BROCELIANDE: A COMPARATIVE STUDY OF ATTRIBUTE PROFILES AND FEATURE PROFILES FROM DIFFERENT ATTRIBUTES

François Merciol[1], Minh-Tan Pham[1], Deise Santana Maia[1], Antoine Masse[2], Christophe Sannier[2]

[1] Université Bretagne Sud – IRISA UMR 6074, Vannes, France; firstname.lastname@irisa.fr
[2] SIRS, Villeneuve-d'Ascq, France; firstname.lastname@sirs-fr.com

KEY WORDS: Remote sensing imagery, tree representation, attribute profiles, feature profiles, multilevel image description

ABSTRACT:

Morphological attribute profiles (APs) are among the most effective spatial-spectral methods to perform multilevel image description based on hierarchical tree-based representation. They have been widely applied to the processing and characterization of remote sensing images, in particular to tackle classification task, in the literature. Recently, a novel extension of APs called FPs has been proposed by replacing pixel gray-levels with some statistical and geometrical features when forming the output profiles. FPs have been proved to be more efficient than the standard APs when generated from both inclusion and partition trees. The motivation of this article is to conduct a comparative study of APs and FPs using different attributes including some novel ones that have not been used in the literature. We also present our developed library called Broceliande, which proposes efficient implementation of APs and FPs to perform remote sensing image classification, with various choices of tree structures as well as attributes. We perform our experiments on two high resolution optical image data sets and provide comparative results of APs and FPs, showing and confirming their effectiveness to describe and classify remote sensing images.

1. INTRODUCTION

Earth observation has become essential for activities connected with the environment, whether to understand it or to preserve it. One of the most efficient manners remains the use of remote sensing imagery. However, due to the increasing amount of data, we have to find both fastest processing and relevant results. Among a great number of spatial-spectral approaches in the literature, morphological attribute profiles (APs) (Dalla Mura et al., 2010, Pham et al., 2018b) have been widely used to process and describe remote sensing images in the literature. The reason relies on their powerful multilevel modeling of the image content and their efficient implementation via tree representation. In fact, the core idea of APs is to explore hierarchical information from a sequence of filtered versions of an image. These images are obtained by the application of different filter rules (called attributes) characterizing the size and shape of objects present in the image. Recently, a novel extension of APs called feature profiles (FPs) (Pham et al., 2017a) has been proposed by replacing pixel gray-levels with the attributes themselves when forming the output profiles. FPs have been proved to be more efficient than the standard APs and the authors showed that APs are indeed a part of the general FPs, in case that the gray-level is used as attribute to form the output profiles. A recent paper has investigated the performance of APs and FPs generated from different kinds of tree structures (min-tree, max-tree, tree of shapes and partition trees such as α-tree or ω-tree) (Pham et al., 2018a). Another one has studied the filtering rules (Das et al., 2020) during AP construction. Nevertheless, tree formation and pruning are not the only factors that affect the performance of APs and FPs. The choice of attributes is even more important in most cases since they decide how the tree should be pruned as well as which output profiles would be formed.

In this article, our motivation is to conduct a comparative study of APs and FPs using different attributes to tackle the classification of optical remote sensing images. Our analysis could help future AP and FP users to have good selection of attributes in order to to perform their multilevel characterization and processing of remote sensing images. In the remainder of this paper, Section 2 briefly summarizes some backgrounds of hierarchical representation using APs and FPs; then presents our developed libraries including TRISKELE and Broceliande as well as the efficient implementation of some promising attributes proposed by our libraries. In section 3, we present our experimental study to perform and compare the performance of APs and FPs using different types of attributes on two remote sensing data sets including one panchromatic image acquired by the IKONOS satellite at 1-m resolution, and one large-scale multispectral image acquired by the Pleiades sensor at 2-m resolution. Section 4 finally concludes the paper and discusses some future works.

2. METHOD

2.1 Hierarchical representation with APs and FPs

Component trees are structured multilevel representations of the components/regions of an image. They are usually divided into two categories that are both explored in this research: inclusion trees and partition trees. Among the inclusion trees, we use the well-established max-tree and min-tree, and the newly proposed median-tree, which aims to approximate a tree-of-shapes through a linear time complexity algorithm. Hierarchical representation is particularly suited for handling remote sensing images, as attested by the vast research on this field. For instance, component trees are the basis for computing the morphological attribute profiles (APs), which have been successfully applied to the classification of high resolution remote sensing images (Dalla Mura et al., 2010, Pham et al., 2018b). For a reminder, APs are multilevel image description tools obtained by successively applying a set of morphological attribute filters (AFs). Given a grayscale image $X : E \rightarrow \mathbb{Z}, E \subseteq \mathbb{Z}^2$, the standard generation of APs on X is achieved by applying a sequence of AFs $\{\phi_k\}_{k=1}^{K}$ based on a tree model (which could be a min-tree, max-tree, tree of shapes, median-tree or other partition trees). The AP descriptor of each pixel p in the definition domain of X is written as:

$$AP(p) = \Big\{X(p), \big[\phi_1(X)\big](p), \big[\phi_2(X)\big](p)\ldots, \big[\phi_K(X)\big](p)\Big\} \tag{1}$$

where $\phi_k(X)$ is the filtered image obtained by applying the attribute filtering ϕ with regard to the threshold k.

