DOCTORAT BRETAGNE LOIRE MATHSTIC

UNIVERSITÉ RENNES 2

Thèse de doctorat de

L'Université Rennes 2

École Doctorale N°601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Florent Guiotte

2D/3D discretization of Lidar point clouds:

Processing with morphological hierarchies and deep neural networks

Thèse présentée et soutenue à Rennes, le 25 janvier 2021
Unité de recherche : UMR 6554 LETG

Rapporteurs avant soutenance :

Beatriz Marcotegui	Professeur, MINES ParisTech, France
Pedram Ghamisi	Dir. de recherche, Helmholtz-Zentrum Dresden-Rossendorf, Germany

Composition du Jury :

Présidente :	À préciser	
Examinateurs :	Maria Vakalopoulou	MCF, CentraleSupélec, France
	Ewa Kijak	MCF, Univ. Rennes 1, France
	Clément Mallet	Dir. de recherche, IGN, France
	Dimitri Lague	Dir. de recherche, CNRS, France
	Pedram Ghamisi	Dir. de recherche, Helmholtz-Zentrum Dresden-Rossendorf, Germany
	Beatriz Marcotegui	Professeur, MINES ParisTech, France
Dir. de thèse :	Thomas Corpetti	Dir. de recherche, CNRS, France
	Sébastien Lefèvre	Professeur, Univ. Bretagne Sud, France

Invité :

Geoffroy Étaix	Docteur, DG Tellus Environment

Guiotte, Florent. 2D/3D discretization of Lidar point clouds : processing with morphological hierarchies and deep neural networks - 2021

Thèse de doctorat de

L'Université Rennes 2

École Doctorale N°601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Florent Guiotte

2D/3D discretization of Lidar point clouds:

Processing with morphological hierarchies and deep neural networks

Thèse présentée et soutenue à Rennes, le 25 janvier 2021
Unité de recherche : UMR 6554 LETG

Rapporteurs avant soutenance :

Beatriz Marcotegui	Professeur, MINES ParisTech, France
Pedram Ghamisi	Dir. de recherche, Helmholtz-Zentrum Dresden-Rossendorf, Germany

Composition du Jury :

Présidente :	À préciser	
Examinateurs :	Maria Vakalopoulou	MCF, CentraleSupélec, France
	Ewa Kijak	MCF, Univ. Rennes 1, France
	Clément Mallet	Dir. de recherche, IGN, France
	Dimitri Lague	Dir. de recherche, CNRS, France
	Pedram Ghamisi	Dir. de recherche, Helmholtz-Zentrum Dresden-Rossendorf, Germany
	Beatriz Marcotegui	Professeur, MINES ParisTech, France
Dir. de thèse :	Thomas Corpetti	Dir. de recherche, CNRS, France
	Sébastien Lefèvre	Professeur, Univ. Bretagne Sud, France

Invité :

Geoffroy Étaix	Docteur, DG Tellus Environment

INTRODUCTION

Contexte

Les systèmes LiDAR (pour *Ligth Detection and Ranging*) aéroportés modernes fournissent des scans 3D extrêmement précis des paysages et des villes. Les données LiDAR sont essentielles pour l'agriculture et la sylviculture, deux secteurs clés pour les prochaines décennies dans le contexte du changement climatique. Dans la sylviculture, les données LiDAR permettent d'évaluer les énergies renouvelables ou d'évaluer le stockage du carbone (Magnussen et al. 2018). L'analyse précise de ces données permet par exemple d'extraire un inventaire des arbres un par un, y compris de leur essences et de leur santé (Shendryk, Broich, Tulbure, McGrath et al. 2016). Les capacités topographiques élevées des systèmes LiDAR sont actuellement utilisées à une échelle fine pour modéliser les pentes pour l'irrigation de l'eau, ou à grande échelle pour obtenir la délimitation des bassins versants utilisée entre autres pour prévenir la pollution de l'eau (Pelletier et Orem 2014). Les données LiDAR sont également adaptées à la détection et à la cartographie des risques de glissement de terrain (Passalacqua et al. 2015). Dans un contexte urbain, les données LiDAR fournissent un moyen unique de caractériser les bâtiments, la végétation et le mobilier urbain, ce qui est essentiel pour connaitre l'occupation des sols ou pour l'urbanisme. Parmi les sujets d'intérêt connexes, citons les îlots de chaleur urbains, l'imperméabilisation des sols, le ruissellement des eaux de pluie et les modèles d'inondation (Yan, Shaker et El-Ashmawy 2015).

Pour ces raisons, de nombreux pays ont développé, ou développent actuellement, des projets de couverture aérienne LiDAR. Pour n'en citer que quelques-uns, les Pays-Bas, la Belgique, le Danemark, la Suisse et la Suède disposent à ce jour d'une couverture complète et accessible en *open data*. D'autres pays ont une couverture haute résolution

partielle, comme les États-Unis[1] (2 593 827.63 km^2 couverts) ou le Canada[2], qui met actuellement à jour ses cartes topographiques de la partie sud du pays avec du LiDAR haute résolution. Récemment, la France a prévu, dans le cadre du plan de relance et d'urgence pour 2021, une couverture à haute densité des zones présentant des problèmes forestiers et agricoles [3]. L'Institut national de l'information géographique et forestière (IGN) devrait assurer la couverture haute densité LiDAR (10 points/m^2). À une échelle plus fine, les villes acquièrent également des données LiDAR à très haute résolution avec des politiques de données ouvertes. Dans cette perspective, les Observatoires des Sciences de l'Univers (OSU) des villes de Nantes et Rennes ont acquis un système LiDAR multispectral en 2015 et plusieurs acquisitions ont été réalisées sur les villes de Nantes, Rennes et les zones d'intérêt environnantes.

La quantité de données augmente de manière exponentielle en termes de couverture et de résolution. Cependant, les données LiDAR actuelles ne sont pas encore pleinement exploitées en raison du manque d'outils méthodologiques efficaces aptes à traiter ces données spécifiques.

Concernant les outils, les hiérarchies morphologiques sont utilisées depuis une décennie dans l'imagerie de télédétection. Les approches morphologiques sont connues pour extraire des caractéristiques multi-échelles fiables tout en étant extrêmement efficaces sur le plan calculatoire. Dans le même temps, la formidable percée de l'apprentissage profond en vision par ordinateur a bouleversé la communauté de la télédétection. L'apprentissage profond promet une précision sans précédent sur une grande variété de problèmes complexes.

Cette thèse évalue la potentialité des hiérarchies morphologiques et des réseaux neuronaux profonds sur les données LiDAR au moyen de plusieurs stratégies de discrétisation.

Cette thèse de doctorat a été soutenue par la Région Bretagne (projet doctoral CAMELOT), Tellus Environment et l'équipe Obelix de l'IRISA. Cette thèse s'est déroulée au Laboratoire Environnement Télédétection et Géomatique (LETG) à Rennes, à l'Institut de recherche en informatique et systèmes aléatoires (IRISA) à Vannes et chez Tellus Environment à Bruz. Une partie de la thèse s'est déroulée à la *Chinese Academy of Sciences* de Pékin dans le cadre d'un échange international.

[1] https://coast.noaa.gov/inventory/
[2] https://open.canada.ca/data/en/dataset/957782bf-847c-4644-a757-e383c0057995
[3] https://www.economie.gouv.fr/plan-de-relance/profils/administrations/doter-france-couverture-lidar-haute-densite-densite-territoires-forestiers-agricoles

Problématique

Nuages de points

Le premier défi lié à l'utilisation des données LiDAR est la quantité d'informations à traiter. Une acquisition de données LiDAR correspond à des centaines de millions de points sur un quartier, et jusqu'à plusieurs milliards pour une ville de la taille de Rennes. De plus, la quantité de données LiDAR augmente avec les politiques publiques et les évolutions technologiques. Les nouveaux systèmes enregistrent de plus en plus de points le long de la composante verticale et les systèmes multispectraux multiplient le nombre d'échantillons par longueur d'onde.

Le deuxième défi concerne la nature des données LiDAR. Les systèmes LiDAR capturent des nuages de points se trouvant dans un espace tridimensionnel. Contrairement aux images conventionnelles qui se trouvent dans un espace discret régulier, pour lesquelles il existe un grand nombre d'outils de vision et de traitement d'image, les nuages de points non structurés nécessitent des approches spécifiques en raison de l'irrégularité de la répartition des points dans l'espace.

Enfin, les surfaces captées par les systèmes LiDAR sont échantillonnées de façon irrégulière. Outre les artéfacts radiométriques, les données LiDAR sont sujettes à des occlusions et à une densité irrégulière. Pourtant, de nombreux algorithmes sont sensibles aux données manquantes et aux échantillonnages irréguliers.

Hiérarchies morphologiques et apprentissage automatique

Il existe peu de travaux explorant la totalité du potentiel des données LiDAR avec des hiérarchies morphologiques. En fait, dans ses définitions fondamentales, la morphologie mathématique repose sur la connectivité entre les pixels d'une image. Étant donné qu'il n'existe pas de définition simple du voisinage dans les nuages de points répartis de façon irrégulière, de nombreux travaux ont choisi de transformer les données en images. Cependant, une quantité importante d'informations significatives peut être perdue.

De façon similaire, il n'existait pas, jusqu'à récemment[4] , de réseau profond conçu pour traiter les nuages de points sans structure intermédiaire. La révolution de l'apprentissage profond est née de la vision par ordinateur appliquée à la reconnaissance des formes, avec des progrès notamment en segmentation et classification d'images. Afin de bénéficier de l'écosystème d'apprentissage profond le plus avancé, il est nécessaire de s'adapter à

[4]Ce champ d'étude est en pleine expansion, avec prêt de 50 publications sur le sujet pendant ces trois dernières années (Guo, Wang et al. 2020).

l'entrée discrète requise par la plupart des réseaux de neurones disponibles actuellement.

Contributions

Cette thèse a pour but de relever les défis liés aux nuages de points LiDAR en utilisant la puissance de l'apprentissage automatique, et plus précisément de l'apprentissage profond. Cependant, les approches d'apprentissage profond nécessitent un nombre important de données d'entraînement. Nous nous intéressons donc également aux hiérarchies morphologiques qui permettent d'extraire des caractéristiques significatives pouvant améliorer la classification supervisée lorsque peu de données d'entraînement sont disponibles. De plus, ces représentations hiérarchiques fournissent une description de la forme des structures qui peut être utilisée sans connaissance préalable.

Dans cette étude, nous visons donc un double objectif :

1. Utiliser les hiérarchies morphologiques pour analyser les données LiDAR et extraire efficacement les caractéristiques à plusieurs échelles ;

2. Tirer parti des réseaux de neurones profonds afin d'obtenir une segmentation et une classification précise des données.

Dans ce but, nous définissons et évaluons différentes stratégies de discrétisation des données. Dans une première partie, nous réorganisons les nuages de points LiDAR en grilles régulières 2D. Nous proposons de dériver plusieurs caractéristiques, en essayant d'extraire une description complète de l'altitude et des valeurs spectrales ainsi que des informations spécifiques. Dans une deuxième partie, nous réorganisons les nuages de points en grilles régulières 3D. Les grilles régulières sont suffisantes pour fournir le contexte de voisinage nécessaire aux hiérarchies morphologiques et les grilles proposées sont adaptées aux couches d'entrée des réseaux de neurones profonds.

Dans les chapitres suivants, nous utilisons des hiérarchies morphologiques sur des données LiDAR discrétisées pour créer des caractéristiques multi-échelles. Nous utilisons ces caractéristiques multi-échelles pour effectuer une classification supervisée des nuages de points urbains afin de produire des cartes de l'occupation du sol. Nous proposons également d'explorer l'espace d'attributs construit à partir du LiDAR pour extraire de manière interactive des structures d'intérêt de grandes zones montagneuses recouvertes de forêts tropicales humides. Nous utilisons ensuite plusieurs réseaux de neurones profonds pour fournir une segmentation et une classification dans un contexte urbain.

D'un point de vue logiciel, la bibliothèque Python SAP[5] a été développée et publiée pendant cette thèse de doctorat. SAP est une bibliothèque open source visant à calculer des descriptions morphologiques multi-échelles et à créer des fonctions de distribution de probabilités basées sur des attributs. Le projet est documenté et vise une bonne maintenabilité.

Organisation du manuscrit

Ce manuscrit est structuré comme suit :

- Le Chapitre 2 dresse un état des lieux de l'utilisation des données, des méthodes de classification traditionnelles et des approches d'apprentissage profond.
- La Partie II traite de la réorganisation des nuages de points LiDAR dans des grilles régulières en 2D. Nous dérivons ensuite des rasters de caractéristiques LiDAR pour produire une cartographie de l'occupation du sol et effectuer une analyse interactive des données.
- La Partie III étend la stratégie de grille avec une troisième dimension et réorganise les nuages de points LiDAR dans des grilles régulières 3D. Les grilles de voxels générées sont utilisées afin d'appliquer des filtres morphologiques sur les nuages de points et de les classifier, ou encore comme données d'entrée dans des réseaux de neurones profonds.
- Dans la Partie IV, nous résumons les travaux menés dans le cadre de cette thèse puis nous soulignons les perspectives ouvertes par nos travaux, en particulier pour une caractérisation efficace et précise des nuages de points.

[5] `https://gitlab.inria.fr/fguiotte/sap`

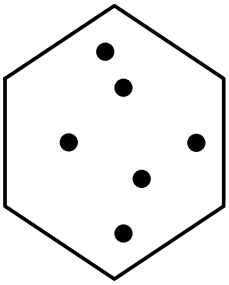
CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I.

Background



CHAPTER 1.

INTRODUCTION

Contents

1.1. Context

Modern airborne LiDAR (for Light Detection and Ranging, see Figure 1.1) systems provide extremely precise 3D scans of landscapes and cities. LiDAR data are essential for agriculture and forestry, two key sectors for the coming decades in the context of climate change. In forestry, LiDAR allows to assess renewable energy or evaluate carbon storage (Magnussen et al. 2018). Precise LiDAR data analysis allows to extract tree wise inventory, including tree species and tree health (Shendryk, Broich, Tulbure, McGrath, et al. 2016). The high topographic capabilities of LiDAR systems are currently used at a fine scale to model slopes for water irrigation, or at large scales to obtain drainage basin delineation, used among other things to predict water pollution (Pelletier and Orem 2014). LiDAR data are also suitable for the detection and mapping of landslide hazards (Passalacqua et al. 2015). In an urban context, LiDAR data provide a unique way to characterize buildings, vegetation and urban furniture, which is essential for land use and urban planning. Several related topics of interest includes urban heat island, soil sealing, rainwater runoff and flooding models (Yan, Shaker, and El-Ashmawy 2015).

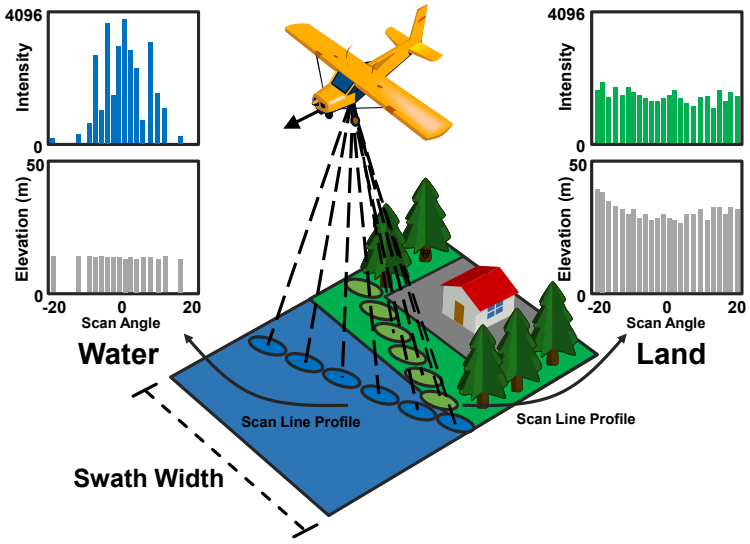


Figure 1.1.: Illustration of airborne LiDAR scanning, the LiDAR system measure distances of objects with a laser along the flight path. Intensities are recorded with the elevations of sensed surfaces. (Illustration from Yan, Shaker, and LaRocque 2019)

For these reasons, many countries have developed, or are currently developing airborne LiDAR coverage projects. To name but a few, the Netherlands, Belgium, Denmark, Switzerland and Sweden have a complete coverage openly available to date. Other countries have a partial high resolution LiDAR coverage, such as the United States of America[1] (2 593 827.63 km^2 covered) or Canada[2], which is currently updating its topographic maps of the southern part of the country with high resolution airborne LiDAR systems. Recently, France has been planning, as part of the recovery and emergency plan for 2021[3], a high density LiDAR coverage of areas with forestry and agricultural challenges. French national geographic institute *Institut National de l'Information Géographique et Forestière* (IGN) should provide the high density LiDAR coverage (10 points/m^2). On a finer scale, cities are also acquiring very high resolutions LiDAR data with open data policies. In this perspective, the Observatories of Universe Sciences (OSU) of Nantes and Rennes have acquired a multispectral LiDAR system in 2015 and several acquisitions were performed over the cities of Nantes, Rennes and surrounding areas of interest.

Technological evolutions on LiDAR systems lead to an increasing precision. Over the

[1] https://coast.noaa.gov/inventory/
[2] https://open.canada.ca/data/en/dataset/957782bf-847c-4644-a757-e383c0057995
[3] https://www.economie.gouv.fr/plan-de-relance/profils/administrations/doter-france-couverture-lidar-haute-densite-densite-territoires-forestiers-agricoles

last decades, the acquisition rate of the systems from the company *Teledyne Optech* increased steadily. Lastly, the acquisition rates improved from 1 800 000 points/s for the *Titan* system to 4 400 000 points/s for the *Galaxy* system (Hartsell et al. 2016; Fernandez-Diaz et al. 2016).

The quantity of data increases exponentially in coverage and resolution. However, actual datasets are not yet fully exploited due to the lack of efficient methodological tools for this specific type of data.

Morphological hierarchies have been used for a decade in remote sensing imaging. Morphological structures are known to extract reliable multi-scale features while being extremely computationally efficient[4]. In the mean time, the tremendous breakthrough of deep learning in computer vision has shaken up the remote sensing community. Deep learning promises unprecedented accuracy on a wide variety of complex issues.

This thesis evaluates the potentiality of morphological hierarchies and deep neural networks on LiDAR data by means of several discretization strategies.

This PhD thesis was supported by *Région Bretagne* (*CAMELOT* doctoral project), *Tellus Environment* and the team *Obelix* from *Institut de Recherche en Informatique et Systèmes Aléatoires* (IRISA). This thesis was carried out at *Laboratoire Environnement Télédétection et Géomatique* (LETG) in Rennes, IRISA in Vannes and Tellus Environment in Bruz. A part of this work was done during a three months international exchange with the Chinese Academy of Science in Beijing.

1.2. Definition of the problem

Over a sea of point clouds

The first challenge when using LiDAR data is the amount of information to process (see for example a small part of the city of Rennes Figure 1.2). A single acquisition of LiDAR data over a district can provide hundreds of millions points, and up to several billions for a city of the size of Rennes. Moreover, the quantity of data will continue to grow according to public policy and technological evolutions. New systems can record more and more points along the vertical component and multispectral systems multiply the samples count per wavelength.

The second challenge concerns the nature of the data itself. LiDAR systems capture point clouds lying in a continuous three-dimensional space. Unlike conventional images

[4] A recent publication reported the use of morphological hierarchies on more than 120 TiB of remote sensing imagery (Merciol, Faucqueur, et al. 2019).

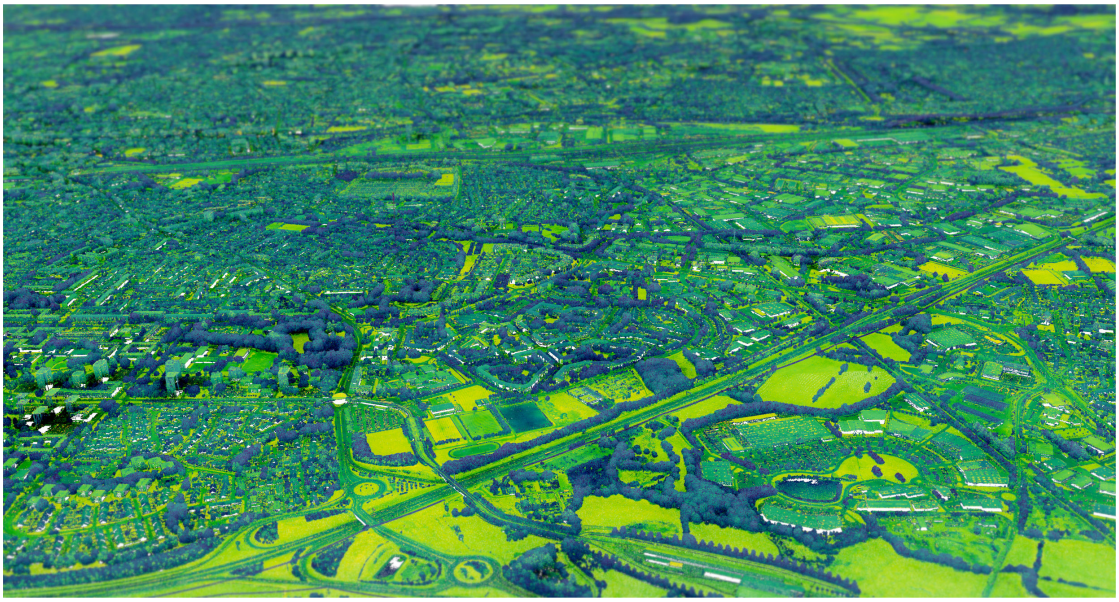


Figure 1.2.: LiDAR point cloud of the city periphery of Rennes. Residential neighborhoods are on the left, sport fields are on the center, a beltway crosses the image, fields and a shopping mall are on the right of the image. The points are colored according to the return intensity of the infra-red laser, which is commonly used to characterize vegetation.

that lie in a regular discretized space, for which a large number of computer vision and image processing tools are available, the unstructured point clouds require dedicated approaches.

Finally, the surfaces scanned by LiDAR point clouds are irregularly sampled. Besides radiometric artifacts, LiDAR data is also subject to occlusions and irregular density. Yet many algorithms are sensitive to missing data and irregular sampling.

Morphological hierarchies and machine learning

Only few works exploring the full potential of LiDAR data with morphological hierarchies exists. In fact, in its fundamental definitions, mathematical morphology relies on connectivity between pixels of an image. Since there is no straightforward definition of neighborhood in irregularly distributed point clouds, many works chosen to transform LiDAR data in images. However by doing so, a substantial amount of meaningful information can be lost.

Up until recently, there was no deep network designed to process point clouds without intermediate structures[5]. The deep learning revolution originated from computer vision for pattern recognition, segmentation and classification applied to images. To benefit from the most advanced deep learning ecosystem, it is necessary to adapt to the discrete input required by the different neural networks available so far.

1.3. Contributions

To address the numerous challenges raised by point clouds, we have unsurprisingly opted for the latest developments in machine learning, and more precisely in deep learning. However, deep learning approaches usually require a rather large amount of training data. Conversely, the morphological hierarchies are known to extract meaningful features that can enhance supervised classification when few training data are available. Furthermore, they can also provide shape description of structures which can even be used without prior knowledge.

The objectives of this doctoral project are thus twofold:

1. Use the morphological hierarchies to analyse LiDAR data and extract multiscale features efficiently.

[5]This specific field is even booming, with nearly fifty publications over the past three years. See the survey on deep learning for 3D point clouds from Guo, Wang, et al. (2020).

2. Take advantage of deep neural networks to obtain precise segmentation and classification of LiDAR data.

To this end we define and evaluate different discretization strategies of LiDAR data. In a first part, we re-organise the point clouds into 2D regular grids. We propose to derive several LiDAR features, trying to extract complete elevation description and spectral values along with LiDAR specific information. In a second part we re-organise the point clouds into 3D regular grids. The regular grids are sufficient to provide the neighboring context needed for the morphological hierarchies, and the proposed grids are adapted to the input layers of state-of-the-art deep neural networks.

In the following chapters, we use morphological hierarchies on discretized LiDAR data to create multiscale features. We use these multi-scale features to perform supervised classification of urban point clouds to produce land cover maps. We also propose to explore the attribute space built from the morphological hierarchies to extract interactively structures of interest from large mountainous areas with tropical rainforests. We then use several deep neural networks to provide segmentation and classification in an urban context.

From a software perspective, the Python package Simple Attribute Profiles (SAP)[6] was developed and published during this PhD thesis. SAP is an open source library aiming to compute several state-of-the-art multiscale morphological features and create distribution probabilities of attributes. The project is well documented and aims at a good maintainability (documentation enclosed Appendix C). This package was notably used for a survey on multiscale morphological features (see Appendix B).

1.4. Organisation of the manuscript

This manuscript is structured as follow:

- The next chapter of Part I draws a state-of-the-art of LiDAR data usage, traditional classifications methods and deep learning approaches. We briefly cover related works on mathematical morphology and hierarchical representations as well.

- Part II deals with re-organisation of point clouds in 2D regular grids. We then derive rasters to produce land-cover mapping and perform interactive LiDAR data analysis. The approaches discussed in this part provides 2D results (see Figure 1.3.a).

[6] `https://gitlab.inria.fr/fguiotte/sap`

- Part III extends the grid strategy with a third axis and re-organise the point clouds in 3D regular grids. The derived voxel grids are used to apply morphological filters on point clouds, and perform point cloud classification. The methods discussed in this part take advantage of the 3D grids to provide point cloud classification (see Figure 1.3.b) and 2D land cover maps (see Figure 1.3.c).

