salembier, PolSAR time se-  
500 ries processing with binary partition trees, *IEEE Transactions on Geo-*  
501 *science and Remote Sensing* 52 (6) (2014) 3553–3567.

- 502 [24] M. A. Veganzones, G. Tochon, M. Dalla Mura, A. J. Plaza, J. Chanus-  
503 sot, Hyperspectral image segmentation using a new spectral unmixing-  
504 based binary partition tree representation, *IEEE Transactions on Image*  
505 *Processing* 23 (8) (2014) 3574–3589.
- 506 [25] S. Valero, P. Salembier, J. Chanussot, Object recognition in hyperspec-  
507 tral images using binary partition tree representation, *Pattern Recogni-*  
508 *tion Letters* 56 (2015) 45–51.
- 509 [26] P. Salembier, S. Foucher, Optimum graph cuts for pruning binary parti-  
510 tion trees of polarimetric SAR images, *IEEE Transactions on Geoscience*  
511 *and Remote Sensing* 54 (9) (2016) 5493–5502.
- 512 [27] A. Al-Dujaili, F. Merciol, S. Lefèvre, GraphBPT: An efficient hierarchi-  
513 cal data structure for image representation and probabilistic inference,  
514 in: *ISMM, International Symposium on Mathematical Morphology, Pro-*  
515 *ceedings*, Vol. 9082 of *Lecture Notes in Computer Science*, Springer,  
516 2015, pp. 301–312.
- 517 [28] B. Perret, G. Chierchia, J. Cousty, S. F. Guimarães, Y. Kenmochi,  
518 L. Najman, Higr: Hierarchical graph analysis, *SoftwareX* 10 (2019)  
519 100335.
- 520 [29] B. Naegel, N. Passat, Interactive segmentation based on component-  
521 trees, *Image Processing On Line* 4 (2014) 89–97.
- 522 [30] J. F. Randrianasoa, C. Kurtz, E. Desjardin, N. Passat, Binary Parti-  
523 tion Tree construction from multiple features for image segmentation,  
524 *Pattern Recognition* 84 (2018) 237–250.
- 525 [31] J. F. Randrianasoa, C. Kurtz, É. Desjardin, P. Gañarski, N. Passat,  
526 Evaluating the quality of binary partition trees based on uncertain se-  
527 mantic ground-truth for image segmentation, in: *ICIP, International*  
528 *Conference on Image Processing, Proceedings*, 2017, pp. 3874–3878.
- 529 [32] J. F. Randrianasoa, C. Kurtz, É. Desjardin, P. Gañarski, N. Passat,  
530 Intrinsic quality analysis of binary partition trees, in: *ICPRAI, Inter-*  
531 *national Conference on Pattern Recognition and Artificial Intelligence,*  
532 *Proceedings*, 2018, pp. 114–119.
- 533 [33] J. F. Randrianasoa, P. Cettour-Janet, C. Kurtz, E. Desjardin,  
534 P. Gañarski, N. Bednarek, F. Rousseau, N. Passat, Supervised qual-  
535 ity evaluation of binary partition trees for object segmentation, *Pattern*  
536 *Recognition* 111 (2021) 107667.

- 537 [34] J. F. Randrianasoa, Représentation d’images hiérarchique multi-critère.  
538 (hierarchical multi-feature image representation), Ph.D. thesis, Univer-  
539 sity of Reims Champagne-Ardenne, France (2017).
- 540 [35] B. Perret, J. Cousty, J. C. Rivera Ura, S. J. F. Guimarães, Evaluation  
541 of morphological hierarchies for supervised segmentation, in: ISMM,  
542 International Symposium on Mathematical Morphology, Proceedings,  
543 Vol. 9082 of Lecture Notes in Computer Science, 2015, pp. 39–50.
- 544 [36] J. Pont-Tuset, F. Marqués, Upper-bound assessment of the spatial ac-  
545 curacy of hierarchical region-based image representations, in: ICASSP,  
546 International Conference on Acoustics, Speech and Signal Processing,  
547 Proceedings, 2012, pp. 865–868.
- 548 [37] A. Blake, C. Rother, M. A. Brown, P. Pérez, P. H. S. Torr, Interac-  
549 tive image segmentation using an adaptive GMMRF model, in: ECCV,  
550 European Conference on Computer Vision, Proceedings, Vol. 3021 of  
551 Lecture Notes in Computer Science, 2004, pp. 428–441.
- 552 [38] S. Alpert, M. Galun, R. Basri, A. Brandt, Image segmentation by prob-  
553 abilistic bottom-up aggregation and cue integration, in: CVPR, Com-  
554 puter Vision and Pattern Recognition. Proceedings, 2007.
- 555 [39] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisser-  
556 man, The Pascal Visual Object Classes (VOC) challenge, International  
557 Journal of Computer Vision 88 (2010) 303–338.