Pruning a tree to take the node value rather than the pixels it refers to provides an AP. When we perform the same reconstruction, not with level depending on the structure of the tree, but with another attribute value we form a feature profile (FP) (Pham et al., 2017a). Specifically, for each pixel p, AP of p obtained by an arbitrary AF ϕ_k is the gray value $X'(p)$, where $X' = \phi_k(X)$ is the image reconstructed from the filtered tree (cf. Eq (1)). Now, let $\Gamma_p(X)$ be the connected component (CC) of X containing p and let f be a feature or an attribute, i.e. a function admitting a CC and outputting a real value, to be extracted. The FP of p will be $f[\Gamma_p(X')]$. More formally, the generation of FPs are defined as follows:

$$FP_f(p) = \Big\{X(p), f\big[\Gamma_p(\phi_1(X))\big], \ldots, f\big[\Gamma_p(\phi_K(X))\big]\Big\} \tag{2}$$

Unlike in AP technique where only one profile is produced from a pruned tree, several features can be simultaneously extracted and stacked to form the final FP. For more information about the extraction of APs and FPs, readers are invited to read their related papers (Dalla Mura et al., 2010, Pham et al., 2017a).

2.2 Developed tools

We have developed tools to enable the hierarchical and multi-level image representation and processing: TRISKELE[1] (Merciol, al., 2017) for tree building and pruning, Broceliande[2] (Merciol, al., 2018) using TRISKELE and Shark Random Forest (Igel et al., 2008) for image classification. The aforementioned component trees are efficiently implemented in TRISKELE and can be used to build real-time user interfaces. All attributes managed by Broceliande can be used both for tree pruning and for image reconstruction from a pruned tree.

Since the first utilization of APs, the standard attributes exploited in the filtering step have been *area*, *moment of inertia* and *standard deviation* (Dalla Mura et al., 2010, Pham et al., 2017b, Pham et al., 2018b). Those attributes are suitable for the processing of large-scale high resolution (HR) images since they can be computed in one pass over the nodes of a component tree. Keeping the software up-to-date requires adding the attributes according to user requests. Therefore, some attributes including *perimeter*, *compactness*, and *rectangularity* have been recently added to our library. All of these attributes are computed in linear time on a component tree.

2.3 Efficient implementation

The added attributes are introduced with the TRISKELE constraints. To maintain an efficient implementation, all algorithms integrated into the software have linear or quasi-linear time complexity. This subsection explains how we can achieve this goal. It starts with a brief presentation of the hidden data structure with compact and linear information that help us to design attribute algorithms. First, with the moment of inertia attribute, we illustrate the bottom-up approach for each attribute process in order to limit access by only one read and at most two writes per element. Then, with the perimeter attribute, we explain how to take advantage of the hierarchical structure to reduce time by taking into account neighborhood pixels. Lastly, with the standard deviation attribute, we present another way to obtain faster results close to the original definition.

2.3.1 Tree and attributes implementation First of all, we use the fundamental properties of hierarchical data representation. Consider a gray scale image (left side of Figure 1). This image is a matrix of pixels. We can build different types of trees (min-tree, max-tree, . . .) to manage the image. We choose to build a max-tree (right side of figure 1).

This tree is a hierarchical representation. All pixels have parent. These parents are called nodes. All nodes (except the root) also have a parent node. This defines an inclusion tree. Because we choose a max-tree, the nodes are organized from light gray (low level of the tree) to dark gray (high level). All nodes can have properties. We represent them with colored bullets in Figure 1.

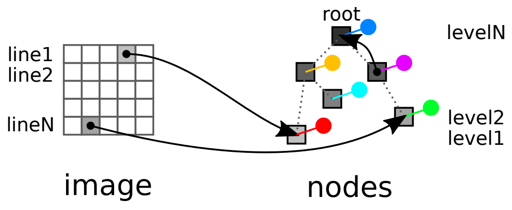


Figure 1. Hierarchical representation of image using a max-tree

TRISKELE uses a compact representation of this data. The tree is defined only with an array of parents (see Figure 2). All pixels and nodes are associated with a unique index. The indexes under "pixel cardinality" refer to the pixels from the first row to the last row of the image. Indexes with higher "cardinality of pixels", refer to tree nodes. Thanks to this building process, all nodes are sorted according to their level. So the light-gray nodes are on the right and the darker-gray ones are on the left.