- Part IV gives a final overview of the works conducted in this thesis. This part concludes with an overview of open challenges and perspectives on future research directions. These perspectives open up new possibilities for efficient and highly precise point cloud characterization (see Figure 1.3.d)

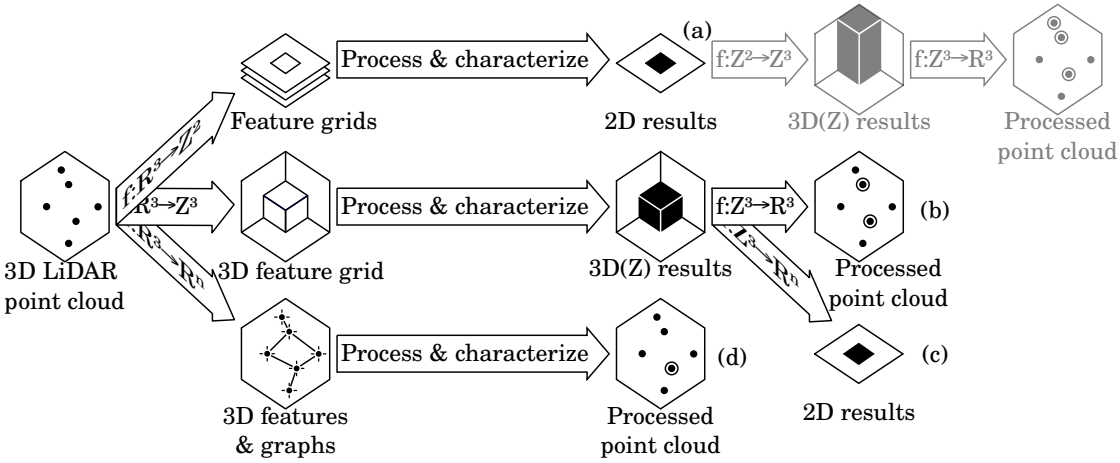


Figure 1.3.: A global overview of the different LiDAR point cloud workflow addressed in this manuscript. The two top rows concern discretization of the point clouds to 2D grids and 3D grids. The greyed part emphasizes methods explored in related works but not considered in this thesis. The last row performs directly the filtering or the classification in the point cloud space.

CHAPTER 2.

STATE OF THE ART

Contents

This chapter aims at introducing the main works associated with the topic of this PhD. In a first part, we present LiDAR systems, data & usage. Then in section 2.2, we introduce the main methods developed to process resulting point clouds, including deep neural networks. As data often embed multiscale objects, we present in section 2.3 hierarchical techniques, issued from mathematical morphology, to perform a multiscale analysis of data represented in 2D/3D regular grids. We then conclude by the needs of new tools based on mathematical morphology to process LiDAR data.

2.1. LiDAR systems & remote sensing usage

LiDAR (Light Detection And Ranging) systems enable to capture high resolution 3D points clouds used in a variety of applications.

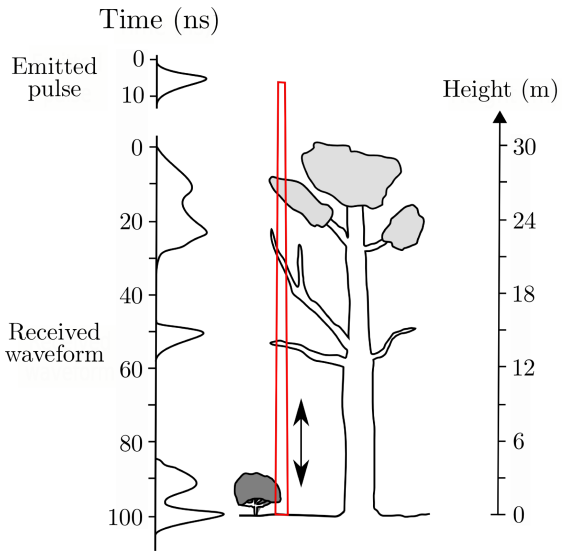


Figure 2.1.: LiDAR system in operation, the laser beam is displayed in red, the interaction between light and surfaces are detected by the system (left). Peaks in the signal trigger the recording of points. Multi-echo systems can record several points per beam while single-echo detect only the first one. Full-Waveforms (FWF) systems are able to record the complete waveform (Illustration from MALLET 2010).

2.1.1. LiDAR technology

The main principle of LiDAR systems is to send a laser pulse and to analyze the backscattered signal (whose peaks enable to localize the position of surfaces, see Figure 2.1). Combined with Global Positioning System (GPS) and Inertial Measurement Unit (IMU), one can extract in a precise way the coordinates associated with relative information (intensity of the peak of the signal for example) of observed surfaces.

Many technical innovations have been made over the past decades. For example multi-echo LiDAR systems record several echoes for a single laser beam, instead of recording only the maximum of the backscattered signal in initial LiDAR systems. This is particularly interesting in forested areas since they can acquire the top of the canopy while detecting ground points below the canopy. Some multi-spectral systems also embed several lasers in order to acquire point clouds of various nature: some wavelenghts are suitable for the characterization of the vegetation (such as infra-red) while others are preferred to perform bathymetry (such as visible green). In addition, the multi-spectral LiDAR data allow to derive some indices commonly used in remote sensing. For example some authors use LiDAR data to process indices related to vegetation chlorophyll content,

vegetation water stress and others (Hakala et al. 2015; Eitel et al. 2016). Let us also note that some systems are able to record the entire backscattered signal from the laser (and not only the position of the peaks of the signal) with a sampling period of 1 ns. Such FWF systems are useful to analyse in a more precise way all objects encountered in the scene by characterizing the underlying surface response to light (Mallet and Bretar 2009). From such data one can also extract dense point clouds with various discretization methods.

In contrast to other methods to acquire 3D data, such as photogrammetry, let us note that:

- LiDAR systems are active sensors, using lasers to measure the distance of objects; LiDAR systems are far less sensitive to illumination and shadows.
- LiDAR systems use pointwise sampling of the underlying surfaces, usually returning the position of the backscattered signal along with its intensity. The data is almost instantaneously available during the sensing with minimal processing.

Various comparisons exists between photogrammetry and LiDAR acquisitions (Gil et al. 2013; Matikainen et al. 2016): depending on the specific topic of the survey, each technique has its advantages and challenges. A complementary approach can benefit from the strengths of each one. For instance in the field of forestry, photogrammetry is known to be better to detect the canopy while LiDAR is better to detect the ground (Jayathunga et al. 2018).

2.1.2. Usage in remote sensing

LiDAR point clouds have led to various applications for many fields of remote sensing in the past decades.

Urban environments

The high accuracy of Airborne Laser Scanning (ALS) has been exploited for several urban applications, LiDAR data provide in particular a unique way to characterize buildings and vegetation. The urban context is conducive to the precise definition of problems based on labeling vegetation and urban structures (see labelized point cloud from the city of Montreal[1] Figure 2.2). Thus, many studies focus on pointwise classification of LiDAR data (Chehata et al. 2009; Mallet, Bretar, et al. 2011), building and

[1] `https://donnees.ville.montreal.qc.ca/dataset/lidar-aerien-2015`

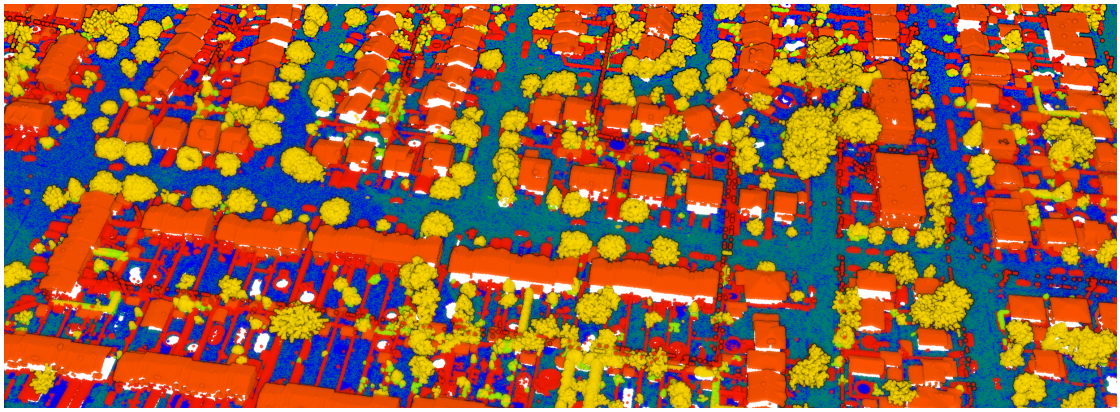


Figure 2.2.: Urban classes from Montreal LiDAR dataset. Each point of the point cloud is associated with a label visualized by a color (e.g. ground in blue, houses in orange, trees in yellow).

object detection (Niemeyer et al. 2014) or point cloud segmentation (Guinard and Landrieu 2017). Other works use LiDAR data as valuable input for advanced modeling applications, such as urban heat island (Chun and Guldmann 2014) or the prediction of building ages and energy performances (Tooke et al. 2014).

Apart from airborne LiDAR yielding to ALS, some systems are mounted on vehicles (known as Mobile Laser Scanning (MLS) systems) or are fixed (known as Terrestrial Laser Scanning (TLS) systems) and are particularly used in natural (mountain, cliffs, etc) or urban environments. In urban environments, MLS are more accurate than ALS at street level, especially for facades and street furniture, but lack the overall topographical and built context. In a complementary way, ALS and MLS can easily be used together though the fact that very few studies exists in this specific setup.

Forestry and agriculture

LiDAR data have been used significantly in forestry applications. The vertical distribution of LiDAR point clouds is indeed of great interest for Above Ground Biomass (AGB) estimation. While some works have used LiDAR data to assess the biomass by detecting the canopy and the ground (Nord-Larsen and Schumacher 2012), other researches have focused on finer analyses by detecting several stratum of vegetation (e.g. shrubs below trees) for multi-layered forests (Ferraz et al. 2012). When using dense point clouds, several works report the delineation of single trees in forest from crown to trunk (Li et al. 2012; Mongus and Žalik 2015) or from trunk to crown (Shendryk, Broich, Tulbure, and Alexandrov 2016) (see Figure 2.3) and assess precise in-

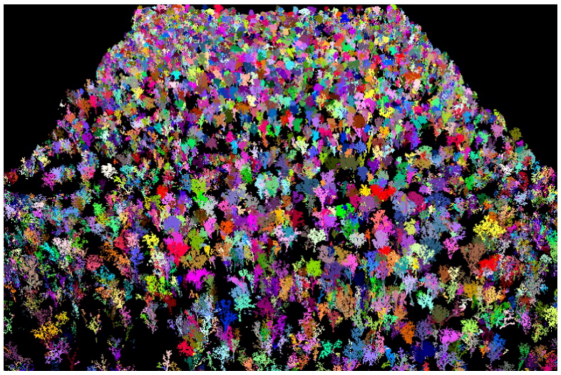


Figure 2.3.: Individual tree segmentation from Shendryk, Broich, Tulbure, and Alexandrov (2016). Each individual tree is assigned a specific color.

ventory of species and tree health (Cao et al. 2016; Shendryk, Broich, Tulbure, McGrath, et al. 2016).

These challenges are also relevant in the agriculture domain, for example to automate orchard tree inventory (Jang et al. 2008). Others agriculture-related studies, such as grapevine parcel delimitation and vine row orientation (Mathews and Jensen 2012), have successfully exploited LiDAR point clouds.

More recently, promising deep learning methods adapted to point clouds (see Section 2.2.1) have enabled to enhance tree species and tree health detection with small Unmanned Aerial Vehicle (UAV) LiDAR system (Briechle et al. 2020), suggesting great opportunities for the future of agriculture and forestry surveying.

Geosciences and archaeology

ALS technology is also widely used in the fields of geosciences and archaeology. These contexts are conducive to detect structures in LiDAR data, such as natural or anthropomorphic landforms.

As an illustration in geomorphology, Brodu and Lague (2012) developed a multiscale descriptor to separate elementary classes such as riparian vegetation and ground in fluvial environment or fresh surfaces and rockfall in cliff environments. Passalacqua et al. (2015) proposed a complete study of high resolution LiDAR data usage and methods for the understanding of mass and energy transfer, that is of high interest in geomorphology and hydrology. From another perspective, we refer the reader to Eitel et al. (2016) which review various LiDAR data combinations (spatial, spectral, temporal) and topics of interest for earth and ecological sciences.

High resolution multi-echos LiDAR data (i.e. more than 10 points per m^2) enable to penetrate the vegetation and therefore to retrieve ground below vegetation. Numerous archaeology works have used this specific property to detect ancient settlements covered by foliage, for example, mapping Maya landscape under tropical rainforest (Chase et al. 2011). Recent works use machine learning techniques and multiscale analysis to automate the detection of archaeological sites (Guyot et al. 2018).

When analyzing the approaches developed in forestry, urban environments or geosciences, one can observe that they share few common workflows, probably because each field has its own challenges and usages.

In the following we present processing methods devoted to the specificity of point clouds.

2.2. Methods for point cloud processing

In this section, we present a comprehensive overview of common methods to process point clouds.

2.2.1. The point cloud space

In its original state, the atomic element of raw LiDAR data is a $3D$ coordinate in $\mathbb{R}^3$ associated with the corresponding intensity. Many works have focused on characterizing or classifying points in this original domain. Since a position in space (and an intensity value) bears little information, a number of studies have proposed to attach hand crafted features or more recently to aggregate spatial information with neural networks before processing the raw point clouds.

Hand crafted features and machine learning

For each point of the point cloud, several LiDAR features can be computed (Chehata et al. 2009; Mallet, Bretar, et al. 2011; Niemeyer et al. 2014). These features are valuable for classification or segmentation applications and are related to normalized intensity, number of echoes for multi-echo systems, heights characteristics (height above ground, variance...), FWF derived features and local neighbourhood analysis. The main approaches are presented below.

Local neighbourhood: spheres and cylinders The idea here is to compute in each point some criteria related to its local neighborhood. While early works only rely on

point-wise characteristics, ignoring the vicinity of the point, some authors exploit the spatial context around each point by searching for neighboring points within spheres (Chehata et al. 2009; Brodu and Lague 2012) or cylinders (Chehata et al. 2009; Niemeyer et al. 2014) to extract contextual information.

In order to efficiently assess neighbouring points, the most common approach is to rely on a Kd-tree (Bentley 1975), modified to perform nearest neighbors queries (Friedman et al. 1977). This Kd-tree is a hierarchical data structure that can be used for 3D point neighbors search. The Kd-tree is a binary tree in which the root represents the whole dataset. The nodes of the Kd-tree are divided in order to balance the point count on the axis with the most variance. This structure provides an efficient way to group neighbor points in nodes and allows to eliminate large portions of space during the exact nearest neighbor searching. To further increase performances, an approximate nearest neighbour search partly based on Kd-tree was proposed and the library Fast Library for Approximate Nearest Neighbors (FLANN) was made openly available (Muja and Lowe 2009).

In (Chehata et al. 2009; Brodu and Lague 2012), the authors derive local dimensionality features in spheres. For each point, a Principal Component Analysis (PCA) is performed on the coordinates of the neighboring points and the associated eigenvalues provide useful information of the local geometry: when a single (resp. two and three) eigenvalue(s) explain(s) the main neighbors variance, the points are distributed in one (resp. two and three) dimension(s). As Brodu and Lague (2012), one can then characterize the geometry with spheres of several diameters for each point, thus providing a multiscale analysis of the point with its surrounding.

Features classification While some authors use simple linear models such as the Conditional Random Field (CRF) (Niemeyer et al. 2014), most of the works use non-linear classifiers, such as the Support Vector Machine (SVM) or the Random Forest (RF). The SVM from Vapnik (1995) is often used with the kernel trick to act like a non-linear classifier (Boser et al. 1992). Several works propose a multi-class SVM, one common strategy when classifying point clouds is named "One against one", where pairs of classes are tested with binary classifiers and the class of the point is assigned by voting (Pontil and Verri 1998). The RF classifier from Breiman (2001) has been shown to be well-suited for remote sensing applications (Belgiu and Drăguţ 2016). This classifier is adapted to differentiate several classes of objects. The RF accepts a high number of dimensions (i.e. features) where others classifiers (e.g. SVM) are sensitive to high dimensionality and require prior feature reduction (e.g. with PCA) or prior feature

selection. The RF was used for point clouds in numerous works (Chehata et al. 2009; Guo, Chehata, et al. 2011; Ni et al. 2017; Landrieu, Raguet, et al. 2017). Moreover, the RF allows to compute easily features importance by permutations of variables and measurement of performance loss (Breiman 2001).

After point-wise classification, a final stage of contextual regularization can be used. Contextual approaches based on a CRF and/or a Markov Random Field (MRF) have been proposed (Niemeyer et al. 2014; Najafi et al. 2014) in this direction. The corresponding results are greatly improved by imposing such spatial smoothness. Another way to perform a structured regularization has been proposed by Landrieu, Raguet, et al. (2017) by optimizing a cost-function associated with specific smoothing terms. However such models lack of interpretability contrary to the previous probabilistic regularizations. Nevertheless, the structured regularization allows to use better functions and regularizers and to provide fast algorithms compared to slow and memory intensive statistical models.

Learned features: deep learning

Recently, many efforts have been proposed to transfer the successes of deep learning from computer vision to 3D point clouds. PointNet from Qi, Su, et al. (2017) acts as a seminal work in the processing of 3D point clouds with deep learning. This neural network uses multilayer perceptrons to process the points directly. PointNet is, however, only suitable for small point clouds with regular density. Above all, the lack of local spatial relationships of PointNet limits the performances and makes it unsuitable for large LiDAR point clouds. This main drawback was addressed shortly after with PointNet++ (Qi, Yi, et al. 2017) by introducing a hierarchical neural network to apply locally PointNet on subsamples that form a nested partition of the full point cloud. In this work authors also addressed the varying density by combining features from multiple scales.

Other works addressed the problem of spatial relationships for large scale LiDAR sets. In Landrieu and Simonovsky (2018), authors proposed a structure named "superpoint graph" to partition the point cloud in homogeneous elements while keeping the contextual relationships between the elements. Each homogeneous element (namely superpoint) from the partition is embedded by a PointNet network. A recursive neural network refine the embeddings by message passing along the edges of the graph (namely superedges) to produce the final classification.

Other approaches propose fully convolutional architectures to bring semantic segmentation with symmetric encoders-decoder for point clouds, see for example (Badrinarayanan

et al. 2017). Notably, THOMAS et al. (2019) proposed a convolution kernel well-suited to point clouds. As common kernels require uniform sampling of the point clouds at several scales, a grid-like structure is adopted to this end. The 3D convolutions are made point-wise with 3D spheres at several scales. Similarly to 2D convolutional neural networks, neural features automatically emerge from the network but for 3D shapes, with notable edge detectors, plan detectors, curvature detectors and others. For large scenes, a tiling-like strategy with overlapping and voting schemes is adopted.

The approaches presented in this section process the 3D point cloud directly. Common problems, such as the amount of data to process, the irregularity of densities or the lack of simple neighboring definition have led many authors to structure the point clouds in regular 2D or 3D grids.

2.2.2. Sampling the point clouds on grids

An appealing and straightforward solution consists to first map the point cloud in a regular grid structure. This structure allows to exploit the large amount of available algorithms of image processing/analysis. We named these representations rasters and voxel grids, with related processes named rasterization and voxelization, respectively for 2D grids and 3D grids. In the following, we present the main techniques to structure the point cloud in grids.

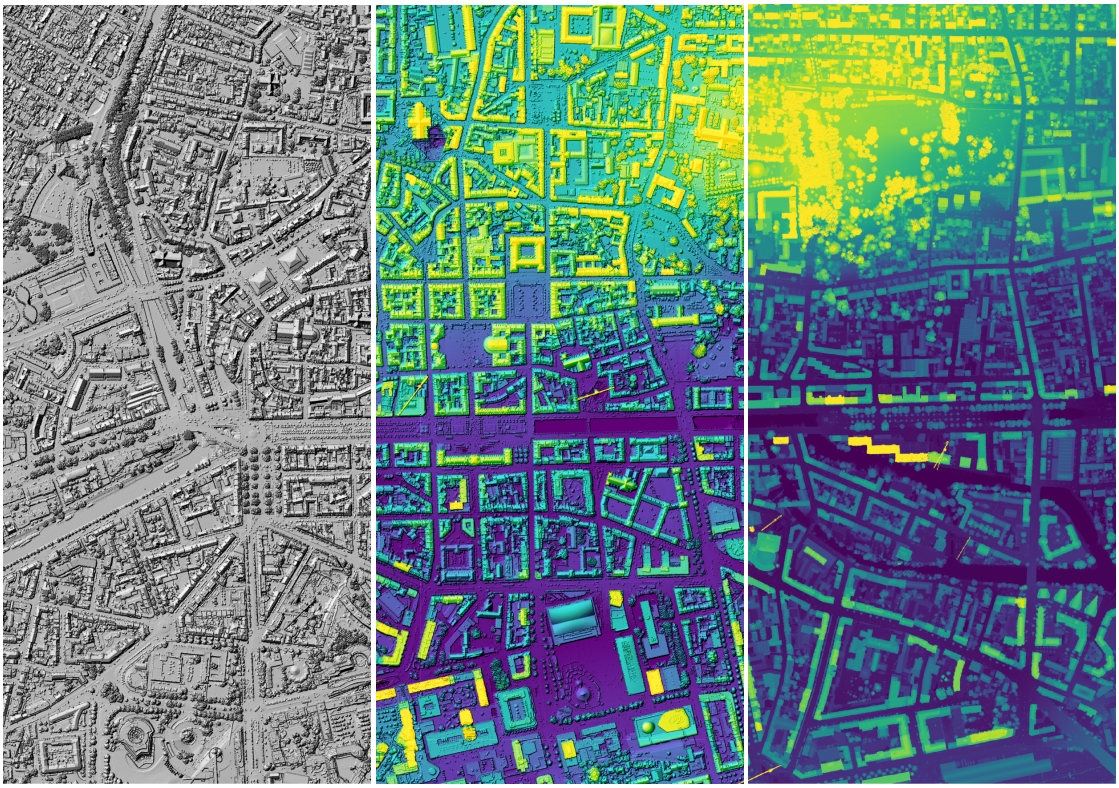
Common rasters derived from LIDAR data

Early on, LIDAR data was simplified to 2D images. The most prominent derived product of LIDAR data is elevation maps, known as Digital Elevation Model (DEM).

Digital Elevation Models A DEM[2]represents the topographic profile of the terrestrial ground surface in a raster format. There are two widespread models: the Digital Surface Model (DSM) and the Digital Terrain Model (DTM).

The DSM includes all the observed structures such as buildings and vegetation. Its computation is straightforward with LIDAR data: for each cell of the raster, only the highest elevation point is kept (see the DSM of Rennes Figure 2.4). The DTM is a model that represents only the bare earth surface, removing in particular buildings and vegetation. As seen previously in Section 2.1.1, LIDAR systems are very good to sample ground points below vegetation and forested cover. This particularity makes

[2]The term DEM is not consistently defined in the literature. We choose to use the term DEM to describe globally any elevation map, it therefore includes the terms Digital Surface Model (DSM) and Digital Terrain Model (DTM).



(a) Hillshades of the DSM (b) Hillshades and elevations (c) DSM elevations

Figure 2.4.: DSM visualizations of the city center of Rennes. The DSM details can be visualized with a virtual light to create lighting and shadows of reliefs (a), the DSM volumes can be visualized with continuous color scales mapping the elevations (c), or composite images can be created with both (b).

LiDAR systems privileged to retrieve DTMs. However, assessing ground points below the canopy or removing large parts of built-up areas is non-trivial, and a significant amount of researches has been conducted on this issue.

Computing DTMs from LiDAR point clouds In 2003, a progressive morphological filter was introduced to derive DTM (Keqi Zhang et al. 2003). The method is based on sliding windows of increasing size, using elevation difference thresholds. In a similar spirit, the Simple Morphological Filter (SMRF) is a filtering approach introduced by Pingel et al. (2013) that uses a filter of linearly increasing window with a slope thresholding. Some other methods are based on the same concept of multiscale windows and elevation functions. One of the drawback of these methods is the need to set several thresholds (for the size of the windows, for the filtering functions) that is often specific to the

dataset. A threshold-free alternative based on skewness was proposed by Bartels and Wei (2010). This method introduces an unsupervised statistical analysis to filter ground points. The algorithm evaluates the skewness (moment of order 3 of the distribution, enabling to analyze the asymmetry in the data) of the point cloud: when the skewness is high, one removes the highest value of the point cloud (assumed to be associated with objects on the ground) while the remaining points, normally distributed, are assumed to belong to the ground. Though efficient, this is not suitable for steep terrains. To cope with this limitation, Mongus and Zalik (2014) introduced the morphological connected operators to filter ground points. The morphologic connected components replace the former morphologic Structuring Elements (SEs) (see Section 2.3 for more in-depth details of mathematical morphology advances). Using the SEs makes indeed difficult to find generic elements adapted to all objects of the scene (they are only suitable for data with specific object sizes) while connected components are more adapted to this issue. The algorithm is extremely efficient, suitable for a variety of object sizes and works fairly well on steep terrains. Another unusual but successful method named Cloth Simulation Filter (CSF) has been proposed by Zhang, Qi, et al. (2016). The method, inspired from computer graphics, simulates a virtual piece of cloth to cover the point cloud turned upside down. The definition of rigidness of the cloth and the simulation of gravity is sufficient to differentiate ground points and non-ground points. For steep terrains, a post-processing method is used. The CSF has few parameters to tune and achieves good accuracy in most terrains.

Other LiDAR derived features Although very marginal compared to DEMs, some authors derived other rasters from the LiDAR data, usually for classification purposes. Lodha et al. (2006) proposed four LiDAR features: normalized height, height variation, normal variation and the LiDAR return intensity. The normalized height, also named Normalized Digital Surface Model (nDSM), is the difference between the DSM and the DTM. The nDSM contains the height of objects relative to the ground. In this work, the height variation is the absolute height difference between the min elevation and the max elevation in a 3×3 pixel window. The derived LiDAR features are used to provide a land cover map using a SVM classification. From multi-spectral LiDAR systems, several intensity maps can be provided (Teo and Wu 2017; Matikainen et al. 2016), and several multi-spectral indices can be computed (Matikainen et al. 2016; Sukhanov et al. 2018), such as Normalized Difference Vegetation Index (NDVI) and Normalized Difference Built-up Index (NDBI). One major advantage to create LiDAR rasters is the straightforward possibility to perform fusion with other data sources, such

as RGB imagery or other multi/hyper-spectral data (Lodha et al. 2006; Teo and Wu 2017; Sukhanov et al. 2018). Let us now turn to the presentation of voxel grids.