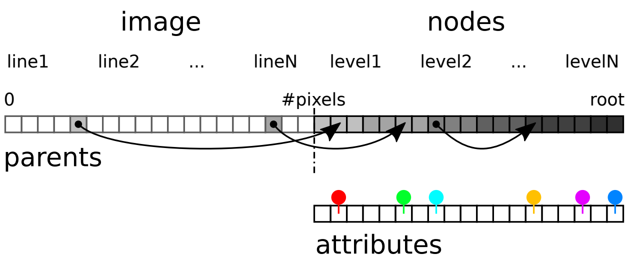


Figure 2. Tree implemented with array

Attributes are only defined for nodes. The attributes can be simple scalar (surface, perimeter, gray value) or multivalued (centroid, bounding box), and they are also stored in another array. The position of an attribute in an array gives the position of the associated node. Here are our efficient calculations of new attributes.

2.3.2 MoI computation The moment of inertia is a morphological attribute. It depends on the centroid distance among the nodes and the child nodes.

[1] `https://gitlab.inria.fr/obelix/triskele/`
[2] `https://gitlab.inria.fr/obelix/broceliande/`

We first compute barycenter with

$$G_j = \frac{\sum_i n_i X_{j,i}}{\sum n_i}, i \in Children, j \in \{1,2\} \qquad (3)$$

Then we computed the moment of inertia

$$MoI_t = \frac{\sqrt{\sum_j \sum_i n_i (G_{j,i} - Gj,t)^2}}{\sum n_i}, i \in Children, j \in \{1,2\} \qquad (4)$$

The issue is that a strict application of this formula involves the processing of nodes from the leaves to the root and, then, it goes back each time to retrieve children values. The computation time impact is significant. The key is therefore to consider each item only once and store its contribution on the fly. The consequence is only one read and only one write per item (pixels and nodes). The complexity is to be $O(n)$.

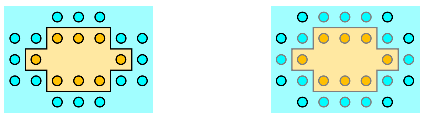
Because the moment of inertia is used with a random forest, only trends are needed. So we can save time by forgetting the square root step. In that case, the Algorithm 1 actually gives MoI^2.

Algorithm 1 Compute MoI^2

```
for all $c \in nodes$ do
    $MoI_c \leftarrow 0$
end for
for all $c \in pixels$ do
    $p \leftarrow parent(c)$
    $d \leftarrow \sum_j (G_{j,c} - Gj,p)^2$
    $MoI_p \leftarrow MoI_p + d$
end for
for all $c \in nodes$ do
    $MoI_c \leftarrow \frac{MoI_c}{Area_c^2}$
    $p \leftarrow parent(c)$
    $d \leftarrow MoI_p + \sum_j (G_{j,c} - Gj,p)^2 Area_c$
end for
```

We assume that the average position of each node (centroid) is mainly stored in G. All attribute algorithms in TRISKELE are designed with the same approach (bottom up).

2.3.3 Perimeter computation The perimeter attribute is a good example to see how to take into account the array data structure to implement the tree structure. The perimeter is the set of pixels that define the border of a node. It depends on the connectivity. Figure 3 illustrates that the higher connectivity (C8) increases the number of pixels involved in perimeter. Note that, even if two areas share the same border, the perimeter depends on their convex (or concave) sides. The perimeter of an embedded node is always shorter than the enclosing node.

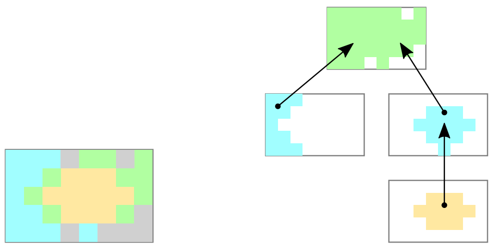


(a) C4 connectivity (b) C8 connectivity

Figure 3. Connectivity impact on perimeter

A simple approach to finding the perimeter could be to create a parent map. In this case, we replace the gray level of each pixel with the index value of its parent. We must consider in this map all the pairs of neighbors (depending on the connectivity) to be sure not to miss any pixel border. On the other hand, we have to be sure not to consider a pixel twice to play this role. This simple approach requires an additional map. Its size is the same as the number of pixels in the original image. With huge images (Giga pixels), an index of parent needs at least 32 bits. The complexity of such algorithm is also inappropriate for handling huge images.

Our approach starts with morphological observations. Consider a simple image (a) with only 3 nodes in Figure 4. We assume it is a gray-scale image, but colored here to highlight the nodes. We build an inclusion tree (e.g. max-tree). The orange pixels have the max value, the green ones have the min values and the blue ones in the middle. We also consider some no-data pixels.



(a) Image (b) Tree

Figure 4. Perimeter data structure

The hierarchical representation is depicted in (b) in the same figure. Figure 5 represents the pixel borders with C4 connectivity. Pay attention on the fact when a pixel becomes a perimeter pixel in a node, and when it loses this property in a parent node. We now develop these different cases, represented by 6 numbered pixels from the figure:

① is orange border with blue node but no longer in parent nodes.