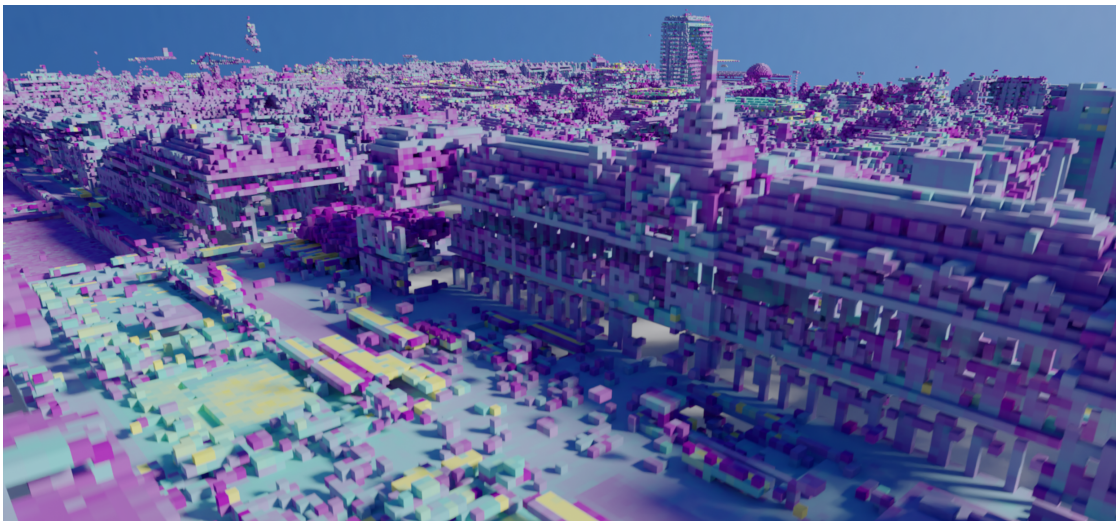


Figure 2.5.: Visualization of a voxelized LiDAR point cloud of the city of Rennes. Points are rearranged in a regular 3D grid, the intensities are averaged in each cell.

Common voxels derived from LiDAR data Voxelization of the LiDAR point cloud is far less common than rasterization. Nevertheless, some specific fields exploit regular 3D grids derived from point clouds. For example, the field of robotics and autonomous driving exploit MLS projected on 3D grids to handle streaming of point clouds in real-time (Hu et al. 2013).

Closer to ALS, several works related to FWF reported using voxelization of the waveforms. Usually a regular voxel grid is defined based on the support vector of the waveform, each sampled intensity being placed in the grid where sums or averaged values are kept in each voxel. For instance, Cao et al. (2016) use voxel-based composite waveforms for the classification of tree species. Alternatively, Wang and Glennie (2015) rely on regular grid structure to facilitate fusion of FWF data with hyperspectral imagery. The regular 3D grid of waveforms data is similar to a hyperspectral cube image.

As many image processing algorithms can be generalized to 3D, including deep convolutional networks, they can be exploited with voxelized LiDAR point clouds. To mention one of them, VoxNet is a convolutional neural network for object recognition based on voxels (Maturana and Scherer 2015). VoxNet uses an occupancy voxel grid as input and a 3D supervised Convolutional Neural Network (CNN) to perform object recogni-

tion. The network analyses occupancy grids of fix resolution size 32×32×32, centered on the object to label.

Despite effective existing techniques, few methods are able to exploit efficiently the multiscale nature of the data. Mathematical morphology is amongst the best frameworks known to date to be able to model complex shape and size structures, and to propose efficient implementations. This in presented in the next section.

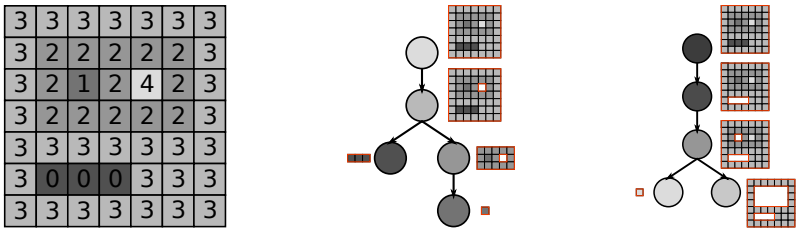
2.3. Mathematical morphology & morphological hierarchies

Mathematical morphology can be summed up as a set of tools to analyze shapes in images.

First morphological operators The seminal morphological studies have been introduced by Matheron (1964) and Haas et al. (1967), several morphological operators based on Structuring Elements (SEs) working on black and white images have been proposed. The well-known operations of erosion, dilation, opening (erosion + dilation) and closing (dilation + erosion) have enabled to process binary images with large efficiency to filter noise, to extract objects of specific shapes, skeletons, among others. Since then, many extensions (including extensions of the operators on grey scales images) have been proposed, we focus in the following on connected and attribute operators.

Connected and attribute operators Connected operators rely on the notion of connectivity. Connectivity defines a connected component as a set of points that may be connected by a path in the image (Salembier and Serra 1995). A local neighborhood is usually defined as the connection between pixels, with 4- or 8-connectivity. Connected operators act by preserving or removing connected components, i.e. not shifting nor blurring the edges of the objects. An efficient connected opening based on attribute operators has been proposed by Breen and Jones (1996). The attribute operators allow to analyze connected components and to filter image regions on the basis of specific attributes rather than using SEs that focuses only on shapes. The main advantage is to preserve specific structures inside images as for instance contours. The opening and closing operations are extended to the class of attribute openings, closings, thickenings and thinnings (Breen and Jones 1996). To process effectively the attribute filtering, antiextensive connected operators have been introduced by Salembier, Oliveras, et al. (1998). The connected operators work on an intermediate structured representation of the image, in the shape of a tree. All these notions are introduced in the following.

2.3.1. Connected components and hierarchical representations



(a) Image (b) Min-tree (c) Max-tree

Figure 2.6.: The min-tree (b) and the max-tree (c) of the image (a). The min-tree has the local minima of the image in leaves while the max-tree has the local maxima in leaves (Illustration from Bosilj, Kijak, et al. 2018).

Max and min trees The structure introduced by Salembier, Oliveras, et al. (1998) is the max-tree[3]. The max-tree allows to use attribute operators efficiently. It consists of hierarchies of nested connected components: the root contains the whole image and the leaves contains the local maxima. The nodes of the max-tree are the nested connected components that are composed of pixels with same grey level h, or with higher grey level than h, in direct vicinity (usually with 4 or 8-connectivity). As a result, the max-tree characterizes the bright structures of the image. The dual of the max-tree is the min-tree, that characterizes the dark structures of the image. Whatever the representation, the original image can be reconstructed from the tree. However, when the tree is pruned before being reconstructed, the result is a thinning or a thickening of the original image, respectively for the max-tree and the min-tree. One interest of the tree hierarchy is to attach various attributes to each node of the tree, that can benefit the structure of the tree to accelerate the processing, then perform attribute filtering on the tree. Filtering the max-tree offers interesting properties, such as the anti-extensivity (or extensivity for the min-tree): the filtered value of a pixel of an image is necessarily less than or equal to the original value of the image. When dealing with increasing attributes (from the leave to the root, e.g. the area attribute), successive filterings with increasing thresholds can result in successive prunings of the max-tree, ensuring the anti-extensivity on successive filterings. Such a property can be used to compute the differential of a filtering and

[3] A close tree representation named component tree was proposed shortly after (Jones 1999). Recent publications imply that the max-tree (and others) are included in the definition of the component trees (Cousty, Najman, and Perret 2013; Bosilj, Kijak, et al. 2018), while others still differentiate the max-tree as a compact representation of the component tree (Souza et al. 2015). In the following, we will use the inclusive meaning, and refer to component trees as the representation of any hierarchical representations.

the original image (or two successive filterings) to visualize the residue, containing the structures removed from the original image (or the previous filtering).

One major advantage of the component trees is that their construction, attribute computation and attribute filtering can be performed in a very efficient way. Numerous building algorithms are available. As an example, Salembier, Oliveras, et al. (1998) used initially a recursive flood-fill algorithm with First-In-First-Out (FIFO) queues. Najman and Couprie (2006) then proposed an algorithm to build components tree in quasi-linear time. Recent optimisations aimed for parallel computation of the max-tree by splitting the image (i.e. using tiles of the full image) to compute local max-tree before a final merge in a single tree (Wilkinson, Gao, et al. 2008; Merciol, Balem, et al. 2017). In addition, modifications of the max-tree have been proposed for distributed tile computing on large computer clusters with message passing (Kazemier et al. 2017; Gotz et al. 2018).

Tree of shapes The tree of shapes proposed by (Monasse and Guichard 2000b) has been designed to provide a unified representation for both bright and dark image structures. The tree of shapes is a compact representation of both the max-tree and the min-tree, characterizing both, the local minima and local maxima. Though this representation looses the anti-extensive property, the tree of shapes is self-dual, meaning that since local maxima and minima are inverted in the dual image, the same tree of shapes is produced.

Originally, the tree of shapes was built by merging a max-tree and a min-tree (Monasse and Guichard 2000a). Since then, more efficient quasilinear algorithms have been proposed (Géraud et al. 2013; Carlinet et al. 2018).

Partition trees The max-tree, min-tree and tree of shapes are inclusion trees. The partition tree is build on other principles: the partition tree represents all the pixels of the image on leaves, contrary to inclusion trees that represent only local minima/maxima in leaves. Partition trees do not use total ordering relations during build time (e.g. relying only on the grey level), making their calculation more complex, but allowing more flexibility in the definition of the connected-components (i.e. structure characterized in the image). The reader is referred to the comprehensive survey on partition and inclusion trees from Bosilj, Kijak, et al. (2018), from which the illustrative Figure 2.7 was taken.

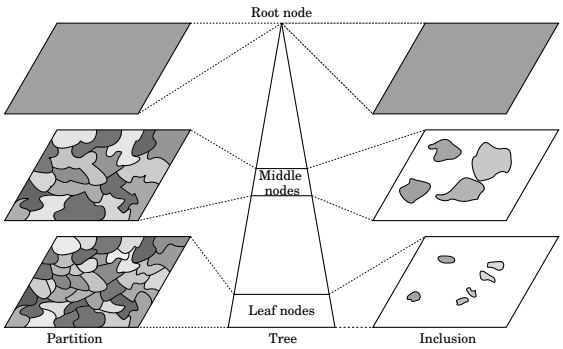


Figure 2.7.: This illustration demonstrates the difference between the partition and inclusion trees. Cuts of the partition tree near its bottom and the middle, as well as the root node are displayed on the left. A set of nodes from the inclusion tree close to the bottom, the middle and the root of the tree are displayed on the right (Illustration from Bosilj, Kijak, et al. 2018).

2.3.2. Mathematical morphology applied to remote sensing

Mathematical morphology has been widely exploited to extract spatial features (Soille and Pesaresi 2002) and to deal with various remote sensing tasks. We illustrate three major contributions in this section: morphological profiles, attribute profiles and pattern spectra.

Morphological profiles

Morphological profiles (MPs) have been introduced for remote sensing by Pesaresi and Benediktsson (2001). The Morphological Profiles (MPs) are able to produce multi-scale description of images, through the use of sequences of morphological reconstruction filterings using SEs of various sizes. The reconstruction filters do not introduce discontinuities and preserve the shapes observed in input images. The MPs are able to model efficiently the contextual spatial and spectral information, significantly improving classification results. However, as the size of images and spatial resolutions continuously increased, the difficulty of calculating the MPs has grown steadily, to the point of discouraging their use on large datasets. As a matter of fact, the structures characterized MPs are dependent of the size and orientation of the SEs, for example: object size can be discriminated by SEs of various size and object elongation is discriminated by multiple linear SEs oriented in all directions. The combination of sizes, shapes and orientations of SEs often induce polynomial computation times.

Attribute profiles

Dalla Mura, Benediktsson, Waske, et al. (2010) introduced the Attribute Profiles (APs) as the extension of the MPs. They propose a more generic framework based on the morphological attribute operators. The Attribute Profiles (APs) are capable of removing entire regions of the image depending on any defined attributes, such as region shape or region size. Similarly to MPs, they provide accurate multiscale spectral and spatial descriptions, according to a set of thresholds applied on an attribute. By using the max-tree and the min-tree to process the attribute filtering, the algorithms are usually quasilinear. For this reasons, APs and their extensions has become widespread in remote sensing.

To date, a large number of extensions of APs have been proposed. Self-Dual Attribute Profiles (SDAPs) have been introduced shortly after by Dalla Mura, Benediktsson, and Bruzzone (2011) and are based on a single tree of shapes rather than both min and max-tree. The tree of shapes allows to filter only one tree (instead of two), and the vector is therefore twice more compact. They have been proven to be more efficient than APs for classification accuracies and processing time. The Histogram-based Attribute Profiles (HAPs) from Demir and Bruzzone (2016) are local histograms computed on the description provided by APs, to capture local texture information (useful for very high resolution imagery). As Histogram-based Attribute Profile (HAP) induces a very high dimensionality and are sensible to the number of histogram bins, Pham, Lefevre, et al. (2018) proposed the Local Feature-Based Attribute Profiles (LFAPs) that are based on the same local texture description but describe only the local mean and variance, achieving better results with more compact description. The Feature Profiles (FPs) from Pham, Aptoula, et al. (2018) proposes to return attributes values in the profiles instead of the level of the connected component. Other works focused on the thresholds selection, usually empirically defined. Notably, Ghamisi, Souza, et al. (2016) introduced the Extinction Profiles (EPs). The Extinction Profiles (EPs) replace the attribute filters by extinction filters. The extinction filters act by preserving or removing completely the regional extrema of the image. The filtering parameter is the number of extrema to be preserved instead of a threshold value, making it less sensitive to image resolution. Let us also note that Cavallaro, Falco, et al. (2017) propose automatic thresholds selection for APs on the basis of granulometry, and Das et al. (2020) propose automatic thresholds selection by considering the whole distribution of an attribute values. Finally, thresholds free APs were proposed from Bhardwaj et al. (2019): the idea is to create path from leaves to root and to define a leaf attribute

function to detect the major change along the path. Some components are automatically removed on the basis of detected major connected component. For more details about APs, their extensions and some experimental comparison, the reader is referred to the survey given in Appendix B.

Pattern spectra

Pattern spectra are another major tool for image analysis and classification offered by mathematical morphology. Pattern spectra, introduced by (Maragos 1989), are multiscale shape (or size) descriptors for an image. They were originally based on openings and closings with SEs to obtain probabilities distribution of shape granulometry (or size granulometry). By combining the shape and size pattern spectra, Urbach, Roerdink, et al. (2007) obtained 2D shape-size pattern spectra whose distribution probabilities can be binned in a 2D shape-size histogram that is representative of the image content. Moreover, Urbach, Roerdink, et al. (2007) proposed to use the max-tree to compute efficiently shape and size attributes. Pattern spectra have been successfully adapted for retrieval of remote sensing images (Bosilj, Aptoula, et al. 2016).

2.4. LiDAR data processed with mathematical morphology

Mathematical morphology operators on point clouds The essential neighbouring information needed for morphological operators is missing from raw point cloud. However, a couple of works have addressed mathematical morphology directly on point cloud space. Calderon and Boubekeur (2014) proposed dilation and erosion operators of point clouds, under the condition that the point clouds represent a known underlying surface. This method was aimed for computer graphics models and is hardly applicable to irregular and complex LiDAR point clouds. Recently, Asplund et al. (2019) proposed a new approach suitable on irregular 3D point clouds. The structuring elements are points sampled in 3D to perform erosion, opening and closing in the 3D point cloud space. They use this approach to segment facades and grounds on MLS urban scenes.

Mathematical morphology on discretized LiDAR features From another perspective, discretized LiDAR data benefit the neighboring information needed to perform erosion/dilation or to construct connected components. The most popular and successful application of mathematical morphology on LiDAR data is undoubtedly ground filtering (e.g. from Keqi Zhang et al. 2003; Pingel et al. 2013; Mongus and Zalik 2014). Please refer to Section 2.2.2 for more details on these methods.

Few other works using morphologic operators to segment DEMs have been proposed. For example, Mongus, Lukač, et al. (2014) used the MPs to segment the ground and buildings and Serna and Marcotegui (2014) used area openings and other mathematical morphology operators on DSM to detect ground and objects, then segment and classify objects (that are reprojected to the original 3D point cloud to derive a 3D result).

Considering 3D voxel grids, Serna, Marcotegui, and Hernández (2016) used an "adaptive" voxelization strategy on MLS data. The voxel are processed on a regular 3D grid vertically aligned on the DTM. Their value is binary and indicate whether there are LiDAR points or not. Binary connected components are computed on the voxels, along with elongation attributes. Finally they proceed to attribute filtering to segment facades.

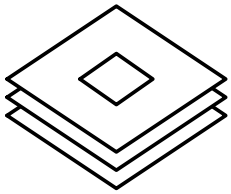
The popular APs have often been used on single DSM generated from LiDAR point clouds, especially in combination with external sources of spectral data (Damodaran et al. 2017; Pedergnana et al. 2012; Liao et al. 2017; Zhang, Ghamisi, et al. 2017; Kwan et al. 2020), neglecting the information potential contained in LiDAR data. Nevertheless, few works have been conducted by extracting several LiDAR features, using APs (Wang, He, et al. 2018) or EPs (Ghamisi and Hofle 2017).

Conclusion

This chapter has introduced the main principles of LiDAR systems and the main approaches including machine learning and deep learning developed to process such data. Among efficient methods, some directly process the 3D point clouds but most of them rather prefer to structure the data in 2D or 3D grids and to use more conventional image/volume processing tools. Among these tools, surprisingly few studies exploit morphological representations, despite efficient results and despite the fact that most ALS embed multiscale structures particularly adapted to morphological approaches. Therefore in this thesis, we suggest to rely on mathematical morphology on specific rasterized or voxelized point clouds. The next chapter presents our first contribution on rasterization strategies for LiDAR point clouds using attribute profiles.

Part II.

2D Rasterization



RASTERIZATION STRATEGIES FOR AIRBORNE LIDAR CLASSIFICATION USING ATTRIBUTE PROFILES

DOI 10.1109/JURSE.2019.8808945

Authors Florent Guiotte, Sébastien Lefèvre and Thomas Corpetti.

Keywords airborne LiDAR, land cover mapping, attribute profiles, multilevel image description, rasterization.

Objectives *We offer a simple method to classify LiDAR data.*
We first simplify the complex 3D LiDAR point cloud into several 2D feature maps. We then use hierarchical representations to create spatial and spectral descriptors. Finally we train a supervised model to predict urban classes on the Houston dataset. We evaluate the LiDAR features individually and we assess the value of hierarchical representations.

Results The more features extracted from LiDAR data, the better the classification. The multi-scale features further improve the classification, they are especially effective on DEMs features.
The method is fast and reliable to provide land cover maps over urban areas.

Contents

Abstract This paper evaluates rasterization strategies and the benefit of hierarchical representations, in particular attribute profiles, to classify urban scenes issued from multispectral LiDAR acquisitions. In recent years it has been found that rasterized LiDAR provides a reliable source of information on its own or for fusion with multispectral/hyperspectral imagery. However previous works using attribute profiles on LiDAR rely on elevation data only. Our approach focuses on several LiDAR features rasterized with multilevel description to produce precise land cover maps over urban areas. Our experimental results obtained with LiDAR data from university of Houston indicate good classification results for alternative rasters and even more when multilevel image descriptions are used.

3.1. Introduction

Airborne LiDAR systems are a common source of aquisition for elevation data. Such systems provide accurate 3D point clouds of the scanned scenery. LiDAR is very popular over urban areas where it brings a valuable complementary source of information when used with multispectral or hyperspectral optical data in order to achieve land cover or land use mapping.

LiDAR data are voluminous, irregularly distributed point clouds coming along with intensity features and acquisition meta-data. Due to this complexity, LiDAR data for land cover mapping are often simplified to a DEM used as additional information for fusion with multispectral or hyperspectral images. In this work, we focused on providing such maps with LiDAR data only using multilevel image description. Classification of several urban classes was derived based on features from LiDAR data such as intensities, elevation and number of echoes.

In the following we review simple yet effective rasterization strategies of LiDAR data, that are subsequently used for multilevel image description with APs. By doing so, we are then able to efficiently derive a precise land cover map through supervised classification.

3.2. Related work

3.2.1. Classification of LiDAR data

Numerous methods have already been proposed in the past decade for LiDAR point cloud classification, coming from various scientific fields such as geosciences (flow, erosion, rock deformations, ...), computer graphics (3D reconstruction) or Earth observation (detection of trees, roads, buildings, ...).

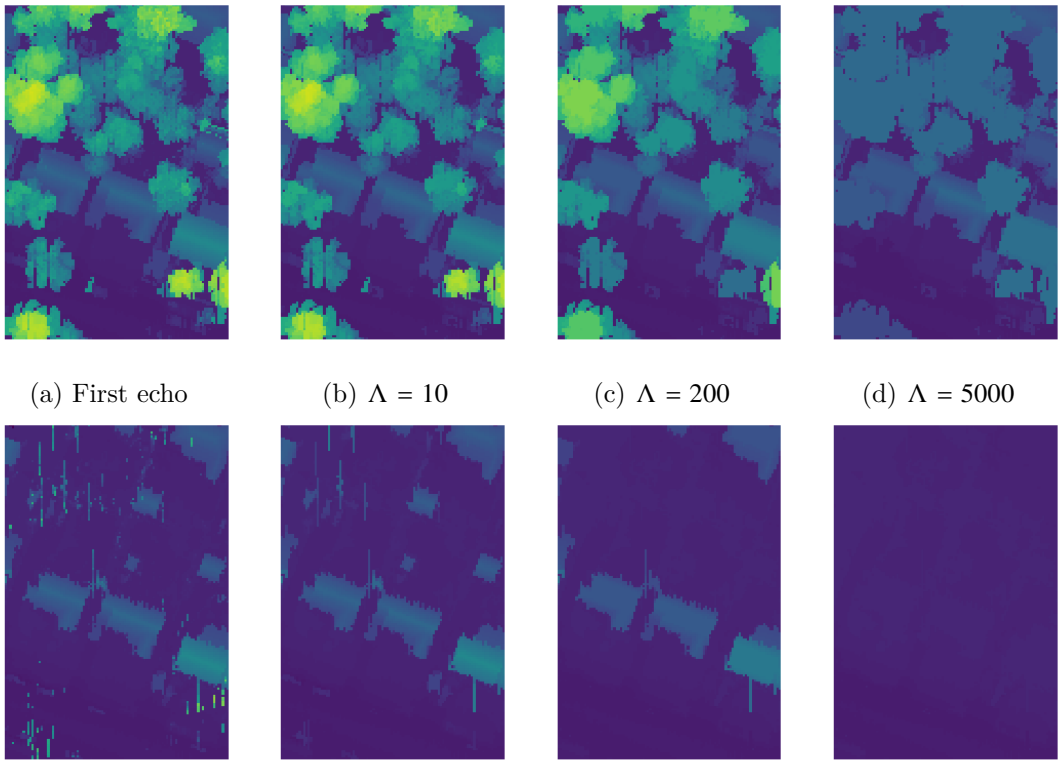
Among efficient techniques, some directly exploit the 3D point cloud structure (Brodu and Lague 2012; Niemeyer et al. 2014; Mallet, Bretar, et al. 2011) while in many applications the point cloud is first binned into a 2D regular grid (rasterization process) on which computer vision approaches can be applied (see e.g. Lodha et al. 2006). Apart from some specific applications where LiDAR points are fused with other data (e.g. hyperspectral images Damodaran et al. 2017; Pedergnana et al. 2012; Ghamisi, Benediktsson, et al. 2015; Khodadadzadeh et al. 2015), most techniques consist in computing features to describe the point clouds, before using such features to classify the scene under study. While first works have been focused on the characterization of single points (often through height and intensity) without including information related to their neighbours (Lodha et al. 2006), more advanced approaches have included spatial

relationships using a set of spheres or cylinders (of variable radius) around each point to extract consistent geometric features (Mallet, Bretar, et al. 2011; Weinmann et al. 2015; Niemeyer et al. 2014). In this context, multiscale local 3D features (main orientation, variability around each point, …) have proven their efficiency to classify LiDAR scenes (see e.g. Brodu and Lague 2012). Even if it is very efficient, the sphere used to assess the neighbourhood of points is isotropic (no orientation is promoted) which is not optimal since the geometry of objects is not taken into account. Therefore other multi-scale approaches have been proposed on LiDAR DEM, such as the popular attribute profiles (Dalla Mura, Benediktsson, Waske, et al. 2010) that produce a multiscale description of the pixel and its surrounding (Ghamisi, Benediktsson, et al. 2015; Khodadadzadeh et al. 2015; Pedergnana et al. 2012) before proceeding to the classification. The main idea behind is to compute multi-scale spatial features by taking into account the geometry of the scene. In this work, we suggest to explore various information derived from the LiDAR point cloud within the framework of attribute profiles.

3.2.2. Attributes profiles

Introduced in 2010 for remotely-sensed images, morphological APs (Dalla Mura, Benediktsson, Waske, et al. 2010) enable a multi-scale description of data driven by their spatial and spectral information. Efficient computation of APs is achieved through tree-based representation of the gray level sets with either a max or a min-tree. They have then been superseded by Self-Dual Attribute Profiles (SDAPs) (Dalla Mura, Benediktsson, and Bruzzone 2011) built from a unique multi-scale representation of an image through the tree of shapes. In this tree, all nodes represent nested connected regions of similar pixels, with leaves made of the local extrema and the root gathering all pixels of the image. Then, successive filterings of the image (or equivalently the tree representation) are performed according to some predefined characteristics computed for each node such as area, moment of inertia, or standard deviation of the connected components. The filtered images are finally stacked together to form description vectors called SDAPs. The concatenation of the SDAPs from different bands in a single vector are called Extended Self-Dual Attribute Profiles (ESDAPs) (Cavallaro, Dalla Mura, et al. 2015). From the concept of SDAPs, the Derivative of Self-Dual Attribute Profiles (DSDAPs) contains the same information but expresses the difference between successive levels of the SDAPs.

Previous works combining APs and LiDAR data only focus on DEM (cf. Sec. 3.2.1).



(a) First echo (b) $\Lambda = 10$ (c) $\Lambda = 200$ (d) $\Lambda = 5000$

(e) Last echo (f) $\Lambda = 10$ (g) $\Lambda = 200$ (h) $\Lambda = 5000$

Figure 3.1.: Some rasters over residential area, for first echo (a-b-c-d) and last echo (e-f-g-h) with original rasters (a-e), SDAP filtering with $\Lambda = 10$ (b-f), $\Lambda = 200$ (c-g) and $\Lambda = 5000$ (d-h). One can observe the interest of the last echo (second line) since structures below vegetation are clearly highlighted.