② is orange border with blue node and blue border with green node, but no longer in parent nodes.

③ is border everywhere with no-data.

④ is blue border with green node but no longer in parent nodes.

⑤ is border with frame image or no-data everywhere since it appear in blue node.

⑥ is border with frame image or no-data everywhere since it appears in green node.

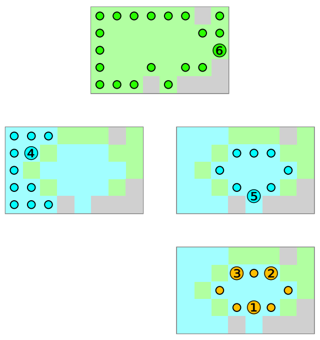


Figure 5. Pixel borders with C4 connectivity

Each pixel joins the tree at a base node. At this moment, it could be surrounded by the same value in flat area and never be considered as a border. Or it may be close to another value. If

Algorithm 2 increasePerimeter ($from$, to)

```
for $c \leftarrow from$ to $to$ do
    $perimeter(c) \leftarrow perimeter(c) + 1$
    $c \leftarrow parent(c)$
end for
```

the value belongs to an undernode (near the leaves), it should simply wait to be joined by this neighbor and never be a border. If the value belongs to a uppernode (close to the root), it will play the role of border until it joins this node. So, the general approach is for each pixel, we consider the higher ancestor of all its neighbors. The algorithm 3 is quasi-linear. Time per pixel depends on efficient way to find ancestor.

Algorithm 3 Compute perimeter

```
for all $c \in nodes$ do
    $perimeter(c) \leftarrow 0$
end for
for all $pi \in data - pixels$ do
    if $isFrameImage(pi)$ then
        $increasePerimeter(pp, root)$
    else
        $pp \leftarrow parent(pi)$
        $max \leftarrow pp$
        $npSet \leftarrow \emptyset$
        for all $n \in neighbors - C4(pi)$ do
            if $isNoData(n)$ then
                $max \leftarrow root$
            else
                $npSet \leftarrow npSet \cup parent(n)$
            end if
            if $max = root$ then
                $increasePerimeter(pp, root)$
            else
                $sort(npSet)$
                for all $np \in npSet$ do
                    $max \leftarrow ancestor(max, np)$
                end for
                $increasePerimeter(pp, max)$
            end if
        end for
    end if
end for
```

The properties of the array 'parents', described in the previous section, help us to implement an efficient method to find ancestors. We just have to compare the index as in Algorithm 4.

Algorithm 4 ancestor (a, b)

```
if $a = root$ or $b = root$ then
    return root
else
    loop
        if $a = b$ then
            return root
        end if
        while $a \leq b$ do
            $a \leftarrow parent(a)$
        end while
        while $b \leq a$ do
            $b \leftarrow parent(b)$
        end while
    end loop
end if
```

The Table 1 gives execution time obtained with the remote sensing image described in the next section. Except for the perimeter attribute, the 13 megapixels of the image are processed in around 10 seconds (depending of attributes). So the throughput is around 1 megapixel/ second. This table and the Figure 6 are produced with the command in our library:

```
channelGenerator src.tif dst.tif -b 3 -f
    ↪ <FeatureProfileName> -t Med -a
    ↪ Area --thresholds 10,100,5000 --
    ↪ auto --time
```

2.3.4 Alternative SD To keep this efficiency we also propose alternative production of the standard deviation attribute. For each node, the standard deviation depends on its mean gray value ($\overline{X}$).

$$sd^2 = \frac{1}{N}\sum(X_i - \overline{X})^2 \qquad (5)$$

The computation of the mean gray values takes time (see Table 1). If we consider the weight of a node, it also depends on the gray value of all pixels included in this node. The weight is the maximum value (resp. minimum value) of all pixels in a min-tree (resp. max-tree). If we consider median-tree with well balanced values, the weight (distance to the median) could be close to the average of gray values the one area. The attribute becomes:

$$sdw^2 = \frac{1}{N}\sum(X_i - W)^2 \qquad (6)$$

where W is the gray weight of the node accord the type of tree. As you see with the visual representation in Figure 6, the SDW results are close to the SD results. We have not yet compared deeply these two approaches, but we suggest it as future works. That is the reason why we implemented this alternative algorithm, witch saves time and memory of mean gray value evaluation.

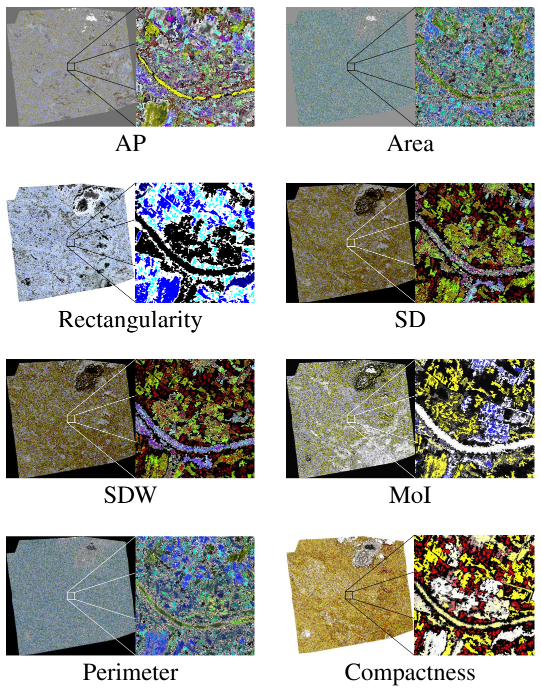


Figure 6. Featues profile with the Podgorica data Figure 8 using fake color (thresholds 10, 100, 5000 as RGB).

3. EXPERIMENTAL STUDY

3.1 Data sets

The first data set is a panchromatic image of size 628 × 700 pixels acquired by the IKONOS Earth imaging satellite with 1-m resolution in Reykjavik, Iceland. This data consists of six

Table 1. Computation time according to the feature profiles with remote sensing image in Figure 8 (run on AMD Ryzen Threadripper 1950X 16-Core Processor, 64GB ram).

	build tree	Area	perimeter	BoundingBox	mean xyz	mean gray	the function	filtering
Attribut Profile	8.73 s	0.53 s					n/a	1.99 s
Area	8.73 s	0.53 s					done	2.15 s
Rectangularity	8.73 s	0.53 s		1.70 s			0.06 s	2.16 s
SD	8.73 s	0.53 s				1.04 s	1.14 s	2.17 s
SDW	8.73 s	0.53 s					0.88 s	2.17 s
MoI	8.73 s	0.53 s			1.30 s		1.66 s	2.18 s
Perimeter	8.73 s	0.53 s	55.99 s				done	2.14 s
Compactness	8.73 s	0.53 s	56.18 s				0.046 s	1.67 s

thematic classes including residential, soil, shadow, commercial, highway and road. The image was provided with a training/test split including 22741 training samples and 98726 test samples. The input image together with its thematic ground truth map for testing and training sets are shown in Fig. 7.

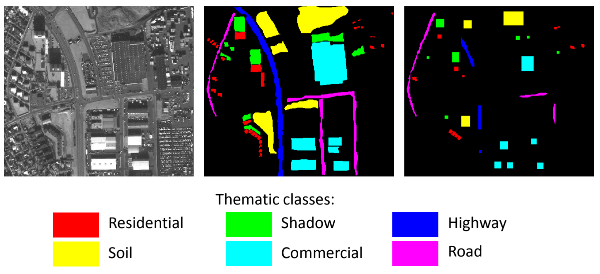


Figure 7. The Reykjavik data of size 628 × 700 pixels (left to right: panchromatic, thematic ground truth with 6 classes and training set)

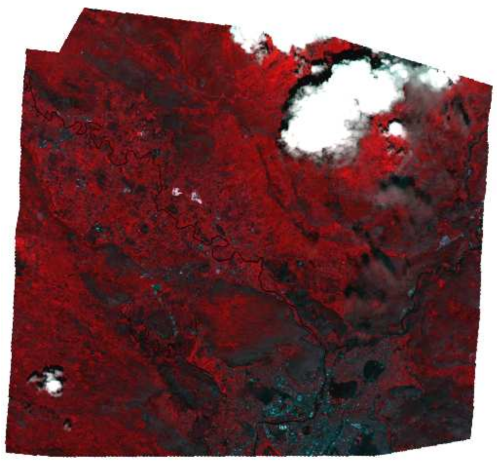


Figure 8. The Podgorica data of size 11948 × 11007 pixels represented in a false color composition: (Near infrered, Red and Green) as RGB.

The second data set is a multispectral image composed of $107,531,923$ pixels acquired by the Pleiades satellite in Podgorica, Montenegro. It is composed of four spectral bands (Blue, Green, Red and Near-Infrared) at 2-m spatial resolution. This image has been partially annotated (Faucqueur et al., 2019) and includes a single class of interest: the small wood features (SWF). To perform classification on this data, a subset of background pixels, *i.e.* pixels that do not belong to the SWF class, was selected automatically. In total, the ground truth is composed of $417,285$ SWF sample pixels and $17,211$ background pixels. Then, $16,000$ ground truth pixels were randomly selected for training a random forest classifier and the remaining pixels were used for testing. Figure 8 presents this image in false color composition: (Near infrered, Red and Green) as RGB.