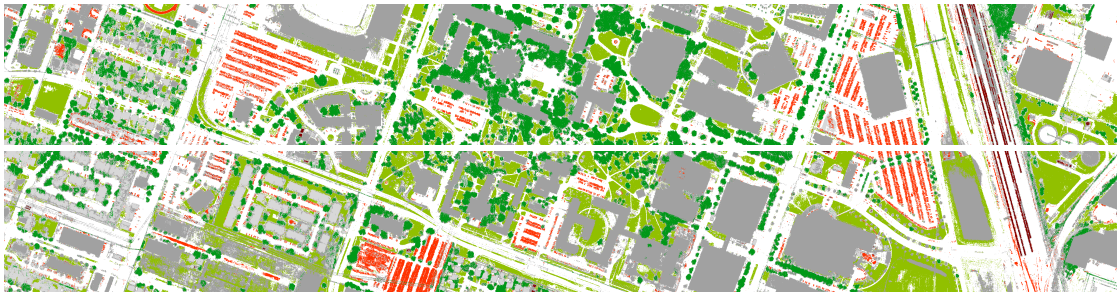


Figure 3.2.: Classification of the scene using the DSDAPs with all features $\{N, I, I_r, D, D_r\}$. The 7 classes are represented as follows: roads in white, grass in green, trees in dark green, residential buildings in light grey, non-residential buildings in medium grey, cars in red and trains in purple.

Yet, in recent LiDAR acquisition systems, multi-spectral information can be extracted and the question of the description of other features than DEM is open.

In this study we aim to enhance existing LiDAR classification methods using SDAPs and their derivative to better describe several features extracted from LiDAR data. Validation is performed on the IEEE DFC 2018 public dataset.

3.3. Rasterization strategies

In this section we provide the features we extracted from LiDAR data and the multi-scale filtering we have chosen for the spatial description.

3.3.1. LiDAR features

LiDAR systems are usually exploited to provide unstructured 3D point clouds used to derive a DEM. Though a DEM brings useful information, additional features issued from LiDAR can still be exploited, in particular:

1. *The spectral intensity associated with the first echo in each spectral band.*

2. *The number of echoes in each spectral band.* Some structures, especially in vegetated areas, do not fully backscatter the laser pulse and yield in multi-echoes signals.

3. *The position of the last echo.* For multi-echoes backscattered signals, this enables to localize the last element encountered. In some situations (especially with trees), this enables to localize the ground surface.

4. *The associated intensities of last echo.* Analogously to the paired information of first echo position and intensities we have used the last echo spectral intensities in addition to its position.

The two latter types of information are usually less employed. Nevertheless, they can help the classification process since all wooded areas are removed, as illustrated by Fig. 3.1a(e).

The rasterization process aims to provide in each cell some representative values of the aforementioned features. For the sake of simplicity, we have chosen to average the values of intensity, elevation and number of echoes contained in each cell. Furthermore, we fill potential empty cells (missing data) through linear interpolation.

3.3.2. Attribute filtering

As previously indicated, we consider here the application of attribute profiles over different rasterized versions of LiDAR data. In this paper, for the sake of simplicity, we have chosen to filter all LiDAR features solely based on the area attribute whose values have been set in the urban context. More precisely, we consider three main scales (in other contexts, automatic approaches can be used (Cavallaro, Falco, et al. 2017)):

- small values (1 to 5 m^2) remove small-sized objects (e.g. power lines) and can be regarded as denoising filters;
- moderate values (5 to 50 m^2) remove medium-sized objects (cars, trees, ...);
- large values (more than 50 m^2) remove larger objects (e.g. buildings) and therefore enable to automatically derive a DTM.

3.4. Experiments

3.4.1. Dataset and setup

Dataset

Our method was tested on the multi-spectral LiDAR acquisition of the University of Houston issued from 2018 IEEE GRSS Data Fusion Contest dataset[1]. The associated ground truth map has a spatial resolution of $0.5m$.

[1] cf http://www.grss-ieee.org/community/technical-committees/data-fusion/2018-ieee-grss-data-fusion-contest/

Classes

We choose generic urban classes (roads, grass, trees, residential buildings, non-residential buildings, cars and trains) of the dataset to evaluate the overall accuracy.

Classifier

We choose random forest (RF) to classify our data since such method provides reliable results with respect both to accuracy and computational efficiency (Liaw, Wiener, et al. 2002).

Feature computation

ESDAPs have been created by filtering each raster with area attributes. According to observations made in Sec. 3.3.2, we have chosen thresholds $\Lambda = \{10, 200, 5000\}$ to compute attributes. In practice, all features mentioned in previous sections are tested independently and combined together. These features are DSM (denoted as D), intensities (denoted as I), intensities of last echo (denoted as I_r), number of backscattered echoes (denoted as N) and position of the last echo (denoted as D_r) . For the sake of comparison, we also ran the classification only with initial rasters (without multiscale analysis through SDAPs) to evaluate the benefits of APs and variants. The ESDAPs can be formulated as:

$$ESDAP = \{SDAP(D), SDAP(I), ..., SDAP(N)\}$$

Train and test data

Unlike most common multi-scale features that rely on spatial windows, attribute profiles rely on specific connectivity that prevents the use of random points to train and test a classifier. As a matter of fact, two pixels in various spatial areas of the image but with similar characteristics are likely to share common nodes, and hence common features. Therefore a random choice of train and test points is unfair. To cope with such a bias, one solution is to spatially split the dataset in two images of same size where the first one is used to train the classifier and the other to test it, and conversely. In practice, we split horizontally the dataset in order to maximize the class distribution in each sub-image.

Validation criteria

From the training split, 10% of the points have been randomly selected to train and evaluate our approach. The process has been repeated 100 times and we provide averaged

overall accuracy (OA) and Cohen's kappa coefficient (κ) for each experience.

3.4.2. Results

Quantitative evaluations are depicted in Tab. 3.1. As expected and reported in previous works (Ghamisi, Benediktsson, et al. 2015; Pedergnana et al. 2012; Damodaran et al. 2017), SDAPs improves the classification when used with DEM D only (first line). This observation is almost valid for all other features. Performances of DSDAPs are slighlty higher than SDAPs except for DEM.

On this dataset, it is surprising to note that multi-spectral intensities (first echo I or last echo I_r) are performing better than DSM D (lines $5, 3, 1$ respectively). This information is likely to be an interesting feature to separate urban elements.

It is also worth noting that regarding positions, the last echo D_r feature is more useful than first echo D (lines $1, 4$) when combined with hierarchical features, with a significant improvement (about $\sim 10\%$). The ability of the last echo to assess buildings can explain such an observation.

Finally, as expected, the combination of all features (last lines) enables to achieve the best accuracy. Though the difference between DSDAPs and SDAPs is limited, the improvement with respect to the baseline is important and this demonstrates the ability of APs-based features to properly classify LiDAR data in urban environments. We can assess an improvement of $\kappa = 0.06$ while comparing pixelwise and DSDAPs description both with all features. We also can assess an improvement of $\kappa = 0.30$ while comparing SDAPs on DSM only with SDAPs on all LiDAR features.

For the sake of illustration, we provide in Fig. 3.2 the classification obtained using all features with DDSDAPs, line 14 of Tab. 3.1. Let us remind that in practice, the complete image has been horizontally split to avoid common features issued from APs between train and test data. Therefore, as some classes appear in few parts of the image, they have not been learned and hence, not been properly classified (for instance the grassy slope at the bottom right has been classified as a building). Despite this difficulty, both quantitative evaluations and qualitative maps are interesting. The prediction map is indeed consistent as the objects exhibit few noise and proper borders. Even if some irregularities on the edges of some objects (e.g. buildings) appear, they are mainly due to the irregularity of the initial point cloud sampling.

	Description	Pixel			E · SDAPs			E · DSDAPs		
	Evaluation metric	OA(%)	κ(×100)	d	OA(%)	κ(×100)	d	OA(%)	κ(×100)	d
1	D	64.79	46.52	1	72.67	58.80	4	69.46	54.77	4
2	N	52.30	8.43	3	52.27	10.01	12	52.74	10.64	12
3	I_r	67.82	51.06	3	70.42	54.35	12	72.54	57.84	12
4	D_r	63.91	45.12	1	73.94	59.73	4	72.76	58.29	4
5	I	68.36	52.18	3	72.00	57.30	12	73.28	59.25	12
6	$\{D, D_r\}$	69.65	54.19	2	77.22	64.85	8	77.24	65.36	8
7	$\{N, D_r\}$	68.64	52.56	4	78.88	67.44	16	78.45	67.05	16
8	$\{I, D\}$	83.97	75.64	4	86.29	78.90	16	87.39	80.71	16
9	$\{I, D_r\}$	82.26	72.84	4	86.97	79.86	16	87.93	81.46	16
10	$\{N, I, D\}$	85.99	78.64	7	87.45	80.68	28	88.69	82.69	28
11	$\{N, I, D_r\}$	84.80	76.70	7	88.29	81.91	28	88.99	83.09	28
12	$\{I, D, D_r\}$	84.59	76.50	5	88.21	81.80	20	89.28	83.56	20
13	$\{N, I, D, D_r\}$	85.87	78.46	8	88.92	82.90	32	90.04	84.74	32
14	$\{N, I, I_r, D, D_r\}$	**86.20**	**78.93**	11	**89.05**	**83.09**	44	**90.16**	**84.91**	44

Table 3.1.: Overall accuracy (OA) and Cohen's Kappa coefficient (κ) for each feature and combination of them using pixels values, SDAPs, DSDAPs (lines 1–4) and their extended versions (lines 5–14). The dimension of the feature vector (d) is also indicated (intensities (I, I_r) and number of echos N are taken in each of the three spectral bands).

3.5. Conclusion

In this study, we have addressed the classification of multi-spectral LiDAR using rasterized features and attribute filtering. Results showed that combination of different rasterized strategies can improve classification with the sole use of LiDAR data. In addition, considering SDAPs to model multiscale spatial organization futher improved our results to the point it allowed us to produce a precise land cover map over urban area. Futhermore, the proposed method is fast (for instance, the map from Fig. 3.2 (1202x4768 pixels) required only 2 seconds for DSDAPs description and 50 seconds for RF classification, considering a laptop CPU (i7-7600U CPU @ 2.80GHz, 4 threads) and it can be used with any supervised classifier (see Merciol, Balem, et al. 2017). Additional improvements can be designed for this method in the future. On the one hand, we can extract many more features from LiDAR such as point density, orientation within a cell, ratio between spectral bands and summarize the cell with other metrics than mean value such as standard deviation or quantiles. On the other hand, we can enhance SDAPs by filtering more attributes such as moment of inertia, by using more advanced APs-based methods such as the Local Feature-Based Attribute Profiles (LFAPs) (Pham, Lefevre, et al. 2018) or even by creating LiDAR specific attributes to be included during the construction of the tree. Regarding classification of overlapping classes (e.g. buildings beneath trees), it could also be interesting to head for 3D classification. With this in mind, 3D ground-truth would be optimal for LiDAR data.

Acknowledgment

The authors thank the National Center for Airborne Laser Mapping and the Hyperspectral Image Analysis Laboratory in University of Houston for data used in this study, and the IEEE GRSS Image Analysis and Data Fusion Technical Committee.

CHAPTER 4.

INTERACTIVE DIGITAL TERRAIN MODEL ANALYSIS IN ATTRIBUTE SPACE

DOI 10.5194/isprs-archives-XLIII-B2-2020-1203-2020
Authors Florent Guiotte, Geoffroy Etaix, Sébastien Lefèvre and Thomas Corpetti
Keywords Pattern Spectra, LiDAR, DTM, mathematical morphology

Objectives *The goal of this work is to characterize structures present in a DTM.*
The structures of the DTM are displayed in a spectrum of attributes (size and shapes). The computation of attributes and the filtering of the DTM is performed with the component trees.

Results The use of the component tree allows fast attribute processing. The user can interactively select structures of interest on the DTM or in the spectrum, corresponding areas are highlighted in the spectrum or in the DTM. The tree structure allows real-time filtering, thus, providing an interactive application. A use case of the interactive application is available Appendix A.
Quantitative result shows good potential of this method to detect gold panning areas in a DTM.

Contents

Abstract The use of high-resolution digital terrain model derived from airborne LiDAR system becomes more and more prevalent. Effective multi-scale structure characterization is of crucial importance for various domains such as geosciences, archaeology and Earth observation. This paper deals with structure detection in large datasets with little or no prior knowledge. In a recent work, we have demonstrated the relevance of hierarchical representations to enhance the description of digital elevation models (Guiotte, Lefèvre, et al. 2019c). In this paper, we proceed further and use the pattern spectrum, a multi-scale tool originating from mathematical morphology, further enhanced by hierarchical representations. The pattern spectra allow to globally and efficiently compute the distribution of size and shapes of the objects contained in a digital elevation model. The tree-based pattern spectra used in this paper allowed us to analyse and extract features of interest. We report experiments in a natural environment with two use cases, related to gold panning and dikes respectively. The process is fast enough to allow interactive analysis.

4.1. Introduction

Data analysis is usually achieved through a representation of the data in an appropriate feature space. In the context of digital images, the histogram of graylevels has been used for decades as a simple and efficient probability density function to characterize image contents. While today feature learning achieved with deep neural networks has shown great success including for remote sensing data, it still requires the availability of massive amounts of data, either with or without annotations (supervised and unsupervised learning, respectively). In the context of interactive data analysis, efficient methods that do not impose such requirements remain appealing.

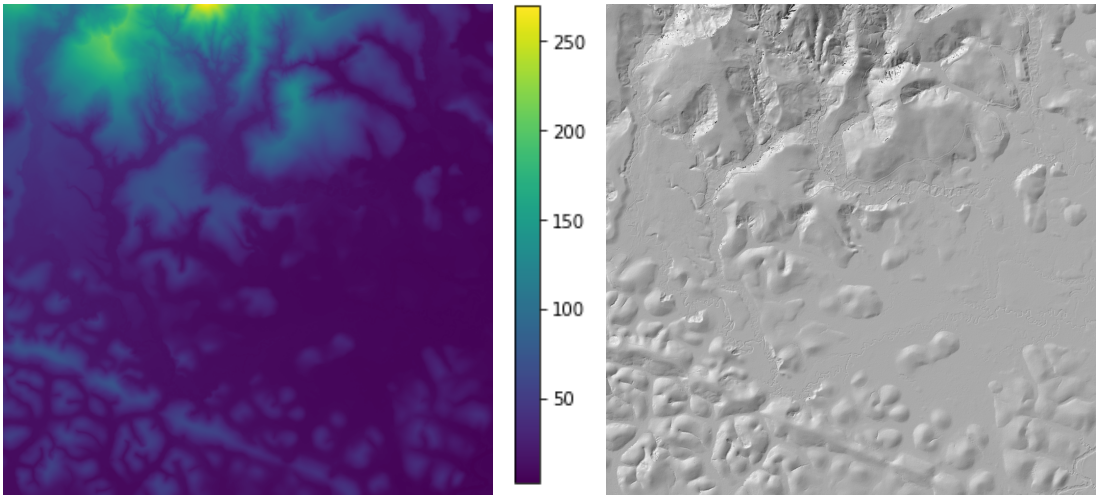
Among existing methods for representing an image content in a predefined feature space, the pattern spectra (Maragos 1989) have established as an advanced solution to describe the image through a probability distribution function of some attributes measured on the image parts. More precisely, and conversely to the histogram of graylevels, the analysis is not conducted at the pixel level but rather on all components or regions present in the image. The underlying scale-space is efficiently built using either level sets or multiscale segmentation, through inclusion or partitioning trees respectively (Bosilj, Kijak, et al. 2018). As far as the attributes are concerned, pattern spectra give access to a wide range of properties beyond the distribution of graylevels, e.g. related to the size or shape of images components. Let us note that the attributes can be combined to provide a multidimensional attribute space in which a further analysis can be conducted. For instance, area, non-compactness and Shannon entropy have been successfully used for aerial image retrieval (Bosilj, Aptoula, et al. 2016). Interactive filtering of satellite images has been explored in (Ouzounis and Soille 2011; Gueguen and Ouzounis 2012).

However, to the best of our knowledge, pattern spectra have never been used in the context of DTM data analysis yet.

4.2. From DTM to Attribute Space

In this paper, we will use an attribute space to characterize structures present in a DTM.

Direct visual interpretation of DTM is feasible for medium and large structures, but it can be difficult especially in mountainous areas where a high range of vertical information is present (cf. Figure 4.1a). While hillshades appear as an alternative way for visual interpretation that is especially efficient for micro-reliefs, it remains difficult to perceive the scale and the depth of objects through hillshades (Figure 4.1b). We claim that



(a) DTM (altitude in m) (b) Hillshades of the DTM

Figure 4.1.: Visualizations of a DTM below vegetation in a mountainous area generated from multi-echo LiDAR system.

pattern spectra can be used as a relevant alternative, since they provide an automatic tool to deal with the high range of vertical values while preserving micro-reliefs.

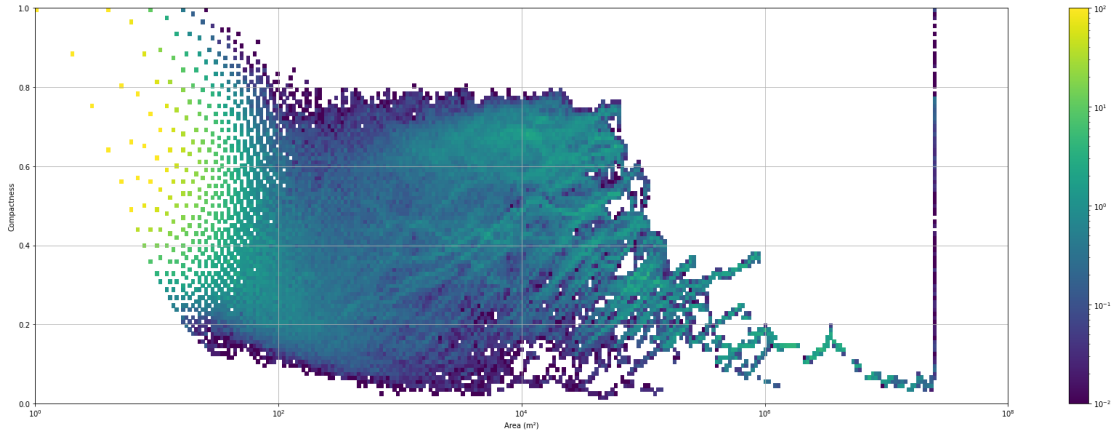


Figure 4.2.: Pattern spectrum (area and compactness) of the DTM with the max-tree.

4.2.1. Morphological Hierarchy

An interesting mean to represent the hierarchy in an image is to use morphological operators. Among them, min-trees, max-trees and pattern spectra (Salembier, Oliveras, et al. 1998) provide reliable models and descriptors, and will be used in this paper. Min-

and max-trees form a hierarchical decomposition of an image X from a domain $E \subset \mathbb{R}^2$ with $E \to \mathbb{Z}$ or $\mathbb{R}$ based on level sets of flat zones and are briefly described below.

Max-tree is composed of nodes, a set of flat zones N_h^k linked together in relation to their level h. The flat zones are the peak components $P_h^k(X)$ valued with the level h. Peak components are determined according to thresholds $\mathcal{T}_h(X)$ over the points x of the image X such that:

$$\mathcal{T}_h(X) = \{x \in E | X(x) \geq h\} \tag{4.1}$$

k is an index over the set $\mathcal{T}_h$. The root node of the tree includes the whole image X with the lowest level h of X, while the leaves contain local maxima. All flat zones are nested for decreasing values of h and this results in the so-called *max-tree*.

Min-tree is constructed conversely using lower level sets. Because of duality, the min-tree hierarchy can also be constructed as a max-tree of the inverted image $(-X)$.

Tree of shapes introduced by (Monasse and Guichard 2000a), describe the image in a self-dual way, similar to the merging of min- and max-tree. The tree characterizes both local maxima and local minima.

A full image can be reconstructed directly from its min-, max-trees and tree of shapes. Efficient techniques exist to construct such trees at a low computational cost. In this paper, the elevations of the DTM are viewed as graylevels to construct min-, max-trees and tree of shapes (note that other trees can be used, including fine-grained partition trees such as the α-tree).

These morphological hierarchies capture the structures inside the DTM. To do so, we rely in this paper on the characterization functions known as pattern spectra.

4.2.2. Pattern Spectra

For each node N_h^k of a tree, many criteria (a.k.a. *attributes*) related to the properties of the peak components (e.g. shape or size, see next section for a discussion of available attributes) can be computed. A $1D$ pattern spectrum can be viewed as the probability density function related to the probability that a component with a given attribute is present in the image. In a similar way, a $2D$ pattern spectrum can be viewed, for a whole image, as the joint probability density function of peak components with two attributes. A direct link between a pair of attribute values and the corresponding areas in the image can be established and depending of the chosen attributes, the different bins of pattern spectra can highlight very meaningful areas.

The pattern spectra can be seen as a histogram describing the distribution of properties (e.g. sizes or shapes) present in the tree. The size and shape attributes are first split into ranges of size or shape classes. We used two rules to create such classes:

Linear partitioning that consists in a straightforward range partitioning between the minimum and the maximum of the attribute with a fixed step. We use this for attributes with a normal distribution (e.g. compactness).

Geometric partitioning where a logarithmic range partitioning appears suitable to describe wide distributions while keeping the ability to characterize small values (e.g. area, height, volume).

In this paper, $2D$ pattern spectra will be used with specific attributes to highlight key information in DTM.

The construction of the 2D pattern spectra S is as follow:

1. Choose the size $i \times j$ of the spectrum.
2. Choose two attributes (e.g. area A and compactness C).
3. Define the classes of the two attributes according to the size i and j and the previous partitioning rules.

Then for each node N_h^k of the tree:

1. Compute the classes c_i and c_j of the two attributes.
2. Compute height difference $\delta_h = h - h_p$ between the node N_h^k and its parent $N_{h_p}^{k_p}$.
3. Increment the cell $S[c_i, c_j]$ with the area times the height difference $A(N_h^k) \times \delta_h$.

4.3. A DTM Perspective on Attributes

From the chosen tree, we recursively compute attributes of the nodes. We consider here different attributes:

area: $A(N_h^k)$ the surface area of the node (i.e. the pixel count in the node).

perimeter: $P(N_h^k)$ the perimeter of the node (i.e. the pixel count of the node contour).

compactness: $C(N_h^k)$ degree of compactness of the shape of the node, defined as $C(N_h^k) = 16A(N_h^k)/P(N_h^k)^2$. Compactness ranges from 1 for compact shapes (e.g. circles) to 0 for non-compact shapes.

height: $H(N_h^k)$ difference between the elevation of the parent of the node $N_{h_p}^{k_p}$ and the elevation of the deepest node in the subtree rooted in the node.

volume: $V(N_h^k)$ of the node is the area times the elevation difference with its parent (so $A(N_h^k) \times \delta_h$), plus the sum of the volumes of all children of the node.

mean altitude: $M(N_h^k)$ of the pixels contained in the node.

altitude dynamics: $D(N_h^k)$ of the node, the difference between the altitude of the deepest minima of his children and the altitude of his closest ancestor that has one of his children with deeper minima.

One major advantage of using hierarchical representations on DTM is the direct link between the altitude of the node in the tree and the altitude of the objects in the DTM. As a result, in addition to the conventional area attribute that can be formulated in square meters, the height and volume attributes can be translated with physical meaning (i.e. respectively in meters and cubic meters).

Depending on the application, we choose one or several attributes to characterize the structures in the DTM.

4.4. Experiments

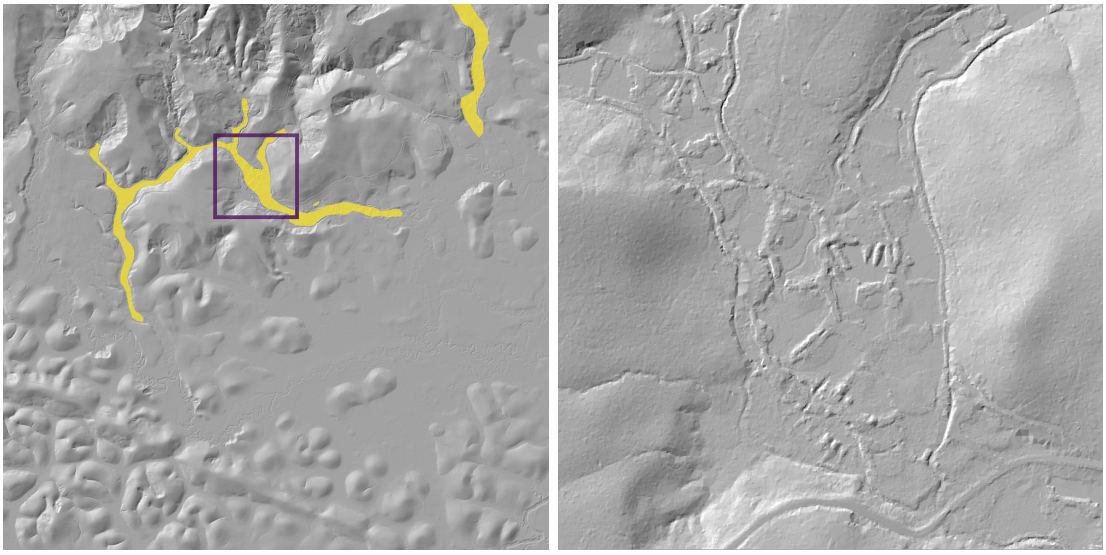
We carried out experiments on a large dataset to assess the efficiency of our method.

4.4.1. Dataset

The study area covers $255\,\mathrm{km}^2$ of tropical forest. The data was acquired by an aerial multi-echo LiDAR system at 100 m above ground level. The point cloud is first processed into a DTM of $1\,\mathrm{m}^2$ pixel resolution. To remove trees and vegetation, only ground points are kept. The raster is processed through triangulated interpolation. The resulting dataset is a large DTM with 32 bits floating-accuracy elevations. For the sake of readability, we used for this paper a sample of this dataset with a size of 5000×5000 pixels (Fig. 4.1a).

The ground truth of the areas of interest has been provided by geologists and geophysicists who are interested in automatically detecting natural lineaments or man-made structures. In the first case, it will be a question of locating structures rich in raw materials, in the second, the zones of gold panning which are a threat to the environment.