3.2 Evaluation method

Recently, area attribute has been used for the classification of small woody features (Merciol et al., 2019). In this study, we will use a similar data, the high resolution Podgorica image, plus the Reykjavik dataset to compare the different attributes with the FPs. The evaluation on the Reykjavik data is based on the common approach used in the literature: a FP is computed on the data, the feature profiles of the training pixels are used to train a random forest and, then, the classification is performed on the entire ground truth. Regarding the classification of the small woody features, here is our evaluation process:

- We start by checking the consistency of the classification samples. The set of reference values is separated: (A) for training and (B) for prediction. We eliminate negative predictions in (B). They are due to an imprecision of the contours of the reference labeling. This provides us a cleaned subset called (C).
- We compute the accuracy for the SWF class of each AP and FP pair. The set (C) is randomly separated: (D) for the training and (E) for prediction. The quality of prediction corresponds to the accuracy between (E) predicted and the reference.
- We use the same input pixels set (E) for all tests. Only the attributes for the cut and for output FPs are different from one test to another.

Thanks to the partnership (IRISA, SIRS-CLS), we drive our study with the real industrial images. The result will be used to improve actual production. Specific hardware requirement is not necessary since the accuracy depends only on the choice of attributes. We provide computation time only to validate the real-time property of our approach.

3.3 Experimental setup

In this section, we explore APs and FPs for the classification of the remote sensing data described in section 3.1. From each image, we compute profiles using the following set of attributes: $A = \{$area, compactness (Li et al., 2013), moment of inertia (of a region) (Li et al., 2013), rectangularity (Rosin, 2003), square of the standard deviation of pixel gray levels (Kumar, Gupta, 2012), perimeter$\}$.

For most pairs (A_1, A_2) of attributes in the set A, we computed a FP using A_1 as a filtering rule and A_2 for projection. Our goals are (a) to compare the overall accuracy of FPs and APs in the classification of HR images and (b) to investigate the adequacy of each pair of attributes in this context.

The threshold values adopted in the filtering stage were selected manually. For the area filtering, we considered the same set of threshold values used in the computation of FPs in (Pham et al., 2017a) for the Pavia data, and the threshold values used in (Merciol et al., 2019) for the Podgorica data:

$$\begin{aligned}\lambda_{a,Rey} &= \{25, 100, 500, 1000, 5000, 10000,\\ &\quad 20000, 50000, 100000, 150000\}\\ \lambda_{a,Pod} &= \{1000, 2500, 5000, 7500\}\end{aligned}$$

For the perimeter attribute, we adopted the same threshold values used for area. Then, for the scale invariant attributes (moment of inertia, standard deviation and compactness), we considered the following set of thresholds commonly used in the literature:

$$\lambda_i = \lambda_s = \lambda_c = \{0.2, 0.3, 0.4, 0.5\}$$

The random forests trained on the Reykjavik and Podgorica data are composed of 100 and 64 trees, respectively. The experiments were performed on a machine with Intel Xeon Gold 6136 CPU, 3.00GHz and 263 GB of memory.

3.4 Results and discussion

Tables 2 and 3 present the classification scores and execution times for the Reykjavik and the Podgorica data, respectively.

In Table 2, we present the overall accuracy (OA), the average accuracy (AA) over all classes, and the kappa coefficient (κ) of the classification of the Reyjavik data using random forests. We also present the learning (train) and prediction times using this approach. Our baseline is the classification based solely on the panchromatic values (Pan) of this data. As mentioned previously, we evaluate the FPs based on different combinations of attributes in the filtering and projection steps. We can see that all tested APs and FPs outperform the baseline approach. In most cases, except for the FPs based on the 'moment of inertia' filtering, using other attributes for projection (instead of gray values) provides significant improvements. This outcome has already been discussed in (Pham et al., 2018a) with respect to the attributes area, moment of inertia and standard deviation. Here, we show that this is also the case for rectangularity, perimeter and compactness. For example, the FP computed using the compactness attribute both for filtering and for projection outperforms the compactness AP by 7.7%, 13.62% and 10.85% in terms of OA, AA and κ, respectively.

Regarding our evaluation on the Podgorica data, our two baselines are the classification based only on the spectral data (composed of four spectral bands), and on the spectral information concatenated to the NDVI channel and to the Sobel gradient of each band. For each method, we show in Table 3 the precision and recall with respect to the SWF class. While that including the NDVI and Sobel bands provides little improvement in recall, using APs and FPs improves the recall by up to 4.66%. Similar to the Reykjavik data, the highest scores are achieved using the FPs obtained with area and perimeter filtering. On the other hand, FPs with 'standard deviation' filtering does not give as good results on this data when compared to Reykjavik. The later may be due to the choice of threshold values, which is left for future research.

Table 2. Scores and execution times of the classification of the Reykjavik data using APs and FPs.