Two classes are presented as examples: gold panning zones, shown in Fig. 4.3a, and dikes.



(a) Gold panning areas (b) Closeup of an area of interest

Figure 4.3.: Visualizations of the ground truth used for the experiments. The ground truth is displayed over the hillshades of Fig. 4.1b. Yellow surfaces are gold panning areas and the purple square is the closeup (b).

4.4.2. Characterization of gold panning sites

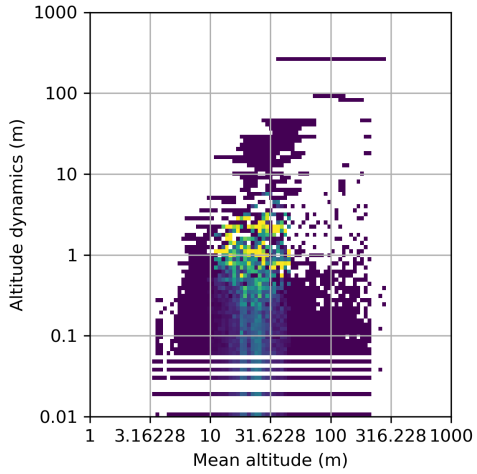
In order to illustrate the potential of our method, we deal here with the characterization of gold panning areas.

The first step is to compute the hierarchical representations of the DTM. We used the Higra library (Perret et al. 2019) to process several hierarchical representations and corresponding attributes. We chose several component trees: max-tree, min-tree and tree of shapes. For each node of a tree, we compute the attributes listed in Sec. 4.3. We then obtain a distribution of attribute values to be described. The range of the attributes were divided into classes.

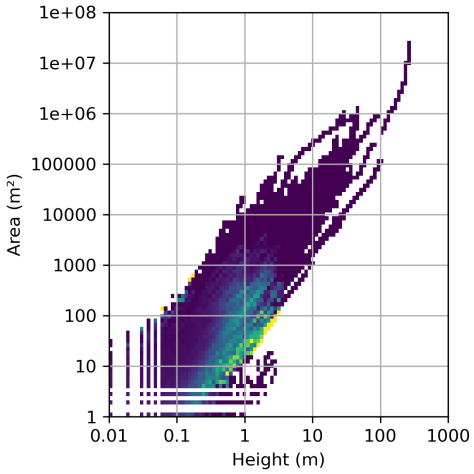
We processed a spectrum by choosing two attributes following the procedure detailed in Sec. 4.2.2. Figure 4.2 shows the spectrum of area and compactness with the max-tree.

The next step is to find the nodes corresponding to the area of interest. We used the ground truth (Fig. 4.3a) as a pixel activation map. To select the nodes of the tree corresponding to the gold panning areas, we traversed the tree in depth, from leaves to root. A node is selected if all the pixels of the activation map included in the node are active. If the node contains a disabled pixel, the node and its ancestors are not selected.

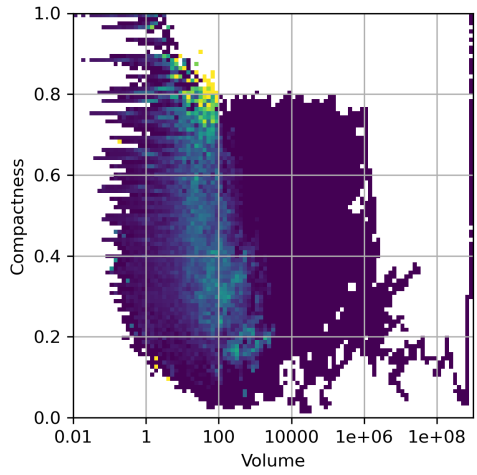
These selected nodes were used to process a new spectrum of the areas of interest.



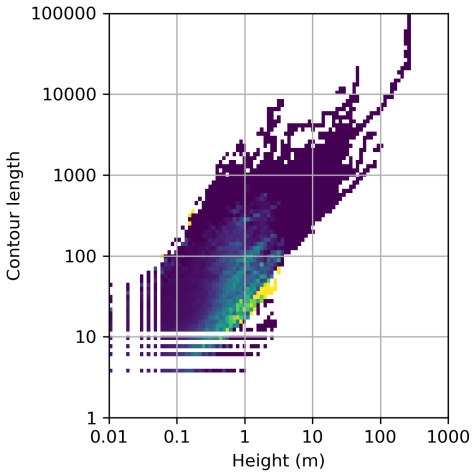
(a) Mean altitude and altitude dynamics



(b) Height and area

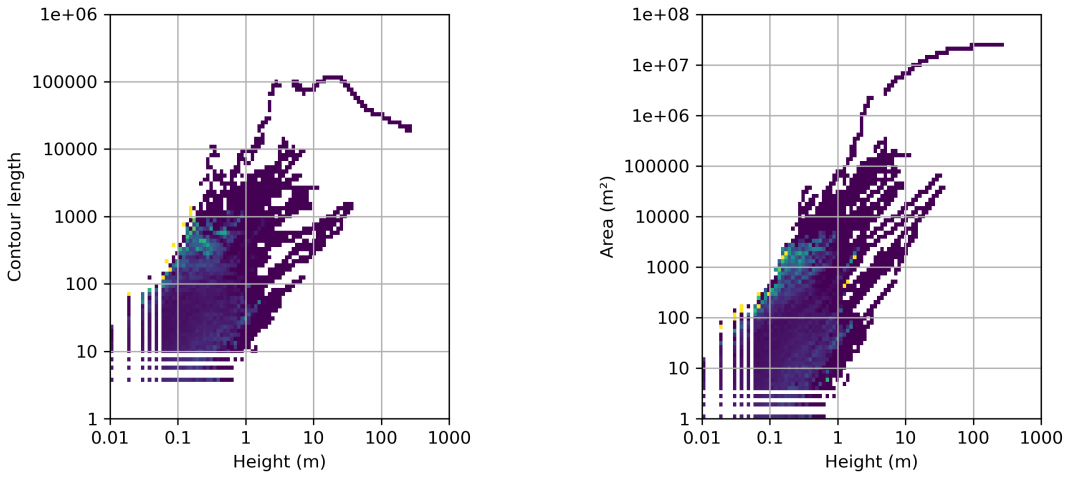


(c) Volume and compactness



(d) Height and perimeter

Figure 4.4.: Activation maps of pattern spectra over the gold panning areas with the max-tree. The bin color represents the percentage of nodes defined as gold panning in the global spectrum. Values range from 0% (purple) to 100% (yellow). The more consistent the spectrum, the better the characterization.



(a) Height and perimeter

(b) Height and area

Figure 4.5.: Activation maps of pattern spectra over the gold panning areas with the min-tree. The bin color represents the percentage of nodes defined as gold panning in the global spectrum. Values range from 0% (purple) to 100% (yellow).

To assess the relevance of a pair of attributes, we then searched for separability between areas of interest and background in the spectrum. We defined a metric based on the intersection of the spectrum of selected nodes and the spectrum of background nodes. The intersection was normalized with the weights of the selected nodes such as:

- *intersection* = 1 if the selected nodes are fully merged in the background nodes spectrum.
- *intersection* = 0 if the selected nodes do not share a single bin with the background spectrum.

This intersection metric is very severe for spectra since an object well segmented is likely to have numerous child nodes mixed with small noise-like objects inherent to hierarchical representations. However, this intersection metric is good enough for ranking the spectra among themselves.

We ran series of spectrum intersections by combining the attributes as well as the trees (i.e. max-tree, min-tree and tree of shapes). We compared the number of meaningful spectra per tree by counting the number of spectra with intersection below 0.95 (Table 4.1). The min-tree achieves good results in this context (Fig. 4.5). The gold panning sites have pits in the ground well characterized by the min-tree (i.e. structures that are lower than their neighborhood). Usually max-tree performs well on DTM and appeared

tree	count
Max-tree	10
Min-tree	23
Tree of shapes	6

Table 4.1.: Count of significant intersections (below 0.95) for each tree type.

more locally consistent in the spectra (Fig. 4.4). Tree of shapes seemed ill-suited for this experiment (Fig. 4.6). However, tree of shapes performances can be explained by our lack of rule to create dual classes with a normal distribution (e.g. height attributes ranges from −40 to +40 in the tree of shapes). Their dual representation calls for further investigations.

To get the most meaningful spectra, we selected the attribute combinations with the lowest intersection scores (Table 4.2). The best combination was achieved by the mean altitude and altitude dynamics spectrum (Figure 4.4a).

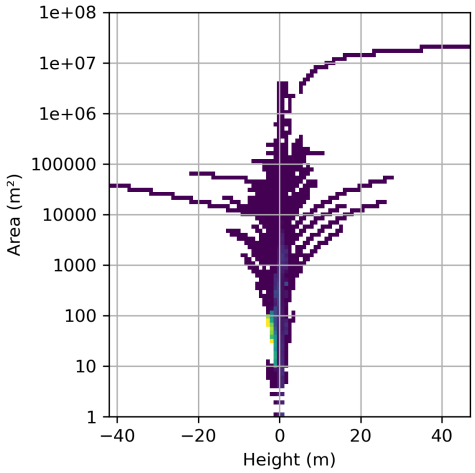


Figure 4.6.: Activation maps of pattern spectra over the gold panning areas with the tree of shapes. The bin color represents the percentage of nodes defined as gold panning in the global spectrum. Values range from 0% (purple) to 100% (yellow).

The best attribute combination had in common the mean altitude and the altitude dynamics. Both are strongly correlated with nodes altitude. In this sample zone, the gold panning areas shared the same elevations. To limit biases, we selected a subset of attributes unrelated to absolute altitude. These attributes are mostly related to

tree	ax	ay	score
Max-tree	mean_altitude	altitude_dyn.	0.4926
Min-tree	mean_altitude	altitude_dyn.	0.6231
Min-tree	mean_altitude	perimeter	0.8042
Min-tree	altitude_dyn.	area	0.8352
Tree of shapes	mean_altitude	altitude_dyn.	0.8499
Min-tree	altitude_dyn.	perimeter	0.8558
Max-tree	mean_altitude	perimeter	0.8639
Min-tree	mean_altitude	height	0.8929
Min-tree	mean_altitude	area	0.8957
Tree of shapes	mean_altitude	altitude_dyn.	0.8999
Min-tree	mean_altitude	compactness	0.9044

Table 4.2.: 2D spectra intersection of gold panning and background (lower is better).

shape (e.g. area, perimeter, compactness, height and volume). The best intersection scores of these attributes are visible in Table 4.4. Among them, we found that the best combination was the area and height spectrum (Figure 4.4b). An interesting spectrum combines the compactness and volume attributes (Figure 4.4c). This last spectrum is an underlying combination of area and height (via volume) together with compactness attributes.

We evaluated 3D pattern spectra following the same methodology. To build the 3D spectrum, we chose 3 attributes and create the subsequent classes. We ran the experiments with the previously selected attributes. The intersection metrics are available in Table 4.4. We can notice an overall better separability. The best attribute combination was area, height and compactness (Figure 4.8). In this figure, one can very well perceive a 3D cluster that characterizes well the gold panning areas.

Let us note that the 2D and 3D pattern spectra presented here can be easily extended to higher dimensions, but they will then require more advanced visualisation techniques.

4.4.3. Dikes Extraction

In this section, we discuss the feasibility of structure extraction from DTM without any prior knowledge. Our goal was to extract structures of interest such as dikes.

We used the max-tree according to the observations made in the previous section. In addition, our aim was to characterize above-ground structures.

We plot the spectrum in Fig. 4.7 (left). We can then interactively select bins in the spectrum. For this example, we made a rectangular selection in green at the bottom

tree	ax	ay	score
Min-tree	height	perimeter	0.9729
Min-tree	height	area	0.9800
Max-tree	height	area	0.9838
Min-tree	compactness	area	0.9877
Max-tree	height	perimeter	0.9879
Min-tree	volume	area	0.9887
Min-tree	volume	compactness	0.9899
Max-tree	volume	perimeter	0.9909
Max-tree	volume	height	0.9911
Tree of shapes	height	compactness	0.9921
Max-tree	volume	compactness	0.9952

Table 4.3.: 2D spectra intersection of gold panning and background with selected attributes (lower is better).

tree	ax	ay	az	score
Min-tree	compactness	height	area	0.5336
Max-tree	compactness	height	area	0.5362
Max-tree	perimeter	compactness	height	0.5643
Min-tree	perimeter	compactness	height	0.5768
Min-tree	compactness	volume	height	0.5983
Max-tree	compactness	volume	height	0.5990
Min-tree	perimeter	height	area	0.6611
Min-tree	perimeter	volume	height	0.6726
Max-tree	compactness	volume	area	0.6751
Min-tree	compactness	volume	area	0.6845
Max-tree	perimeter	compactness	volume	0.7107
Max-tree	perimeter	height	area	0.7388
Tree of shapes	compactness	height	area	0.7394

Table 4.4.: 3D spectra intersection of gold panning and background with selected attributes (lower is better).

center of the spectrum to select structures with height between 35 m and 60 m and compactness around 0.10 (i.e. rather non-compact shapes). We pruned the tree to keep only the nodes from the selected classes. The results can be displayed in different ways:

- Retrieve the footprint of the structures. It is useful for visual inspection by displaying the results as an overlay over the DTM or hillshade visualizations (on right side of Figure 4.7).
- Reconstruct the altitudes of the pruned tree with direct filtering (Salembier, Oliveras, et al. 1998).
- Reconstruct the selected structures from the tree with the subtractive filtering. Subtractive filtering introduced by (Urbach and Wilkinson 2002) allows to sum the altitude differences of the nodes with their direct parents. The filtering result, when used with DTM, is the height of structures selected in the tree.

For dike detection, a false positive structure detection is visible at the bottom left of the image (Fig. 4.7 right). Since compactness is a ratio involving the perimeter, it is expected but undesired to get such results with non-convex shapes. In future works, we will investigate the use of elongation attributes such as the first moment invariant of Hu (Hu 1962) for this use case.

4.5. Conclusion

This study focuses on DTM structure characterization with the pattern spectra. The pattern spectra offer many benefits when used with DEM. On the one hand, the pattern spectra offer a global and multi-scale description of the objects contained in the DEM. On the other hand, the direct link between DEM values and level definition used in the pattern spectra gives an intrinsic meaning of the attributes commonly used on mathematical morphology. Furthermore, the use of underlying hierarchical representations to compute the pattern spectra and reconstruct characterized objects enables interactive applications.

Future works will be carried out on the use of new attributes suitable with DEM and user interface for 3D spectrum exploration.

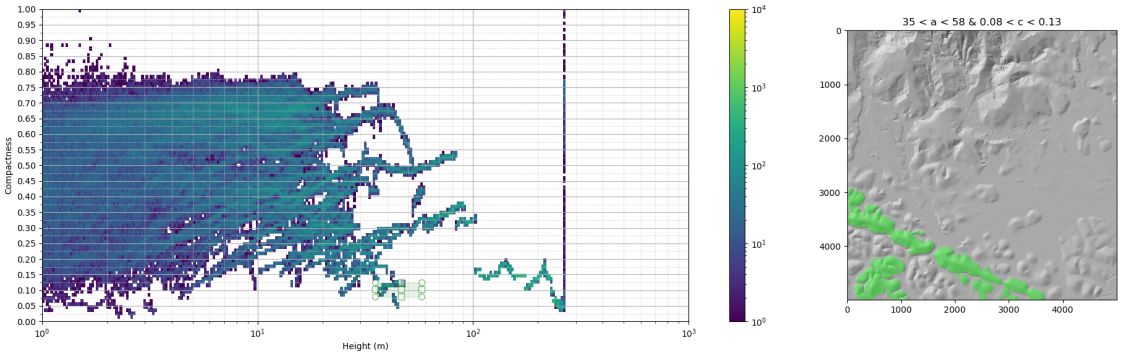


Figure 4.7.: Left: Interactive selection on pattern spectrum (height and compactness) of the DTM with the max-tree. Right: Structures of the DTM corresponding to the selection are displayed with a green overlay on top of the hillshade visualization.

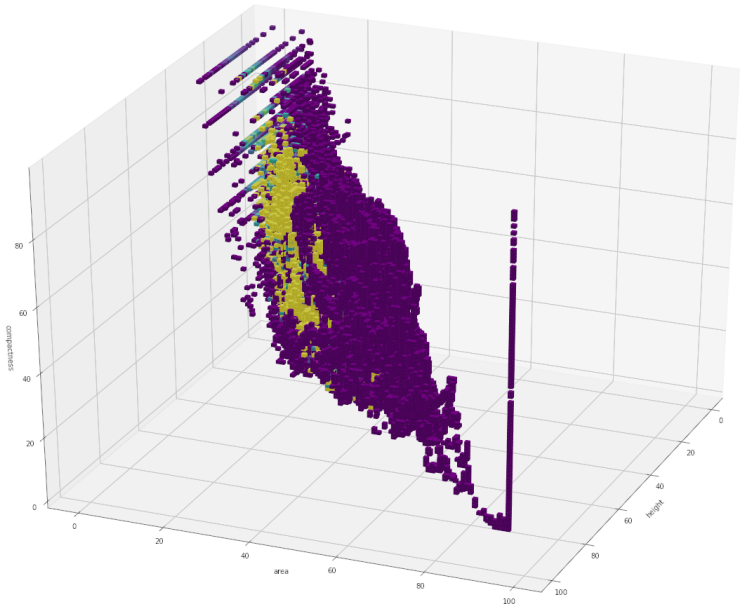


Figure 4.8.: Activation map of 3D pattern spectrum (area, height and compactness). The bin color represents the percentage of nodes defined as gold panning in the global spectrum. Values range from 0% (purple) to 100% (yellow).

SEMANTIC SEGMENTATION OF LIDAR POINTS CLOUDS: RASTERIZATION BEYOND DIGITAL ELEVATION MODELS

DOI 10.1109/LGRS.2019.2958858

Authors Florent Guiotte, Minh-Tan Pham, Romain Dambreville, Thomas Corpetti, Sébastien Lefèvre

Keywords LiDAR, DEM, rasterization, Deep Learning, Semantic Segmentation.

Objectives *This work propose to segment LiDAR data with deep neural networks.*
We use the same LiDAR features and the same evaluation framework than in Chapter 3. We use the deep model SegNet over the LiDAR features to provide a supervised segmentation over the University of Houston.

Results As expected, the results are improved in compared to the ones from Chapter 3. We report the same trend regarding the good potential of DEMs to bear meaningful information for semantic segmentation of urban classes.
This method is robust and reliable to segment land cover maps over urban area.

Contents

Abstract LiDAR point clouds are receiving a growing interest in remote sensing as they provide rich information to be used independently or together with optical data sources such as aerial imagery. However, their non-structured and sparse nature make them difficult to handle, conversely to raw imagery for which many efficient tools are available. To overcome this specific nature of LiDAR point clouds, standard approaches often rely in converting the point cloud into a digital elevation model, represented as a 2D raster. Such a raster can then be used similarly as optical images, e.g. with 2D convolutional neural networks for semantic segmentation. In this letter, we show that LiDAR point clouds provide more information than only the DEM, and that considering alternative rasterization strategies helps to achieve better semantic segmentation results. We illustrate our findings on the IEEE DFC 2018 dataset.

5.1. Introduction

Thanks to their very high resolution, LiDAR point clouds are known to be of very high interest to identify complex structures, especially in urban environments, such as trees, road, cars…However, as related point clouds are voluminous and irregularly distributed, land cover mappings are in many studies often simplified to a DEM used as additional information for fusion with multispectral or hyperspectral images.

In this study, we start from the idea that this rasterization step related to the production of a single DEM is not optimal as many additional information embed in the LiDAR point cloud is lost. We then rather prefer to focus on the extraction of more advanced rasterized maps. In a first attempt with the same dataset and the same evaluation protocol(Guiotte, Lefèvre, et al. 2019c), we computed a series of attribute profiles on 2D grids containing various information extracted during the mapping from 3D to 2D (number of points in a cell, first echo, last echo, …); these features have fed a simple Random Forest classifier with efficient results. In this paper, we extend this work by considering a deep learning network.

More than designing the most adapted network, our aim is rather to show that LiDAR point clouds provide more information than only the DEM, and that considering alternative rasterization strategies helps to achieve better semantic segmentation results. We illustrate our findings on the IEEE DFC 2018 dataset.

The paper is organized as follows. We review related works in Sec. 5.2. We then present in Sec. 5.3 various strategies that can be employed to map a 3D point cloud into a 2D raster. After recalling the deep network architecture used in this paper (Sec. 5.4), we report in Sec. 5.5 the outcomes of our experiments conducted on the IEEE DFC 2018 dataset. We finally conclude the paper in Sec. 5.6.

5.2. Related work

LiDAR point clouds have become a popular remote sensing data source for land cover mapping. Recent developments have allowed precise point cloud segmentation, especially using. However large point clouds like those provided by airbone LiDAR are more challenging for direct end to end learning because of the large amount of data and their unstructured nature, as opposed to regular 2D grids in images(Landrieu and Simonovsky 2018). Therefore, to address this problem, many authors either suggest to reorganise the point cloud into regular 2D grids and/or to exploit the multispectral information. These directions are detailed in the following.

As a matter of fact, recent LiDAR sensors now provide multispectral signals, through the generation of a point cloud specific to a given wavelength. A few studies have been reported with such data. We can mention (WANG, TSENG, et al. 2014) which showed that using dual wavelength led to substantial improvements in land cover mapping w.r.t. a single wavelength. In (WICHMANN et al. 2015), a multi-spectral LiDAR system was used to classify ground with pattern matching classifier applied pointwise on intensities and NDVI.

Another possibility consists in using various rasters computed on a LiDAR point cloud. Let us note that this idea is not totally new, and a few recent attempts have been made in this direction, as for example (SUKHANOV et al. 2018) were the DFC 2018 dataset is classified with LiDAR only or together with other optical data. Several features are extracted based solely on LiDAR information (e.g. median to altitude, intensity and number of echoes) or combined with other information (composition of spectral intensities, intensity ratio, brightness, difference between DSM, etc), and/or with local features computed solely on intensity. Experiments were conducted using several classifiers: RF, gradient boosting machine and CNN on 20 classes. While this study had shown the relevance of combining multiple data sources to process the DFC 2018 dataset, it did not allow to derive any conclusion regarding the relevance of alternative LiDAR rasters and their specific performance with deep semantic segmentation networks.

In a very recent study (GUIOTTE, LEFÈVRE, et al. 2019c), we have shown that such alternative LiDAR rasterizations actually provide additional information source that can help to describe the contents of a remotely-sensed scene, and improve its classification. This was demonstrated using well-established multilevel features (attribute profiles) and classifier (RF). Nevertheless, given the widely-recognized performance of deep learning for semantic segmentation, the interest of such rasters as inputs to a deep network has still to be demonstrated. This is the goal of this letter and the next section discusses about rasterization strategies.

5.3. rasterization of unstructured point clouds

The main benefits of LiDAR rasterization are:

- To **reduce the complexity** since data are represented on a regular grid;
- To provide a **regular sampling** (easier to manipulate neighbours) instead of dealing with irregular point clouds;
- To have a prior **known** number of data unlike unknown number of point clouds;

- To **reduce the radiometric and altimetric artefacts** thanks to the aggregation of values.

In general the use of a DEM only is not optimal since LiDAR systems enable to capture complex patterns in the three dimensions and a DEM aggregates the vertical information. This loss in the vertical direction is prejudicial since many urban objects are characterised by their vertical structure. For example in vegetated areas, the information in the vertical direction enables to capture the whole trees structure (and not only their surface). In addition, the vertical information detects ground below vegetation or objects below trees such as residential buildings, roads and cars. Therefore, we suggest here to provide rasters where such information in the vertical direction is kept. In comparison with other works (SUKHANOV et al. 2018), we tried to summarise this vertical component by creating several feature maps based on the vertical distribution, in addition to usual DEMs. The general rasterization process can be defined in three steps detailed below:

1- The reorganisation of the point clouds by binning them into a regular grid. More formally, we apply a transformation $\mathcal{PR}_{h,f}$ (for "points to raster", associated with a discretization step h and an information function f) defined on the dataset as:

$$\begin{aligned}
\mathcal{PR}_{h,f}:\ & \mathbb{R}^3 \times \mathbb{R} \ \longrightarrow {}_h \times \mathbb{R} \\
& \{x, y, z, \mathcal{I}\} \longmapsto \ \{i, j, \mathbf{I}(_{i,j})\} \text{ with:} \\
& \begin{cases}
{}_{i,j} \quad \text{the set of points s.t.} \\
i \text{ s.t. } x_m + ih \leq x < x_m + (i+1)h \\
j \text{ s.t. } y_m + jh \leq y < y_m + (j+1)h \\
\mathbf{I}(_{i,j}) = f(i, j, {}_{i,j}, \mathcal{I})
\end{cases}
\end{aligned} \tag{5.1}$$

with ${}_h$ the raster grid, x_m and y_m the minimum values of all points x and y in the dataset . The rule of function f is to associate to each cell location (i, j) an information related to the data point ${}_{i,j}$ included in the cell. Its value is discussed below.

2- The extraction of Lidar feature maps. Many functions f can be defined to provide rasters. For example a DEM high and a DEM low uses respectively the positions of the maximum and the minimum z coordinates i.e., first and last returns inside a cell ${}_{i,j}$:

$$\begin{aligned}
f_{Dh}(i, j, x, y, z, \mathcal{I}) &= \max(z) \text{ s.t } (x, y, z) \in {}_{i,j} \\
f_{Dl}(i, j, x, y, z, \mathcal{I}) &= \min(z) \text{ s.t } (x, y, z) \in {}_{i,j}
\end{aligned} \tag{5.2}$$

To enable more flexibility, other functions can be used such as the intensity of the highest and lowest points:

$$f_{Ih}(i,j,x,y,z,\mathcal{T}) = I(x^p,y^p,z^p),\ p \text{ being the point s.t.} \\ z^p = \max(z)_{(x,y,z)i,j} \tag{5.3}$$

$$f_{Il}(i,j,x,y,z,\mathcal{T}) = I(x^p,y^p,z^p),\ p \text{ being the point s.t.} \\ z^p = \min(z)_{(x,y,z)_{i,j}} \tag{5.4}$$

or the number of echoes per cell:

$$f_N(i,j,x,y,z,\mathcal{T}) = |_{i,j}| \tag{5.5}$$

3- The interpolation of empty cells. If the discretization step h is small, empty bins are likely to appear. In this work, we fill them using a linear interpolation.

5.4. Neural network

Deep learning approaches and particularly deep convolutional neural networks are currently unrivalled at the top of the state of the art for semantic segmentation applications. To evaluate the pertinence of our different rasters, we exploited the SegNet model (Badrinarayanan et al. 2017) which has been widely used for semantic segmentation in computer vision domain. In remote sensing, this network has also proved its effectiveness on multispectral images with visible (RGB) and infrared bands in (Audebert, Le Saux, and Lefèvre 2018). The SegNet model relies on an encoder-decoder architecture based on convolutional layers of the VGG-16 network (Simonyan and Zisserman 2015), followed by batch normalization, rectified linear unit (ReLU) and then pooling and unpooling layers (w.r.t the encoder and decoder, respectively) (Badrinarayanan et al. 2017). The input of SegNet has usually three channels by default. In our work, not only each of LiDAR rasters but also their combinations will be used

5.5. Experiments

We first introduce the dataset and the LiDAR rasters we used before discussing the efficiency of the proposed rasters.

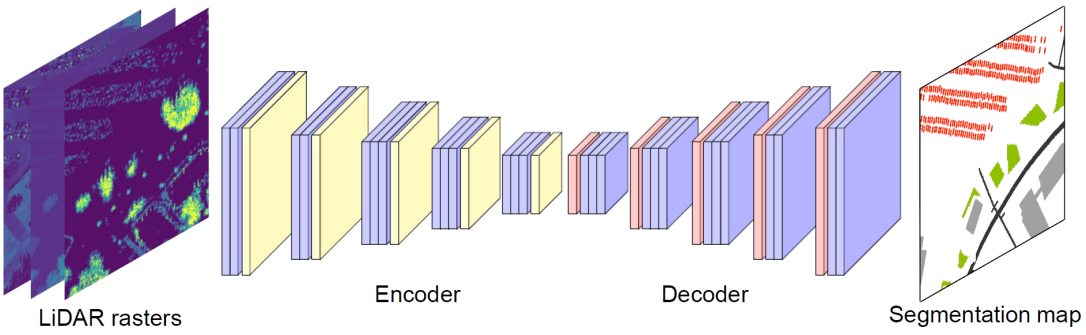


Figure 5.1.: Overview of the SegNet architecture with LiDAR rasters as input.

5.5.1. Dataset

We chose the multi-spectral LiDAR acquisition of the University of Houston issued from 2018 IEEE GRSS Data Fusion Contest dataset (Xu et al. 2019) to support our experiments. The associated ground truth map has a spatial resolution of 0.5 m. The original 20 classes have been reduced to 7 generic urban classes (roads, grass, trees, residential buildings, non-residential buildings, cars and trains) to evaluate the overall accuracy.

5.5.2. LiDAR feature maps

To generate our rasters, the grid step was set to $h = 0.5$ m to fit with the ground truth. We removed the first and last 0.1 percentiles of the point cloud based on elevation distribution since they are more likely to be outliers.

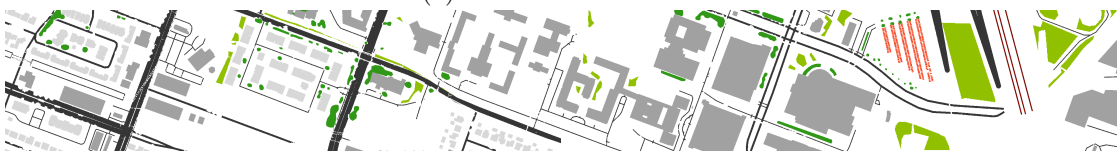
We chose to gather the geometric information contained in the 3 wavelengths to get a dense point cloud. With this composite point cloud we created several elevation rasters:

- Highest and lowest point in the cell (i.e. DEM and "reversed" DEM), noted D_h, D_l and corresponding to the use of function f_{Dh} (5.2));

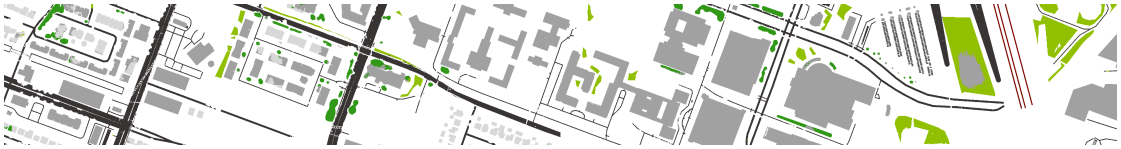
Then, with each point cloud separately we created intensity and echo rasters:

- Intensity of the highest and lowest points in the cell (noted I_h, I_l and corresponding to function (5.3));
- Number of echoes per beam in the cell (noted N and corresponding to function (5.5)).

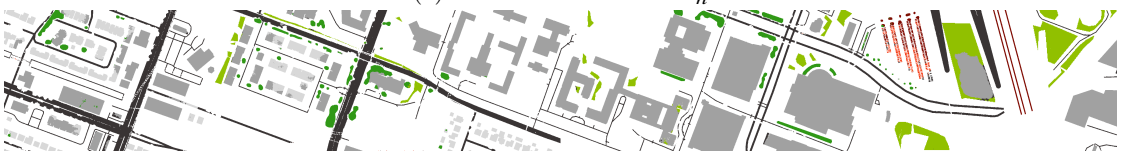
(a) Ground truth train



(b) Ground truth test



(c) Classification with D_h



(d) Classification with I_h, I_l, D_h

Figure 5.2.: Illustration: Horizontal disjoint split between training set (a) and test set (b); (c) Classification result using DEM (D_h) only; (d) Classification result using the combination of $\{I_h, I_l, D_h\}$.

5.5.3. Classification

Training phase: The dataset has been divided into a training set and a test set with an horizontal split from the original data (*disjoint split*) as shown on Fig.5.2-(a)-(b). We used the code from (Audebert, Le Saux, and Lefèvre 2018)[1] to perform all the experiments with parameter setting as default in (Audebert, Le Saux, and Lefèvre 2018) (learning rate 0.01 with momentum 0.9 and weight decay 5×10^{-4}) for a fair comparison. As the input size of our SegNet model varies w.r.t. the rasters or different feature combinations, the network was trained from scratch. During the training, we randomly extracted 256×256 image patches from the training set. Batch size was set to 16 and all experiments were stopped after 20 epochs.

5.5.4. Experimental results

In table 5.1, we present the accuracy per-class, the Average Accuracy (AA), Overall Accuracy (OA) and Cohen's Kappa coefficient (κ) for each feature (see section 5.5.2) and their combination. As one can observe, the use of a DEM only is globally far from optimal (except for grass and roads where this value is really meaningful while other features slightly disturb the identification) despite the fact that most studies exploit only this property when rasterizing LiDAR point clouds. The use of the last echos (position D_l and intensity I_l) enable to greatly improve the classification. These echos are related to structures behind vegetated areas and provide very relevant information, as noticed on the classification results. As for the number of echos N, its value combined with other features enables to discriminate more properly only trees (where many echos are included) and residential areas (where only one echo is present) but it does not improve the overall classification in our experiments. Finally, the combination of the DEM, first and last intensities enables to provide the best classification results. This demonstrates the fact the LiDAR data are very rich and are currently not optimally exploited when they are rasterized in a DEM only.

5.6. Conclusion

In this letter, we explored the use of alternative rasters (beyond the standard DEM) to classify LiDAR point clouds. We measured the performance of a well-established deep neural network for multispectral semantic segmentation with different rasters extracted from the multispectral LiDAR point cloud provided with the IEEE DFC 2018.

[1] `https://github.com/nshaud/DeepNetsForEO`

		Per-class accuracy (%)							Evaluation metrics		
		Grass	Trees	Residential	Non-res building	Roads	Cars	Trains	AA(%)	OA(%)	κ(×100)
1	D_h	**95.62**	77.66	44.18	98.72	**99.83**	0.09	100.00	73.72	90.13	85.24
2	N	16.08	92.48	46.50	97.63	95.39	6.50	100.00	64.94	87.27	78.37
3	I_l	56.94	89.35	41.96	**99.39**	97.16	98.81	99.94	83.36	86.98	80.10
4	D_l	90.93	89.67	22.09	99.12	99.57	0.00	99.94	71.57	87.67	81.31
5	I_h	59.82	95.15	47.78	98.27	96.43	**99.98**	99.96	85.34	87.54	81.14
6	$\{N, I_h, D_h\}$	93.15	**97.72**	39.47	98.99	99.66	13.42	100.00	77.49	90.66	86.01
7	$\{I_h, I_l, D_h\}$	81.64	94.12	76.95	98.36	99.38	57.10	100.00	**86.87**	**93.88**	**91.00**
8	$\{I_h, I_l, D_h, D_l\}$	87.63	92.84	60.20	93.32	96.98	68.16	100.00	86.44	92.31	88.57
9	$\{N, I_h, I_l, D_h, D_l\}$	79.13	95.35	**82.70**	96.28	99.19	10.05	100.00	80.38	92.42	88.97

Table 5.1.: Per-class accuracy, average accuracy (AA), overall accuracy (OA) and Cohen's Kappa coefficient (κ) for each feature and combination of them using Segnet.

Our results show that the DEM is not the most discriminative feature, and that alternative features can be more helpful for land cover mapping.

Furthermore, an advantage of our map-based method is to allow us to rely on image (raster) segmentation networks with no or very small adaptation effort, instead of requiring to design specific networks dedicated to point clouds. Since semantic segmentation of images is a very active topic in computer vision, our approach will allows LiDAR processing tasks to benefit from future developments in the field.

Among future works, we would like to see if combining the different features in a same network leads to better results. Indeed, it is a promising direction given our preliminary results with non-deep learning techniques (Guiotte, Lefèvre, et al. 2019c). Furthermore, we plan to investigate deep architectures among those well-established for semantic segmentation.

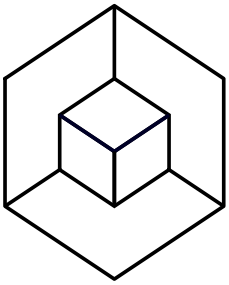
Acknowledgment

The authors would like to thank the National Center for Airborne Laser Mapping and the Hyperspectral Image Analysis Laboratory at the University of Houston for acquiring and providing the data used in this study, and the IEEE GRSS Image Analysis and Data Fusion Technical Committee.

The authors would like to thank the helpful comments of the anonymous reviewers of this article.

Part III.

3D Voxelization



ATTRIBUTE FILTERING OF URBAN POINT CLOUDS USING MAX-TREE ON VOXEL DATA

DOI 10.1007/978-3-030-20867-7_30
Authors Florent Guiotte, Thomas Corpetti and Sébastien Lefèvre.
Keywords Point clouds, Max-tree, Rasterization, Voxel and Attribute filtering.

Objectives *This work provides a framework to use hierarchical representations on LiDAR data while preserving the 3D structure of the point cloud.* We re-organise the complex 3D LiDAR point clouds into regular 3D voxel grids. We then build the max-tree of the 3D voxels and perform attribute filtering. The results are reprojected into the 3D point cloud space.

Results This setup allows to process point clouds via an intermediate voxel grid. The method is working efficiently at the cost of discretization artifact: the resolution of the grid determines the atomic precision in the point cloud. *The proposed method allows to process efficient attribute filtering on LiDAR point clouds.*

Contents

Abstract This paper deals with morphological characterization of unstructured 3D point clouds issued from LiDAR data. A large majority of studies first rasterize 3D point clouds onto regular 2D grids and then use standard 2D image processing tools for characterizing data. In this paper, we suggest instead to keep the 3D structure as long as possible in the process. To this end, as raw LiDAR point clouds are unstructured, we first propose some voxelization strategies and then extract some morphological features on voxel data. The results obtained with attribute filtering show the ability of this process to efficiently extract useful information.

6.1. Introduction

Thanks to the advances both in technologies such as laser scanning (LiDAR) and in methods from photogrammetry, digital point clouds form a popular object of study in many scientific fields such as geosciences (flow, erosion, rock deformations, …), computer graphics (3D reconstruction), urban environments analysis from Earth Observation (detection of trees, roads, buildings, …). In the context of urban scenes, they provide a rich 3D information w.r.t. digital 2D photographs.

Despite their growing interest, only limited studies have explored how to apply mathematical morphology on point clouds (Calderon and Boubekeur 2014; Peternell and Steiner 2007). The usual approach remains to first rasterize the point cloud to obtain a digital image (also called a raster of pixels) on which standard morphological operators are applied (Serna and Marcotegui 2014). This strategy was also recently followed for morphological hierarchies on LiDAR point clouds (Guiotte, Lefèvre, et al. 2019c).

In this paper, we claim that discretizing a point cloud into a 2D raster leads to an oversimplification of the image that greatly reduces the potential of morphological operators. Thus, we consider here a 3D discretization in a voxel grid as illustrated in Figure 6.1. Such an approach allows us to benefit from efficient algorithms that have been introduced for morphological hierarchies, while still maintaining the 3D information. We illustrate our solution with a very popular processing, namely attribute filtering, that is applied on an urban point cloud. The reported results show the relevance and the potential of this approach.

The paper is organized as follows. In Section 6.2 we review morphological approaches (especially those based on hierarchies) for point clouds and 3D processing. We then explain the different steps of our method in Section 6.3, before illustrating it in the urban remote sensing context in Section 6.4. Section 6.5 concludes this manuscript and provides directions for future research.

6.2. Related Work

6.2.1. Mathematical morphology on point clouds

Conversely to digital images that are usually defined on a discrete 2D grid, a point cloud is characterized by a sparse set of points defined with continuous coordinates. So dealing with a 3D point cloud raises many issues including the lack of efficient processing algorithms. One of the most critical questions is the definition and fast computation of

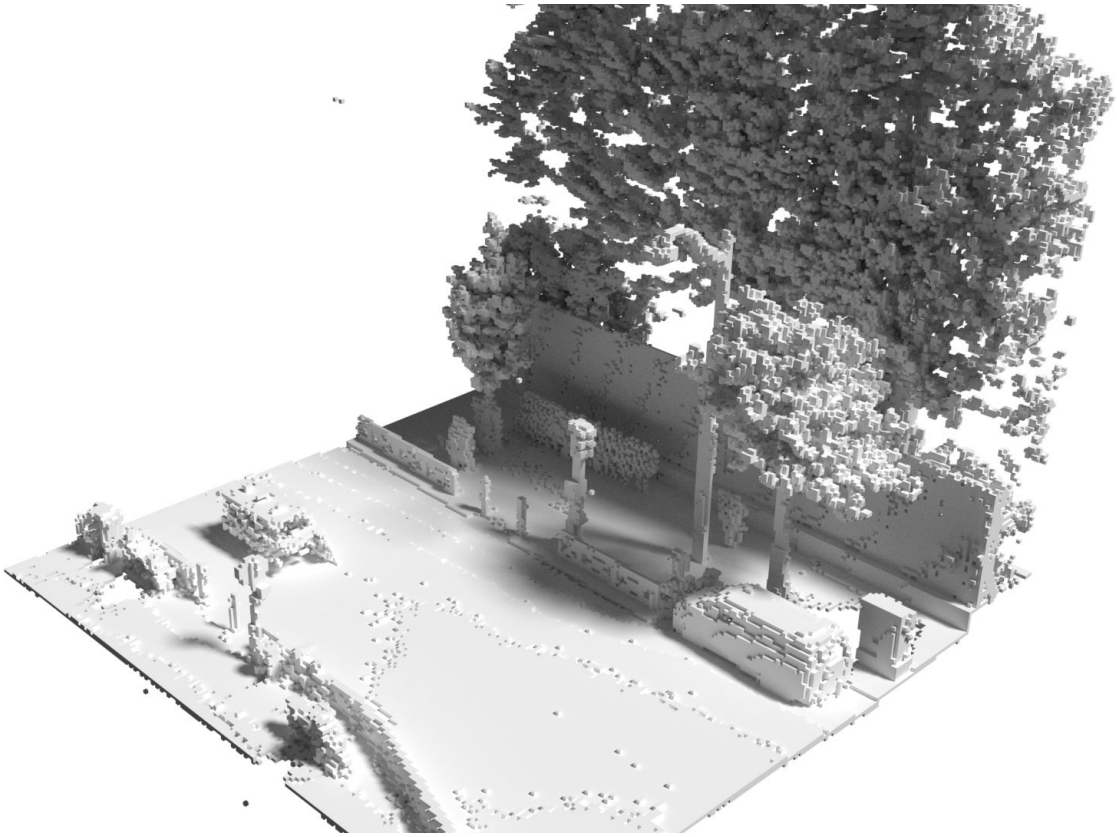


Figure 6.1.: LiDAR data projected into a voxel grid.

the neighborhood of a point, which is a fundamental concept in morphology.

As already stated, the usual approach is to project the 3D point cloud in a discrete grid (either 2D or 3D) where the neighborhood computation becomes straightforward. Thus, LiDAR point clouds were considered in (Gorte and Pfeifer 2004) to characterize vegetation (trees). The authors use the point density to define voxel values, that are further processed with 3D adaptation of standard 2D morphological methods. In the astronomical context, the discretization proposed by (Ferdosi et al. 2010) also relies on the density, this latter being estimated using adaptive kernel. A max-tree is then used to find local maxima that allow for identification of relevant subspaces for clustering the data.

More closely related to our study, a few attempts have been made to process urban point clouds. In (Aijazi et al. 2013), the segmentation and classification of an urban point cloud is achieved by means of super-voxels. They are created using a distance computed in the feature space between voxel properties such as distribution (by mean, variance) of spectral values (intensity or color information). Finally, a reprojection step is involved to obtain the resulting labeled point cloud. The same tasks are addressed by (Serna and Marcotegui 2014), with a different approach though. The point cloud is projected in a DEM before applying 2D morphology and classification. The results are then reprojected into a point cloud. More recently, the same authors have addressed segmentation of facades with attribute profiles (Serna, Marcotegui, and Hernández 2016). Combination of DEM morphology and attribute filters (elongation) were considered on binary 3D images denoting an occupancy grid. Beyond these works on 2D and 3D rasters, a few works have been conducted directly on the continuous space of 3D points cloud, such as (Calderon and Boubekeur 2014; Peternell and Steiner 2007). However the morphological methods introduced in these papers are dedicated to point clouds describing surfaces, and as such cannot be used with LiDAR data since a surface can not be reconstructed in each situation (points are also likely to belong to the inside of objects, for example in vegetation areas).

6.2.2. Morphological hierarchies on 3D images

The extension of morphological hierarchies (e.g. max-tree) from a 2D image to a 3D volume is rather straightforward. Unsurprisingly, it led to several works attempting to use it on 3D voxels, especially in the medical domain. Early work in (Wilkinson and Westenberg 2001) has for example used the max-tree to filter 3D images with volume and inertia attributes. Later on, the max-tree was used to filter and visualize the medical

images (Westenberg et al. 2007) with three new 3D moment-based attributes (elongation, flatness and sparseness). Another 3D attribute was proposed in (Kiwanuka, Ouzounis, et al. 2009) to estimate the sphericity of objects. It was based on the computation of surface and volume of connected components and aims to be more efficient than the previous measures. Roundness of objects was estimated through another 3D attribute for max-tree filtering in (Kiwanuka and Wilkinson 2012). Filtering medical images has also received a lot of attention until very recently, e.g. (Dufour et al. 2013; Grossiord et al. 2015; Urien et al. 2017; Padilla et al. 2018). Finally, we can mention the work of (Géraud et al. 2013) to compute the tree of shapes of nD images.

6.3. Method

In order to filter point clouds using hierarchical representations, we propose to rely on a prior discretization of the continuous domain into a regular 3D voxel grid (instead of a 2D raster). This intermediate representation allows us to use directly mathematical morphology with the 3D data. We then reproject the results of the morphological filtering into the continuous domain (i.e. as a new point cloud).

6.3.1. From point cloud to voxel grid

A raw dataset issued from LiDAR acquisitions lives in $\mathbb{R}^3 \times \mathbb{R}$ where each data $= \{x, y, z, \mathcal{I}\} \in$ is such that the intensity taken in location (x, y, z) is $\mathcal{I}$. Though very interesting, the irregularity of available locations (x, y, z) prevents from the use of tools devoted to structured data with ordering relations as images or volumes. To cope this difficulty, we suggest to transform this dataset into a structured volume. This "voxelization" step aims at defining the data on a regular 3D grid $_h \subset \mathbb{N}^3$ with a given spatial resolution h (for the sake of simplicity, we consider here isotropic resolutions but the method can be applied with anistropic ones) such that the value taken in any point $(i, j, k) \in {}_h$ represents an information about initial data. This information can either be a boolean (related to the presence/absence of points in the voxel), the number of LiDAR points into the voxel, the average/standard deviation of associated intensities, the average/standard deviation of associated elevations, the majority label (for 3D labels), etc.

More formally, we apply a transformation $\mathcal{PR}_{h,f}$ (for "points to voxels", associated

with a discretization step h and an information function f) defined as:

$$\begin{aligned}
\mathcal{PR}_{h,f} :\ & \mathbb{R}^3 \times \mathbb{R} \ \longrightarrow\ _h \times \mathbb{R} \\
& \{x, y, z, \mathcal{I}\} \longmapsto\ \{i, j, k, \mathbf{I}\} \text{ with:} \\
& \qquad \begin{cases} i & \text{s.t. } x_m + ih \leq x < x_m + (i+1)h \\ j & \text{s.t. } y_m + jh \leq y < y_m + (j+1)h \\ k & \text{s.t. } z_m + kh \leq z < z_m + (k+1)h \\ \mathbf{I} & = f(i, j, k, x, y, z, \mathcal{I}) \end{cases}
\end{aligned} \tag{6.1}$$

with x_m (resp. (y_m, z_m)) the minimum value of all points x (resp. y, z) in the dataset . The rule of function f is to associate to each voxel location (i, j, k) an information related to the original data points. Let us denote $\mathcal{I}_{i,j,k}$ the set of intensities $\mathcal{I}$ of points (x, y, z) inside a voxel (i, j, k) (i.e. fulfilling the 3 first conditions of (6.1)). Its cardinal is noted $|\mathcal{I}_{i,j,k}|$.

Many functions f can be defined, as for example:

- **Boolean** (noted f_b):

$$f_b(i, j, k, x, y, z, \mathcal{I}) = \begin{cases} 1 & \text{if } |\mathcal{I}_{i,j,k}| \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

- **Density** (noted f_d, similar to (Ferdosi et al. 2010)):

$$f_d(i, j, k, x, y, z, \mathcal{I}) = |\mathcal{I}_{i,j,k}| \tag{6.3}$$

- **Empirical average intensity** (noted f_a):

$$f_a(i, j, k, x, y, z, \mathcal{I}) = \frac{1}{|\mathcal{I}_{i,j,k}|} \sum \mathcal{I}_{i,j,k} \tag{6.4}$$

- **Empirical standard deviation of intensity** (noted f_s):

$$f_s(i, j, k, x, y, z, \mathcal{I}) = \sqrt{\frac{1}{|\mathcal{I}_{i,j,k}|} \sum \left(\mathcal{I}_{i,j,k} - f_a(i, j, k, x, y, z, \mathcal{I})\right)^2} \tag{6.5}$$

with f_a and f_s defined only if $f_b \neq 0$. Depending on the sought applications, many other functions can be used, for example related to the geometry (normal surface, main

orientation, ...) or any other features of $\mathcal{T}_{i,j,k}$.

It should be outlined that empty cells can occur from two situations: 1) empty spaces into the scene or 2) missing data because of occlusions. Several approaches are possible to deal with such empty voxels (affecting a value 0, linear interpolation, ...). In this study, and without loss of genericity, we chose to assign them the null value.

Once the voxelization transformation $\mathcal{PR}$ is performed, our data $(i,j,k,\mathbf{I}) \in {}_h \times \mathbb{R}$ can be represented through a volume V such that:

$$\begin{aligned} V : \quad {}_h &\longrightarrow \mathbb{R} \\ (i,j,k) &\longmapsto \mathbf{I} \,. \end{aligned} \tag{6.6}$$

6.3.2. Attribute filtering with the max-tree of voxels

Max-tree

Attribute filtering is a popular tool in mathematical morphology. It operates on connected components of an image (if binary) or of its level sets (if greyscale). As a connected filter, it does not shift object edges but proceeds by removing the components that do not fulfill a given criterion (related to the aforementioned attribute). It benefits from efficient implementation through the image representation as a max-tree.

As already stated, the usual definition of the max-tree for 2D images remains valid in case of 3D volumes. Only the connectivity needs to be updated, from 4- and 8-connectivity in 2D to 6-, 18- and 26-connectivity in 3D. The upper level sets of the volume V are obtained from successive thresholdings of the grey levels $l \in \mathbb{R}$ and noted

$$\mathcal{L}_l = \{(i,j,k) \in {}_h \mid V(i,j,k) \geq l\} \,. \tag{6.7}$$

We index by c the connected components within a level set, i.e. $\mathcal{L}_{l,c}$. These components are nested and form a hierarchy called the max-tree. The leaves of the tree correspond to the regional maxima while the root contains the whole volume.

Filtering

The max-tree structure provides an efficient way to filter its nodes (i.e. the connected components of the level sets). Such a filtering relies on some predefined criteria called attributes, whose values are usually computed for each node during the tree construction step.

We distinguish here between two kinds of attributes that are scale-dependent and scale-invariant, respectively. In the former category, we can mention the volume and surface (i.e. 3D counterparts of the 2D area and perimeter, respectively), as well as dimensions of the bounding box or the convex hull. Examples of scale-invariant attributes are distributions of grey levels (e.g. standard deviation, entropy), measures computed with moments of inertia (e.g. compactness, sphericity), or moment invariants (e.g. elongation, flatness). The interested reader is referred to (Salembier and Wilkinson 2009) for more details.

We provide below a formal definition of the three attributes that have been used in the experiments reported in this paper:

- **Height** (noted A_h):
$$A_h(\mathscr{L}_{l,c}) = \max_{i,j,k}(k) - \min_{i,j,k}(k) \tag{6.8}$$
- **Volume** (noted A_v):
$$A_v(\mathscr{L}_{l,c}) = |(i,j,k)| \tag{6.9}$$
- **Extent** (also named "fill ratio" (Hernández 2009), noted A_e):
$$A_e(\mathscr{L}_{l,c}) = \frac{A_v(\mathscr{L}_{l,c})}{A_v(\mathscr{B}(\mathscr{L}_{l,c}))} \tag{6.10}$$

with $\mathscr{B}(\cdot)$ the bounding box of a set. Let us note that for the sake of conciseness, the condition $(i,j,k) \in \mathscr{L}_{l,c}$ was systematically omitted in the right part of the previous equations.