Projection	**OA (%)**	**AA (%)**	κ	**Train**	**Prediction**
Pan	62.53	52.45	0.5143	1s	1.32s
FPs with area filtering (21 dimensions)					
Gray values (AP)	80.68	76.5	0.7559	1.45s	1.87s
Area	84.54	80.2	0.8038	1.53s	1.79s
Rectangularity	82.33	79.25	0.777	1.54s	1.75s
Standard deviation	83.19	78.44	0.787	1.68s	1.94s
Moment of inertia	80.3	77.88	0.7524	1.52s	2.11s
Perimeter	82.22	78.9	0.7754	1.55s	1.86s
Compactness	82.69	78.9	0.7809	1.50s	1.84s
FPs with 'moment of inertia' filtering (9 dimensions)					
Gray values (AP)	64.94	55.83	0.5481	0.75s	1.76s
Area	64.6	55.38	0.5435	0.67s	1.7s
Rectangularity	64.83	55.73	0.5466	0.74s	1.82s
Standard deviation	64.92	55.73	0.5474	0.78s	1.76s
Moment of inertia	64.67	55.55	0.5445	0.76s	1.72s
Perimeter	64.59	55.39	0.5436	0.84s	1.79s
Compactness	64.7	55.58	0.5451	1.31s	1.9s
FPs with 'standard deviation' filtering (9 dimensions)					
Gray values (AP)	74.37	67.95	0.6725	0.92s	1.77s
Area	74.37	67.96	0.6726	0.95s	1.76s
Rectangularity	74.38	67.96	0.6726	0.95s	1.75s
Standard deviation	74.26	67.92	0.6712	0.96s	1.87s
Moment of inertia	74.33	68.03	0.6722	0.91s	1.80s
Perimeter	74.29	68.02	0.6717	0.91s	1.84s
Compactness	74.27	67.92	0.6715	0.96s	1.81s
FPs with perimeter filtering (21 dimensions)					
Gray values (AP)	78.36	73.88	0.727	1.40s	2.02s
Area	79.19	76.42	0.7377	1.38s	1.91s
Rectangularity	81.48	78.25	0.7664	1.41s	2.02
Standard deviation	81.31	77.1	0.7641	1.35s	1.97
Moment of inertia	78.67	76.39	0.7323	1.38s	2.12
Perimeter	79.47	76.96	0.742	1.30s	1.93s
Compactness	80.07	77.24	0.7492	1.45s	1.96s
FPs with compactness filtering (9 dimensions)					
Gray values (AP)	64.31	55.46	0.5407	1.29s	2.20s
Area	69.75	68.22	0.6231	1.66s	2.59s
Rectangularity	70.8	68.69	0.6355	1.83s	2.85s
Standard deviation	74.05	70.83	0.6742	1.73s	2.68s
Moment of inertia	73.37	69.92	0.6647	1.12s	2.98s
Perimeter	70.91	68.94	0.6373	1.57s	2.53s
Compactness	72.01	69.06	0.6492	1.6s	2.57s

4. CONCLUSION AND FUTURE WORKS

We have conducted a comparative study of attribute profiles and features profiles using different attributes (from which some new ones have been proposed) applied to remote sensing image classification task. Results obtained on both 1-m panchromatic Reykjavik image (IKONOS sensor) and 2-m multispectral Podgorica image (Pleiades sensor) have confirmed the effectiveness of both APs and in particular FPs in terms of classification performance as well as computation time. As a conclusion, FPs are more flexible than APs with their several output options. The choice of attribute for tree pruning plays a crucial role in such an approach. Area attribute has been proved to be the best one in our work as well as in several literature studies, while the newly proposed perimeter and compactness could provide other alternative options. All experiments have been

Table 3. Scores and execution times of the SWF classification of the Podgorica data using APs and FPs.

Projection	Precision (%)	Recall (%)	Train	Prediction
Spectral	99.93	87.94	0.95s	7'38s
Spectral + NDVI + Sobel	99.93	88.47	1.32s	8'
FPs with area filtering (30 dimensions)				
Gray values (AP)	99.98	93.13	1.63s	7'17s
Area	99.98	92.65	1.91s	7'19s
Rectangularity	99.98	92.82	1.86s	7'17s
Standard deviation	99.99	92.54	2.13s	7'30s
Moment of inertia	99.98	92.01	1.85s	7'22s
Perimeter	99.98	92.47	1.79s	7'9s
Compactness	99.98	92.70	1.81s	6'52s
FPs with 'moment of inertia' filtering (30 dimensions)				
Gray values (AP)	99.94	88.51	1s	10'3s
Area	99.94	88.15	1.61s	9'26s
Rectangularity	99.94	88.43	1.66s	9'16s
Standard deviation	99.94	88.38	0.99s	8'51s
Moment of inertia	99.94	88.51	0.87s	8'13s
Perimeter	99.94	88.18	1.09s	8'32s
Compactness	99.94	88.24	1.26s	8'34s
FPs with 'standard deviation' filtering (30 dimensions)				
Gray values (AP)	99.94	88.35	1.4s	9'45s
Area	99.94	88.35	1.36s	9'19s
Rectangularity	99.94	88.35	0.97s	8'45s
Standard deviation	99.94	88.35	1.48s	10'7s
Moment of inertia	99.94	88.35	0.98s	9'20s
Perimeter	99.94	88.35	0.99s	8'23s
Compactness	99.94	88.35	1.62s	8'22s
FPs with perimeter filtering (30 dimensions)				
Gray values (AP)	99.98	93.13	1.60s	6'46s
Area	99.99	92.78	2.02s	7'13s
Rectangularity	99.99	92.96	1.88s	7'54s
Standard deviation	99.99	92.57	1.87s	7'2s
Moment of inertia	99.98	92.21	1.76s	7'10s
Perimeter	99.98	92.25	1.92s	7'21s
Compactness	99.98	92.63	1.86s	6'53s
FPs with compactness filtering (30 dimensions)				
Gray values (AP)	99.94	88.39	2.23s	7'57s
Area	99.95	88.52	2.09s	8'12s
Rectangularity	99.96	88.84	2.35s	8'12s
Standard deviation	99.95	88.87	2.51s	8'21s
Moment of inertia	99.94	88.31	2.34s	8'37s
Perimeter	99.96	89.5	2.17s	8'12s
Compactness	99.96	89.11	2.33s	8'16s