The aforementioned attributes are either increasing or non-increasing, depending if their value is increasing from leaves to root or not. The filtering simply consists in assessing each connected component by comparing its attribute value to a given threshold T, and retaining only the filtered set $\mathscr{L}' \subseteq \mathscr{L}$ defined as

$$\mathscr{L}'_l = \left\{ \mathscr{L}_{l,c} \,|\, A\left(\mathscr{L}_{l,c}\right) \geq T \right\} . \tag{6.11}$$

While the filtering with an increasing attribute is achieved through pruning the tree (i.e. removing all descendant nodes of $\mathscr{L}_{l,c}$ if $A(\mathscr{L}_{l,c}) < T$), considering a non-increasing attribute leads to pruning and non-pruning strategies (Salembier, Oliveras, et al. 1998; Urbach and Wilkinson 2002) that remove full branches or isolated nodes, respectively.

The final step of the filtering is to reconstruct the filtered volume F based on remaining

nodes of the max-tree, i.e.:

$$\begin{aligned} F : \quad & {}_h \longrightarrow \mathbb{R} \\ & (i,j,k) \longmapsto \max_{l \in \mathbb{R}} \left((i,j,k) \in \mathscr{L}'_l \right) . \end{aligned} \tag{6.12}$$

6.3.3. Reprojection to the 3D point cloud

After having performed attribute filtering (or any other morphological processing) on the 3D volume, the filtered volume $F(i,j,k)$ needs to be reprojected in the original continuous set of coordinates to produce a dataset $\{x,y,z,\mathscr{F}\} \in \mathbb{R}^3 \times \mathbb{R}$. To this end, we assign to all initial points embedded in each voxel (i,j,k) the value $F(i,j,k)$. Let us note that more complex functions could have been considered here (e.g. interpolation taking into account the position of (x,y,z) in (i,j,k) and intensities of the neighboring voxels). Our choice mathematically reads as applying the inverse transformation function $\mathscr{RP}$ (for "voxels to points") defined as:

$$\begin{aligned} \mathscr{RP} : \quad & {}_h \times \mathbb{R} \longrightarrow \mathbb{R}^3 \times \mathbb{R} \\ & \{i,j,k,F(i,j,k)\} \longmapsto \{x,y,z,\mathscr{F}\} \text{ with:} \\ & \begin{cases} (x,y,z) = (x_e,y_e,z_e) \in \ \text{s.t.} \\ \qquad \begin{cases} x_m + ih \le x_e < x_m + (i+1)h \\ y_m + jh \le y_e < y_m + (j+1)h \\ z_m + kh \le z_e < z_m + (k+1)h \end{cases} \\ \mathscr{F} = F(i,j,k) \end{cases} \end{aligned} \tag{6.13}$$

The proposed scheme allows us to consider the 3D information contained in the 3D point cloud by processing the associated volume. We will illustrate in the next section the relevance of such an approach.

6.4. Experiments

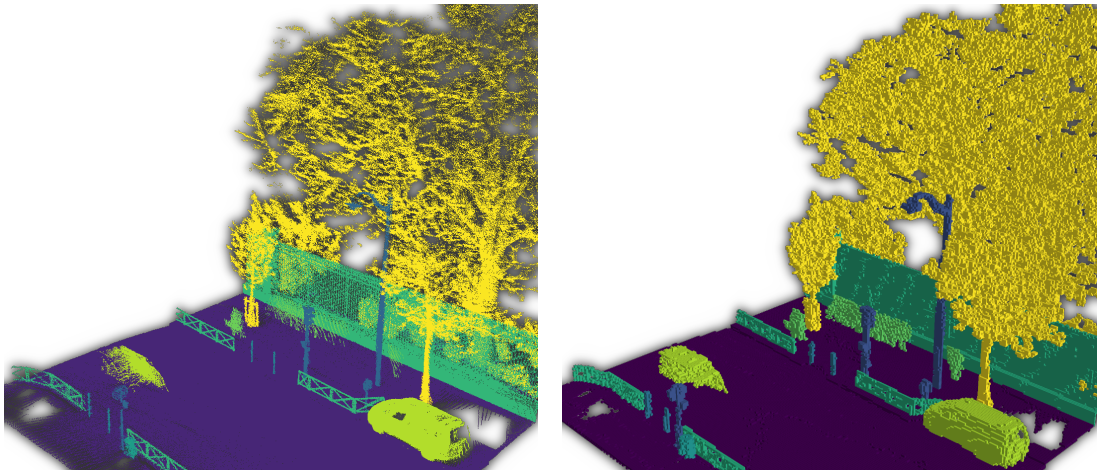
6.4.1. Dataset

Experiments have been carried out on the Paris Lille 3D dataset (Roynard et al. 2017). The LiDAR tiles considered here have been acquired by a MLS on a street of Paris. As the acquisition source is close to the ground, the point density varies greatly according to

the distance of the scanned object (as a consequence, density based max-tree will tend to remove distant objects). However the point cloud density of this dataset is significantly high (between $1,000$ to $2,000$ per square meters), which gives flexibility in the choice of the spatial resolution h in the voxelization process. Additional data is available, with LiDAR intensity return and label associated with each point in the cloud.

6.4.2. Experimental setup

As a first experiment, we have chosen to filter the labelled point cloud, which is illustrated in Figure 6.2a. For this point cloud we have fixed the step of the voxel grid to $h = 10$ cm. The labels are given with the dataset and ordered as follows: void (value 0), unclassified (1), ground (2), road sign and traffic light (4), bollard (5), trash can (6), barrier (7), pedestrian (8), car (9) and vegetation (10). This order has been used to construct max-tree on the $[0, 10]$ value range. We have represented for each cell of the grid the majority-class of the points (Figure 6.2b).



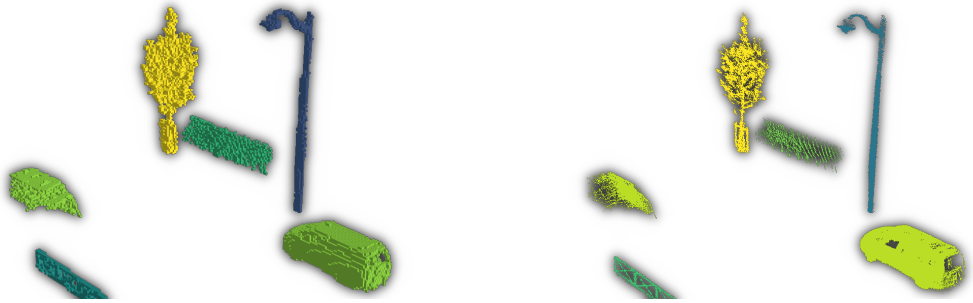
(a) Point cloud (b) Voxel grid

Figure 6.2.: Visualization of the valued point cloud and the corresponding voxel grid. The 5 classes are represented as follows: road in purple, cars in green, fences in teal, trees in yellow and urban furniture in blue.

We have built the max-tree considering 26-connectivity (i.e. two voxels (i, j, k) and (i', j', k') are neighbors if $\max(|i - i'|, |j - j'|, |k - k'|) = 1$). The tree is augmented with spectral features such as mean grey level and standard deviation and also with spatial features such as the volume, the bounding box and several geometric ratios. During the filtering process, we used the direct non-pruning strategy for the sake of simplicity and

to retrieve all the objects corresponding to the required description on non-increasing criteria (extent in our case). Then the filtered max-tree is transformed back into a 3D volume and reprojected into a point cloud (therefore function F in (6.13) is the label value).

With this experiment we were able to interactively filter objects from the max-tree with geometric object attributes (e.g. volume, height, compactness) in order to choose the appropriate threshold values. The illustration given in Figure 6.3a is the voxel grid result of a filtering with the volume criterion set as $1,000 < A_v \leq 5,000$. We then transfer the filtered result back into the point cloud (Figure 6.3b).

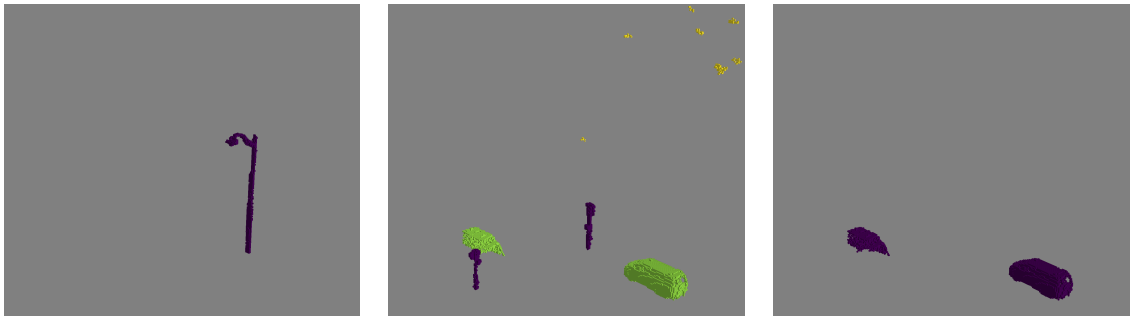


(a) Filtered voxel grid (b) Filtered point cloud

Figure 6.3.: Visualization of the filtered voxel grid from (Fig. 6.2b) with an area criterion (a) and the reprojection into the point cloud (b).

An additional example of attribute filtering is given in Figure 6.4. We can observe the relevance of the height attribute to extract tall objects (e.g. the lamp post is the only object with a height comprised between 10 and 13 meters, Figure 6.4a). Considering the extent attribute with value between 0.14 and 0.16 allows us to highlight road signs, cars and a few branches, as shown in Figure 6.4b. Finally, it is possible to combine multiple attribute for a more precise filtering, e.g. objects with height between 1.5 and 3 meters, a volume greater than $1,000$ and an extent between 0.14 and 0.16 match cars in Figure 6.4c.

Among the current limitations of the proposed approach, we have noticed that some small blocking artifacts were appearing at the border between two objects in the point cloud (see close-up view in Figure 6.5). These artifacts are directly linked with the grid



(a) Height filtering (b) Extent filtering (c) Multi-attribute filtering

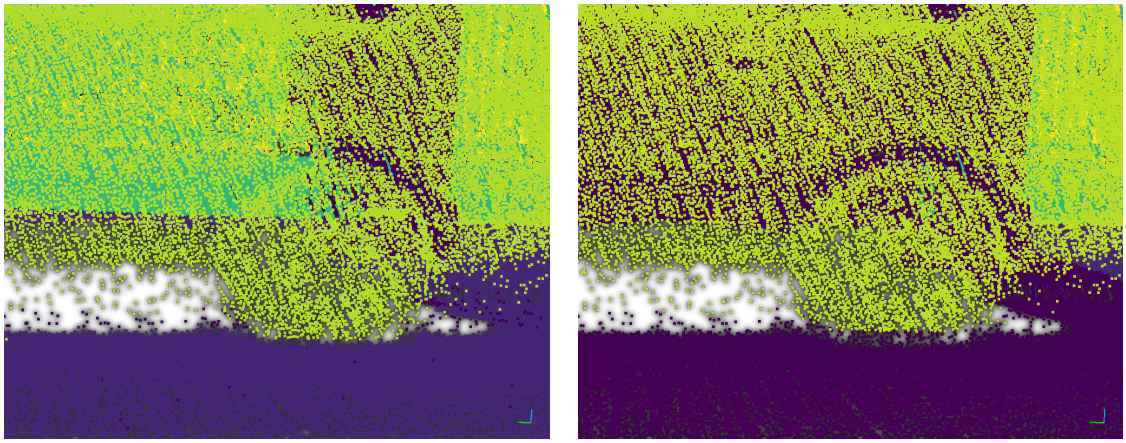
Figure 6.4.: Different attribute filters on V: connected components characterized by (a) $A_h \in [100, 130]$, (b) $A_e \in [0.14, 0.16]$, (c) $A_h \in [15, 30]$, $A_v > 1,000$, and $A_e \in [0.14, 0.16]$.

discretization method and depend on the voxel size. In our experiments, we observe that artifacts remain small with $h = 10$ cm.

The previous experiments showed that attribute filtering is useful to filter a labeled point cloud. The labels are either defined by visual expert analysis or by automatic classification of the raw LiDAR data. It is also possible to filter directly such raw data. We illustrate some results of preliminary experiments with LiDAR intensity in Figure 6.6. We can see here the relevance of attribute filters to remove the noise in the point cloud. Indeed, the noise can be easily characterized by geometric attributes (e.g. small volume nodes correspond to points disconnected from the rest of the scene, see Figure 6.6a). It is also relevant to remove extreme intensity values (i.e. outliers) with the max-tree.

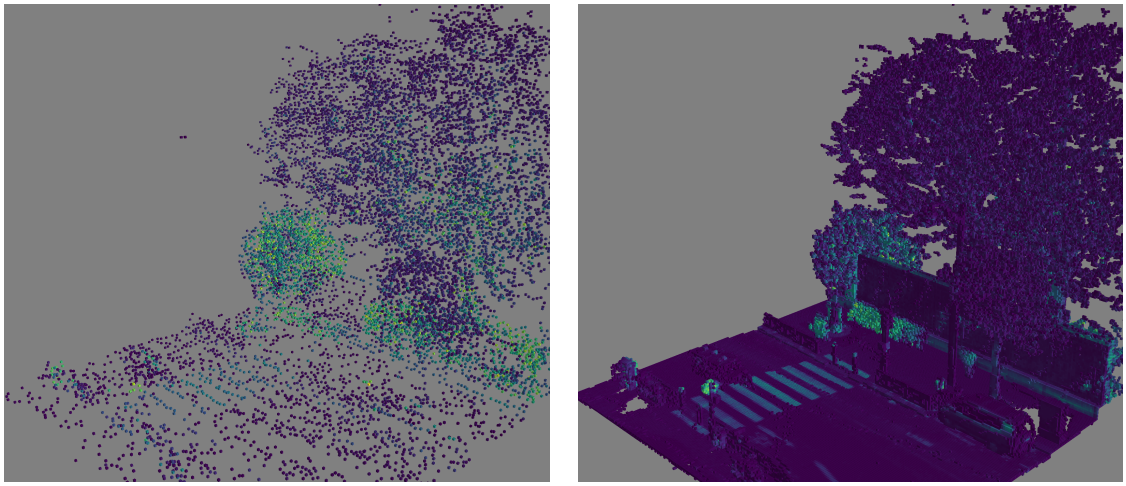
6.5. Conclusion

While most approaches for applying mathematical morphology on point clouds are relying on a prior rasterization step into a 2D image, we explore here a different strategy. Indeed, we rather suggest a discretization of the space into voxels to build a 3D volume instead of a 2D image. This choice is motivated by the straightforward extension of morphological operators (including hierarchies) to nD data considering a connectivity of higher dimension. It allows us to benefit from a richer hierarchical representation where each node contains a set of voxels from which advanced features can be computed. We illustrate the relevance of such a framework with the popular attribute filtering, that is applied here on the voxel hierarchy before reconstructing a filtered point cloud. The results obtained on an urban point cloud show the performance of the proposed strategy,



(a) Initial point cloud (b) Filtered point cloud

Figure 6.5.: Close-up of the area where an object is in contact with the ground. In the filtered point cloud (b), blocking artifacts may appear at the boundary between objects (in this image part of the wheel is "blocked" in the road.



(a) Noise (b) Denoised

Figure 6.6.: Denoising of raw LiDAR data: outliers and isolated points are removed from the scene using the volume attribute: (a) $A_v < 2$ identifies noise removed from the scene, (b) $A_v \geq 2$ performs the denoising.

that goes far beyond the standard 2D processing usually considered in the literature.

Future work will include extending our proposal to other hierarchical models (e.g. min-tree, tree of shapes) as well as considering some other tree-based tools beyond the standard attribute filtering. Note than an alternative could be to a create nearest neighbour graph representation from the raw set of points and perform filtering using max-tree on this graph. Besides, multiscale description with attribute profiles and data segmentation with hierarchical cuts (to name a few) would be of great interest to deal with point clouds such as those considered in remote sensing. Finally, the blocking artifacts discussed in the paper call for some further studies.

VOXEL-BASED ATTRIBUTE PROFILES ON LIDAR DATA FOR LAND COVER MAPPING

DOI 10.1109/IGARSS.2019.8899129
Authors Florent Guiotte, Sébastien Lefèbre and Thomas Corpetti.
Keywords LiDAR data, relation network, full waveform and land cover mapping.

Objectives *This work use the framework defined in Chapter 6 to classify a point cloud using 3D APs.*
We re-organise the complex 3D LiDAR point clouds into regular 3D voxel grids. We build the max-tree of the 3D voxels to compute multi-scale features with the APs. We then train a supervised classifier to predict urban classes on the Paris Lille dataset.

Results Unlike previous rasterization where the vertical component of the point clouds is lost, the APs of voxelized LiDAR data allow to compute efficient 3D multi-scale descriptors.
This method allows to compute efficient 3D multi-scale descriptors to perform supervised classification of LiDAR point clouds.

Contents

Abstract This paper deals with strategies for LiDAR data analysis. While a large majority of studies first rasterize 3D point clouds onto regular 2D grids and then use 2D image processing tools for characterizing data, our work rather suggests to keep as long as possible the 3D structure by computing features on 3D data and rasterize later in the process. By this way, the vertical component is still taken into account. In practice, a voxelization step of raw data is performed in order to exploit mathematical tools defined on regular volumes. More precisely, we focus on attribute profiles that have been shown to be very efficient features to characterize remote sensing scenes. They require the computation of an underlying hierarchical structure (through a max-tree). Experimental results obtained on urban LiDAR data classification support the performances of this strategy compared with an early rasterization process.

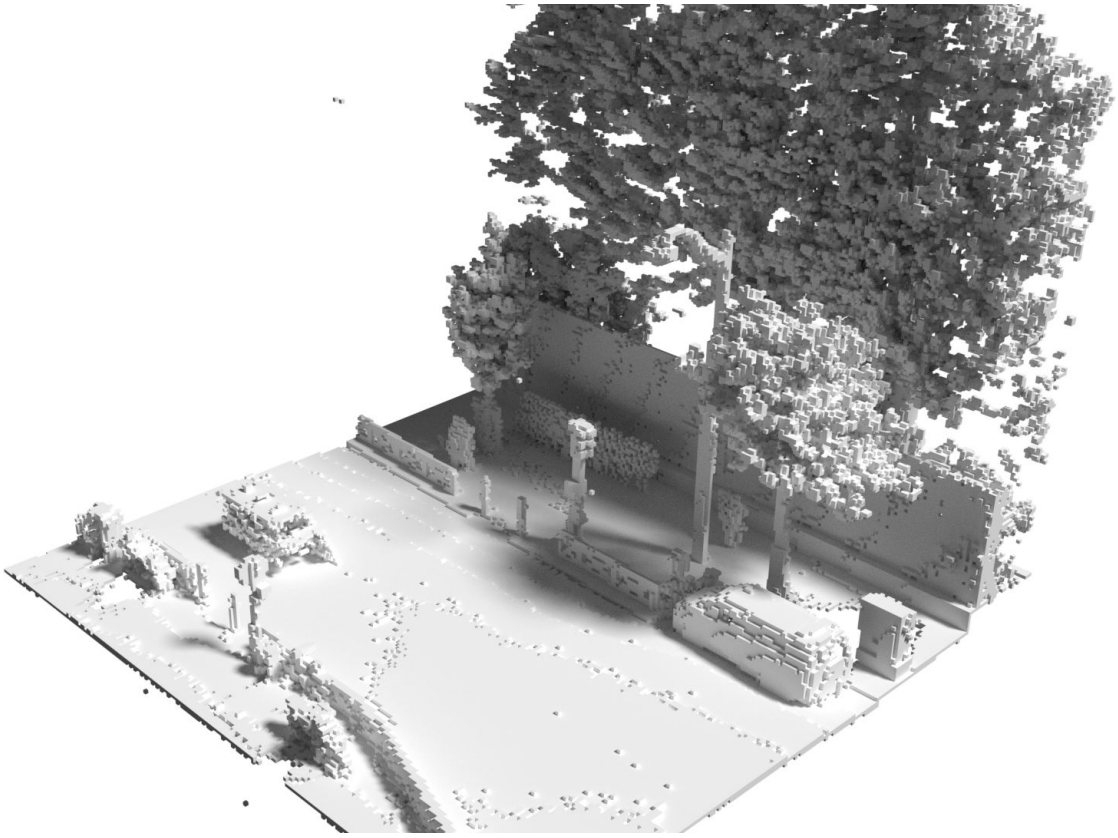


Figure 7.1.: LiDAR data represented within 3D voxel space: one can notice the information richness contained in the vertical distribution which is usually lost during the rasterization phase.

7.1. Introduction

Since a decade, LiDAR acquisitions are more and more exploited in a large variety of fields such as geosciences (flow, erosion, rock deformations, …), computer graphics (3D reconstruction), urban environments analysis and of course Earth Observation (detection of trees, roads, buildings, …).

Unlike images defined on regular 2D grids for which a large number of computer vision and image processing techniques are available, the non regular domain on which 3D point clouds are defined requires the conception of alternative and dedicated approaches. In addition, the multi-scale aspect of structures embedded in 3D LiDAR point clouds calls for the use of multi-scale techniques.

Despite relevant results, the direct projection of raw data on a 2D grid prevents us from analysing into details the complete 3D structures. Conversely, it is possible to

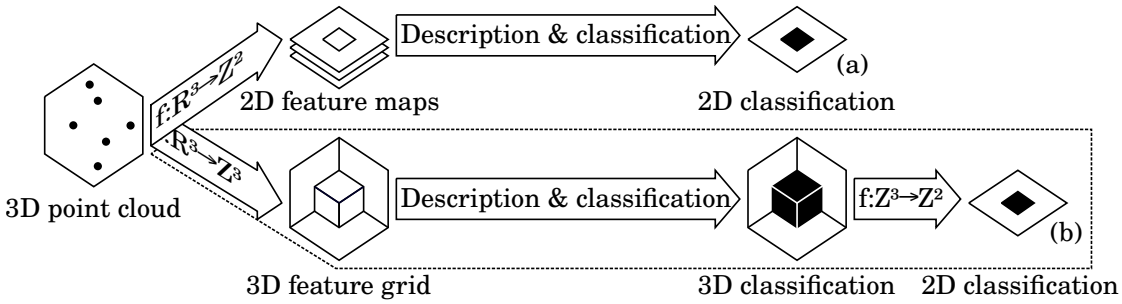


Figure 7.2.: Different LiDAR data classification approaches: (a) usual methods with 2D rasterization process beforehand, (b) proposed method with postponed 2D rasterization.

analyze the 3D point clouds by computing dedicated 3D features and then project these structures on 2D regular grids. We explore such strategy in this paper, since it provides rich 3D features as shown in Figure 7.1. For the sake of illustration, we focus here on attribute profiles that are popular feature in remote sensing. These features are extracted through morphological hierarchies that provide a rich and efficient multi-scale analysis framework. The overall process is schematized in Figure 7.2.

7.1.1. Attribute Profiles

Morphological attribute profiles are well-established pixel features that embed spatial information (Dalla Mura, Benediktsson, Waske, et al. 2010). These features are built from the successive application of attribute filters, which aim to simplify the image by removing some of its connected components that do not fulfill a given criterion. More precisely, each connected component of the level sets is characterized by some attributes, and the simplification (or filtering) is achieved by comparing each attribute value to a predefined threshold. This framework is particularly appealing due to its high efficiency that is ensured by the underlying tree structure (or hierarchical image representation). Processing tree nodes instead of raw pixels leads to a severe decrease in terms of computational cost, allowing this framework to be relevant even for large-scale studies. Since their introduction in (Dalla Mura, Benediktsson, Waske, et al. 2010), attribute profiles have been widely used and several recent extensions have been proposed to strengthen their expressivness (Pham, Lefèvre, et al. 2018) (e.g. we will use differential APs made of differences between successive values in the profile). We will use these features to characterize 2D or 3D structures.

7.1.2. Point Clouds and Mathematical Morphology

The study in (Calderon and Boubekeur 2014) is interesting since authors perform mathematical morphology on points in a sound continuous mathematical framework. Their idea consist in computing a surface from the point cloud on which erosion/dilation and associated morphological features can be computed. Though very relevant, this approach can not be applied with multi-echo LiDAR data since surfaces can not be computed. Indeed, in this context we not only extract points related to the boundaries of objects but in some situations (vegetated areas for example), some points can be included inside embedding shapes. Therefore the reconstruction of a single surface for each object is impossible.

For that reason, many authors have tried to compute morphological features on regular 3D data (volumes) defined on voxels. Hence in (Gorte and Pfeifer 2004) the authors extend 2D morphological algorithms to 3D voxels for LiDAR point clouds of trees. The density of points in each cell is kept for the value associated with each voxel, as in (Ferdosi et al. 2010) for astronomical applications. In (Serna, Marcotegui, and Hernández 2016), the same principle is used on binary 3D images (1 if a point is inside the voxel, 0 elsewhere) for the segmentation of facades from urban point clouds, using morphology on DEM with specific filters related to elongation.

In a connected idea, the study in (Aijazi et al. 2013) performs segmentation and classification of urban point clouds with super-voxels. Such super-voxels are created on the basis of raw data (intensity, R, G, B, ...) and specific features (mean, variance of raw data, geometric organization, ...). Results are then re-projected by associating the same value to each point inside a voxel.

7.1.3. Trees and 3D images

Max-trees have been introduced for 2D data in (Salembier, Oliveras, et al. 1998). They have then been extended in (Wilkinson and Westenberg 2001) to 3D voxels on medical images. Such trees are exploited for filtering objects on the basis of inertia attributes. In a similar way, the authors in (Westenberg et al. 2007; Kiwanuka, Ouzounis, et al. 2009; Kiwanuka and Wilkinson 2012) have proposed new 3D geometric attributes to isolate more complex structures (elongation, flatness, sphericity, roundness, sparseness, ...). Therefore a large variety of studies of been proposed based on mathematical morphology on voxels in the medical image community (Dufour et al. 2013; Grossiord et al. 2015; Urien et al. 2017; Padilla et al. 2018).