conducted using our two libraries, TRISKELE and Broceliande, which are publicly available for the research community.

As mentioned in Section 2.3.4, our libraries include both SDW and SD as options to compute the standard deviation. Since the computation of SDW is much more efficient than SD, one of our future works could be to conduct a deep comparative study on their qualitative performance to determine if SDW could permanently replace SD in future use of this attribute. Another work is to integrate the APs and FPs without thresholds (threshold-free) into the libraries as well as make them available to other types of remote sensing data such as time series, Radar and Lidar data.

REFERENCES

Dalla Mura, M., Benediktsson, J. A., Waske, B., Bruzzone, L., 2010. Morphological attribute profiles for the analysis of very high resolution images. *IEEE Transactions on Geoscience and Remote Sensing*, 48(10), 3747–3762.

Das, A., Bhardwaj, K., Patra, S., 2020. A comparison of different filtering strategies used in attribute profiles for hyperspectral image classification. *International Conference on Intelligent Computing and Smart Communication 2019*, Springer, 1017–1026.

Faucqueur, L., Morin, N., Masse, A., Remy, P.-Y., Hugé, J., Kenner, C., Dazin, F., Desclée, B., Sannier, C., 2019. A new Copernicus high resolution layer at pan-European scale: small woody features. C. M. U. Neale, A. Maltese (eds), *Remote Sensing for Agriculture, Ecosystems, and Hydrology XXI*, 11149, International Society for Optics and Photonics, SPIE, 268 – 278.

Igel, C., Heidrich-Meisner, V., Glasmachers, T., 2008. Shark. *Journal of Machine Learning Research*, 9, 993-996.

Kumar, V., Gupta, P., 2012. Importance of statistical measures in digital image processing. *International Journal of Emerging Technology and Advanced Engineering*, 2(8), 56–62.

Li, W., Goodchild, M. F., Church, R., 2013. An efficient measure of compactness for two-dimensional shapes and its application in regionalization problems. *International Journal of Geographical Information Science*, 27(6), 1227–1250.

Merciol, F., al., 2017. Tree Representations of Images for Scalable Knowledge Extraction and Learning for Earth observation. https://gitlab.inria.fr/obelix/triskele/.

Merciol, F., al., 2018. Broceliande is a tools for classification base on TRISKELE and Random Forest. https://gitlab.inria.fr/obelix/broceliande/.

Merciol, F., Faucqueur, L., Damodaran, B. B., Rémy, P.-Y., Desclée, B., Dazin, F., Lefèvre, S., Masse, A., Sannier, C., 2019. Geobia at the terapixel scale: Toward efficient mapping of small woody features from heterogeneous vhr scenes. *ISPRS International Journal of Geo-Information*, 8(1), 46.

Pham, M.-T., Aptoula, E., Lefèvre, S., 2017a. Feature profiles from attribute filtering for classification of remote sensing images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(1), 249–256.

Pham, M.-T., Aptoula, E., Lefèvre, S., 2018a. Classification of remote sensing images using attribute profiles and feature profiles from different trees: a comparative study. *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, IEEE, 4511–4514.

Pham, M.-T., Lefèvre, S., Aptoula, E., 2017b. Local feature-based attribute profiles for optical remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 56(2), 1199–1212.

Pham, M.-T., Lefèvre, S., Aptoula, E., Bruzzone, L., 2018b. Recent developments from attribute profiles for remote sensing image classification. *arXiv preprint arXiv:1803.10036*.

Rosin, P. L., 2003. Measuring shape: ellipticity, rectangularity, and triangularity. *Machine Vision and Applications*, 14(3), 172–184.