Surprisingly only few studies have been made in the EO context despite the fact that

3D data become here more and more important.

7.2. Method

Following the bottom of Figure 7.2, the overall methodology consists in: *i)* building a volume composed of regular voxels from the 3D point cloud; *ii)* computing 3D morphological features on this volume and using these features in a data analysis process (e.g., classification, clustering, filtering, …); and *iii)* projecting the results back into a 2D grid. These steps are described below.

7.2.1. From LiDAR point clouds to 3D rasters

The voxelization process consists in putting in each voxel of the volume the information related to the 3D points embedded in this voxel. Depending on the application, such information can be a binary map related to the presence/absence of points, the number of LiDAR points into the voxel, the average/standard deviation of associated intensities, the average/standard deviation of associated elevations, the label in majority (for 3D labels), …

7.2.2. 3D hierarchical representations

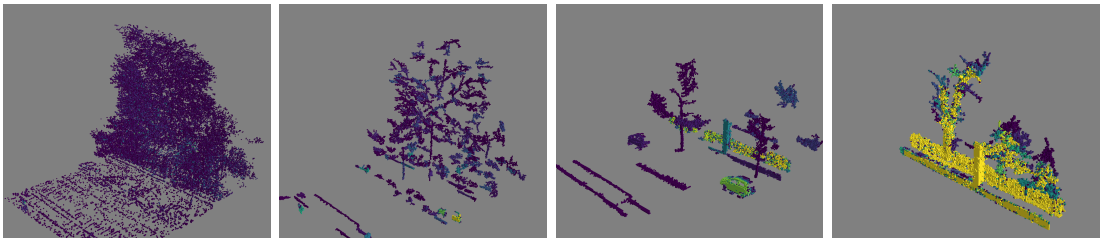
Construcing min- and max-trees is then achieved following methods introduced in Section 7.1.3. We consider here a 26-connectivity (i.e. two voxels are neighbors if they share a plane, edge or vertex).

7.2.3. AP on 3D data

As already stated, we illustrate here the voxelization process with APs which have been introduced in Section 7.1.1. For the sake of simplicity, we associate to each voxel a scalar value on which an ordering relation (required to build the tree) can be straightforwardly defined. We consider several attributes: intensity distribution, volume, surface, bounding box, and moment-based. These attributes are then compared to a set of thresholds to filter the max-tree and then to generate the APs.

7.2.4. From 3D to 2D maps

In order to project the 3D information (defined on (X, Y, Z)) in 2D grids (defined on (X, Y)), one needs to transform volumes into maps and to summarize the vertical infor-



(a) $0 < \Lambda \leq 10$ (b) $50 < \Lambda \leq 100$ (c) $500 < \Lambda \leq 1000$ (d) $10000 < \Lambda \leq 20000$

Figure 7.3.: Visualisation of the hierarchical representation involved with APs using the Derivative of Attribute Profiles (DAPs).

mation in axis Z. Different strategies are possible depending on the data and are listed below:

Elevation rules:
1. Surface model: points with higher altitude in the vertical dimension (i.e. higher objects as trees or buildings appear first);
2. Terrain cover: points with lower altitude in the vertical dimension (i.e. ground and streets appear first);

Label rules:
3. Priority rule: labels are kept depending on their priority (e.g. cars first, then trees, then roads, ...);
4. Majority rule: the majority label in the vertical axis is kept;

Value rules:
5. Average/standard deviation of each value in the vertical axis.

7.3. Experiments

7.3.1. Dataset

Experiments have been conduced on the Paris Lille Dataset (Roynard et al. 2017), which is made of a very high spatial resolution point cloud (approx 1000 ~ 2000 point per m^2) available online. As the laser scan is performed from a car in the middle of the street, the density of points depends mainly on the distance with respect to the car but not on the nature of objects. Therefore, this quantity can not be taken into account.

As for the intensity, which is usually of poor quality in such data, most of associated values are constant. Similarly to all LiDAR data, the intensity value also depends on the distance of objects w.r.t the scanner. Average intensities and elevations are the two information kept in each voxel. The scene has been classified in 5 objects: cars, street, fences, trees and urban furniture.

7.3.2. Experimental setup

In each experiment, the volumes have been computed on a voxel grid of $10cm^3$ spatial resolution. APs (2D or 3D) have been computed with area and volume criteria. The thresholds have been empirically set to $\Lambda_V = \{10, 50, 100, 200, 500, 1000, 1 \times 10^4, 2 \times 10^4, 1 \times 10^5\}$ for volume and $\Lambda_A = (\sqrt[3]{\Lambda_V})^2$ for area. A first set of experiments related to 3D classification (with generic name *TD* for "Tri-Dimensions") has been conducted to evaluate the benefits of 3D AP on 3D data: i) with intensity (noted TD_i); ii) with intensity and elevation (TD_{ie}); iii) with APs built on intensity (TD_{Ai}); iv) with APs built on intensity and elevation (TD_{Aie}).

Then, a second set of experiments is devoted to the evaluation of 3D AP features for 2D classification (with generic name *BD* for "Bi-Dimension"). To this end, we keep the best 3D features for 3D classifications (issued from the first series) and we compare their performances in 2D and 3D versions for 2D classification. 3D labels are reprojected on 2D grids based on the majority label. Classification have been performed using Random Forest technique. Some illustrations of hierarchical representations using Differential-AP with various thresholds of volumes are visible in Figure 7.3. Validation criteria rely on Overall Accuracy and Cohen's Kappa coefficient. During 3D to 2D projection, the majority label in the vertical component has been kept. In practice, training and validation data have been taken from various tiles of the dataset.

7.3.3. Results

Voxel grid classification

Evaluation measures for 3D classification are reported in Table 7.1. One can see that the elevation feature enables to improve the classification with intensity only, and that the use of APs enables to improve the overall accuracy of the classification (the overall accuracy and kappa of TD_{Ai} –resp. TD_{Aie}– is higher than the ones of experiment TD_i –resp. TD_{ie}). The best combination TD_{Aie} provides really satisfying results and an illustration of classification is depicted in Figure 7.4. This first set of experiments

	Description method	OA(%)	κ(×100)
TD_i	Intensity	75.28	48.80
TD_{ie}	Intensity and elevation	91.10	82.92
TD_{Ai}	Intensity APs	78.82	58.63
TD_{Aie}	Intensity APs and elevation	**93.66**	**87.99**

Table 7.1.: Overall accuracy (OA) and Cohen's Kappa coefficient (κ) in 3D voxel grid space, with and without AP.

	Description method	OA(%)	κ(×100)
BD_{2DAie}	Intensity and elevation APs 2D	98.25	96.61
BD_{3DAie}	Intensity APs and elevation 3D to 2D	**99.36**	**98.76**

Table 7.2.: Overall accuracy (OA) and Cohen's Kappa coefficient (κ) in 2D classification with the best features issued from Table 7.1 in 2D and 3D versions.

justifies the use of APs and Random Forest for 3D data classification. Let us now evaluate the benefits of 3D features for 2D classification.

Pixel grid classification

In this second series of experiments, 2D classifications have been performed on the basis of the best 3D features in 2D and 3D versions. More precisely, we have computed 2D classification with APs computed on 2D intensity and elevation (noted $\boldsymbol{BD_{2DAie}}$) and 2D classification with APs computed on 3D intensity and elevation and 2D reprojected in the pixel grid (noted $\boldsymbol{BD_{3DAie}}$). Quantitative results are given in Table 7.2. It is interesting to observe that as expected the generation of 2D maps on the basis of 3D features enables to improve the quality (both in terms of overall accuracy and kappa) of the 2D classifications, which is a really interesting property. An illustration is visible on Figure 7.5.

7.4. Conclusion

In this paper, we have proposed to classify LiDAR data on the basis of 3D features. Unlike most studies that rasterize the 3D point cloud in a first step, we have suggested to rather voxelize the 3D point cloud, before computing some hierarchical features on this volume and perfoming a subsequent supervised classification. The rasterization is

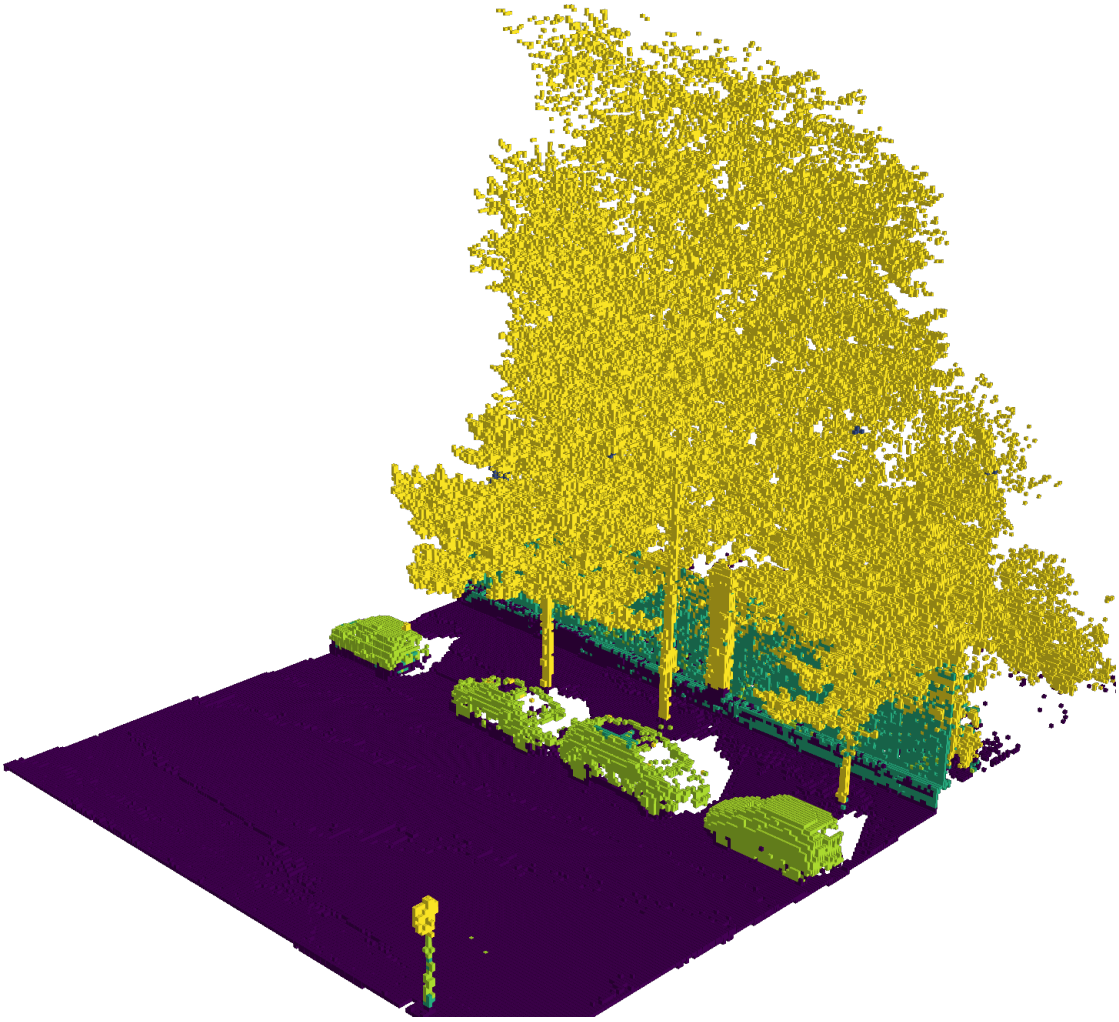


Figure 7.4.: Classification of the scene in the 3D voxel grid space using intensity data with APs and elevation information.

performed only in a final step. This process has the advantage to keep the 3D information of structures along the process instead of removing this information through early rasterization. The experimental results on 3D urban LiDAR data have quantitatively demonstrated the relevance of such an approach.

7.5. Acknowledgment

The authors would like to thank the authors of Paris Dataset (ROYNARD et al. 2017) for providing this public dataset.

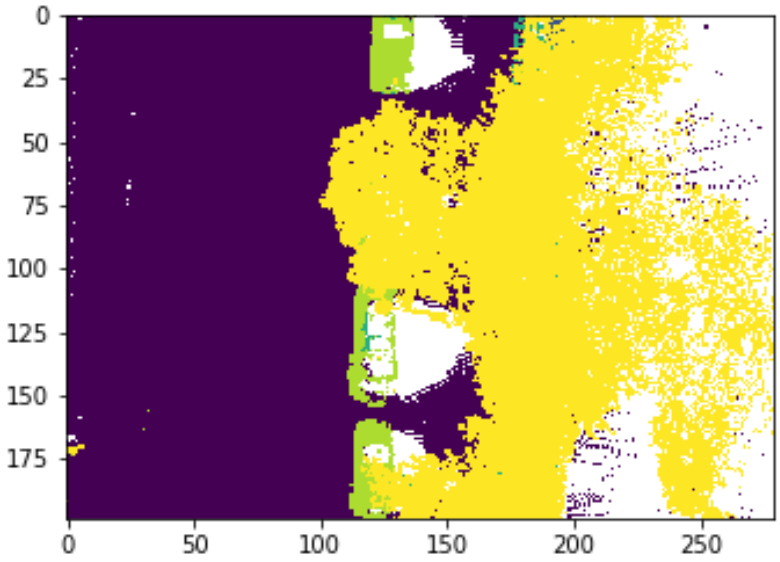


Figure 7.5.: 2D classification result obtained from Figure 7.4 voxel grid reprojected into the 2D map space using the surface rule (see 7.2.4).

RELATION NETWORK FOR FULL-WAVEFORMS LIDAR CLASSIFICATION

DOI 10.5194/isprs-archives-XLIII-B3-2020-515-2020

Authors Florent Guiotte, Meng Bin Rao, Sébastien Lefèvre, Ping Tang and Thomas Corpetti.

Keywords LiDAR data, relation network, full waveform, land cover mapping

Objectives *This work proposes to use a deep neural network designed for hyper-spectral on LiDAR pseudo-waveforms to provide a 2D land-cover map.* The LiDAR point cloud is re-organised in a 3D grid with a high resolution on the vertical axis. Pseudo-waveform are created along the vertical axis. The generated pseudo-waveforms are used as input in a neural network initially tailored for hyperspectral images.
This work was made in collaboration with the Chinese Academy of Science in Beijing.

Results The ortho-waveforms are easily processed using the 3D grids. The ortho-waveform grid is similar to an hyperspectral image cube. The deep neural network designed to extract spectral information is able to extract meaningful spatial relation.
We report good classification results by taking into account the vertical component.

Contents

Abstract LiDAR data are widely used in various domains related to geosciences (flow, erosion, rock deformations, etc.), computer graphics (3D reconstruction) or earth observation (detection of trees, roads, buildings, etc.). Because of the unstructured nature of remaining 3D points and because of the cost of acquisition, the LiDAR data processing is still challenging (few learning data, difficult spatial neighboring relationships, etc.). In practice, one can directly analyze the 3D points using feature extraction and then classify the points via machine learning techniques. In addition, recent neural network developments have allowed precise point cloud segmentation, especially using the seminal pointnet network and its extensions. Other authors rather prefer to rasterize / voxelize the point cloud and use more conventional computers vision strategies to analyze structures. In a recent work, we demonstrated that Digital Elevation Models (DEM) is reductive of the vertical component complexity describing objects in urban environments. These results highlighted the necessity to preserve the 3D structure of the point cloud as long as possible in the processing. In this paper, we therefore rely on ortho-waveforms to compute a land cover map. Ortho-waveforms are directly computed from the waveforms in a regular 3D grid. This method provides volumes somehow "similar" to hyperspectral data where each pixel is here associated with one ortho-waveform. Then, we exploit efficient neural networks adapted to the classification of hyperspectral data when few samples are available. Our results, obtained on the 2018 Data Fusion Contest dataset (DFC), demonstrate the efficiency of the approach.

8.1. Introduction

Because of their ability to capture complex structures, many domains related to geosciences and earth observation are making increasing use of LiDAR data. Such systems provide indeed accurate 3D point clouds of the scanned scene which has a large number of applications ranging from urban scene analysis (Chehata et al. 2009; Guiotte, Pham, et al. 2020; Shan and Aparajithan 2005), geology and erosion (Brodu and Lague 2012), archaeology (Witharana et al. 2018) or even ecology (Eitel et al. 2016).

However, the processing of such data is not obvious since unlike N-dimensional images, the spatial irregular distribution of the point clouds makes tricky (both from a theoretical and computational point of view) the computation and use of spatial features. Moreover, though efficient recent neural network have been designed for LiDAR and unstructured point clouds (Landrieu and Simonovsky 2018; Qi, Su, et al. 2017; Qi, Yi, et al. 2017), at the moment the lack of labeled data limits the use of advanced learning techniques.

Many strategies exist to deal with this issue. While some of them directly exploit the 3D point cloud structure (Brodu and Lague 2012; Niemeyer et al. 2014; Mallet, Bretar, et al. 2011), in many applications the point cloud is first binned into a **2D regular grid** (so-called "rasterization process") on which computer vision approaches can be applied (see e.g. (Lodha et al. 2006)). While first works have been focused on the characterization of single points (often through height and intensity) without including information related to their neighbours (Lodha et al. 2006), more advanced approaches have included spatial relationships using a set of spheres or cylinders (of variable radius) around each point to extract consistent geometric features (Mallet, Bretar, et al. 2011; Weinmann et al. 2015; Niemeyer et al. 2014). Among others, we have demonstrated in (Guiotte, Lefèvre, et al. 2019c; Guiotte, Pham, et al. 2020) that the various rasterization strategies may have an important impact on the final result.

Complementary to rasterization, it is also possible to bin the point cloud into a **3D regular grid** (a.k.a. "voxelization process") where all points are processed via voxels (Gorte and Pfeifer 2004; Aijazi et al. 2013; Guiotte, Lefèvre, et al. 2019a; Serna and Marcotegui 2014) using point-to-voxels and voxels-to-point projections. This approach enables to keep the 3D structure of the data while using more conventional 3D-processing tools.

As an intermediate structuration strategy, we propose in this paper to map the point cloud into **ortho-waveform** maps. This has the advantage to provide 2D-(multi/hyper)spectral data where in each pixel, a signal corresponding to a reconstructed waveform

observed in the orthogonal direction is given. Therefore, the 3D structure is kept while one can still process 2D data, similar to hyperspectral ones. To deal with the fact that only few labeled data are in general available, we suggest to process such ortho-waveforms using neural networks adapted both to hyperspectral data and to few learning samples. To this end, the re-combinination (or pairing) of samples is an efficient approach to increase the amount of input training data. The resulting architectures are known as relation networks where multiple inputs are taken into account: one labelled sample per class (called support sample) and one query sample to be classified. The network outputs similarities between the query sample and the support sample per class. This relation network is combined with a submodule, which is designed to extract common features (similar to the prototype of each class) of multiple samples per class, for the extraction of spatial-spectral features (Rao et al. 2019) and here the classification of ortho-waveforms.

The organisation of the paper is as follows: in the next section, we present the generation of ortho-waveforms from the 3D point clouds. Then in Sec. 8.3, we present the spatial-spectral relation network used for classification. Finally, we illustrate the benefits of this approach in the experimental part in Sec. 8.4, before concluding our paper in Sec. 8.5.

8.2. Generation of ortho-waveforms

To exploit the 3D structure of LiDAR data while using 2D processing tools, we create ortho-waveforms from initial full waveforms signals. More formally, let us define:

- the LiDAR acquisitions in $\mathbb{R}^3 \times \mathbb{R}$, where each data $= \{x, y, z, \mathcal{I}\} \in$ is such that the intensity taken in location (x, y, z) is $\mathcal{I}(x, y, z)$;
- $_h \subset \mathbb{N}^2$ a 2D grid with spatial resolution h (for the sake of simplicity, we consider here isotropic resolutions but the method can be applied with anistropic ones as well);
- g_σ a 1D Gaussian filter of standard deviation σ.

For each pixel $(i,j) \in {}_h$, the associated spectrum $p(i,j)$ is computed as

$$p(i,j) = g_\sigma * {}_{\mathbf{z}}(i,j) \tag{8.1}$$

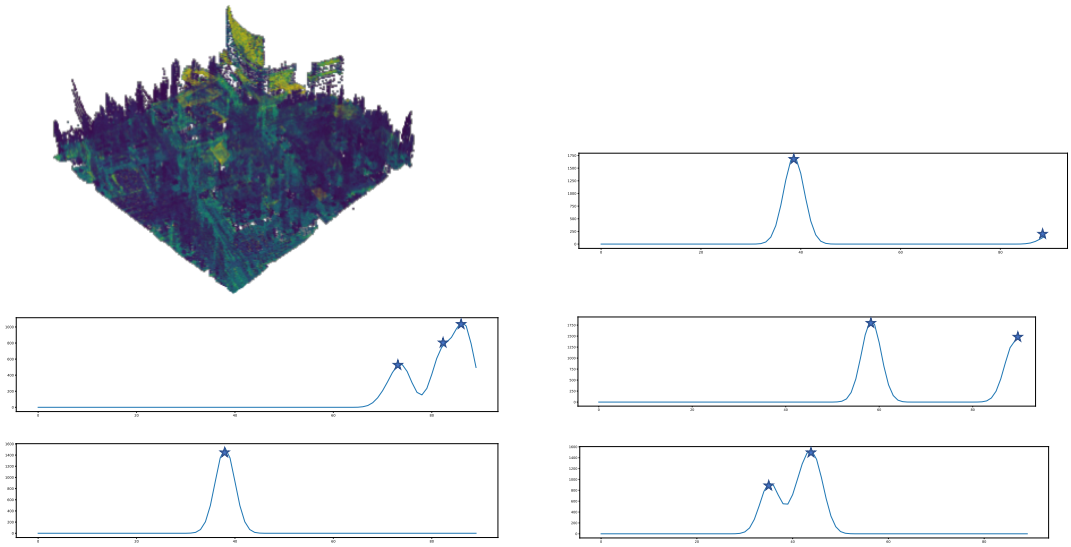


Figure 8.1.: Illustration of ortho-waveforms (blue curves) computed from raw data (top-left and star points) for 5 spatial points.

where $_{\mathbf{z}}(i,j)$ is a vector of diracs containing all vertical positions included in the spatial pixel (i,j) weighted by their corresponding intensities $\mathcal{I}$:

$$_{\mathbf{z}}(i,j) = [\mathcal{I}(x_1,y_1,z_1)\delta_{\uparrow},\ \ldots,\ \mathcal{I}(x_n,y_n,z_n)\delta_{\uparrow}] \tag{8.2}$$

with $(x_k,y_k), k \in [1,N]$ the N spatial coordinates in the point cloud included in pixel (i,j), z_k their corresponding vertical values and $\delta_{\uparrow}$ the dirac function. This provides, in each pixel, ortho-waveform data as illustrated in Fig. 8.1 where the original dataset and some waveforms are illustrated. The next section introduces the relation-network that we used to process such data.

8.3. Spatial-Spectral Relation Network

The spatial-spectral relation network (SS-RN) (Rao et al. 2019) was designed to classify hyperspectral images. Not only it learns the relation between 3D features (spectral features and spatial features) of the samples, but also it iteratively learns the similarities between a query sample and several samples per class. The overview of SS-RN is

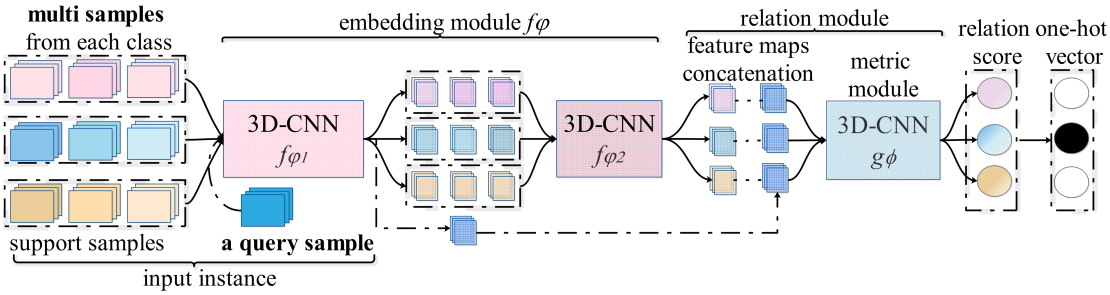


Figure 8.2.: An example of SS-RN architecture for hyperspectral image classification.

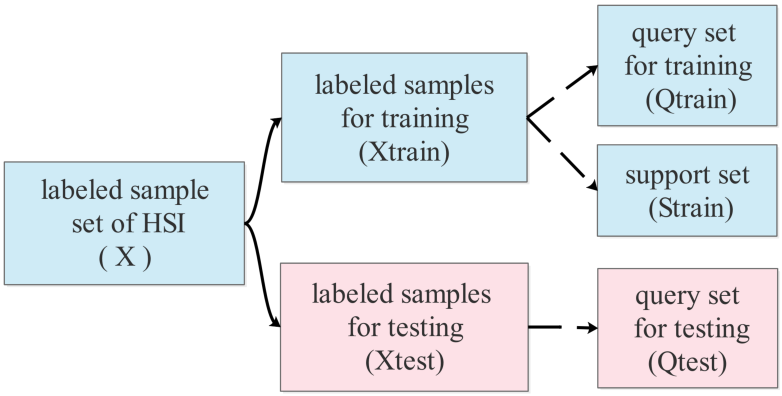


Figure 8.3.: The organization of labelled samples under the framework of SS-RN.

presented in Fig. 8.2. The SS-RN method consists of the following main parts: input construction, embedding module and relation module. In the following, we successively introduce these three parts in detail.

Multi-Support Sample Recombination

The proposed SS-RN exploits the training set by episode-based training. In each training iteration, an input instance is formed by randomly selecting one query sample and several randomly selected labelled samples (called support samples) per class. Here the query sample is the sample to be classified, and the selected support samples per class represent its class.

Consider a dataset $X = \{x_i\}_{i=1}^{N}$ in sample space $R^{d\times w\times w}$ which contains N labeled samples. Here d is the number of spectral bands, $w \times w$ is the spatial neighbouring window size. Let $y_i \in \{1, 2, \ldots, C\}$ is the class label of x_i and C is the number of classes. The organization of labelled samples under the framework of SS-RN is presented in Figure 8.3. Firstly, we split X into the training set X_{train} and the testing set X_{test} with no

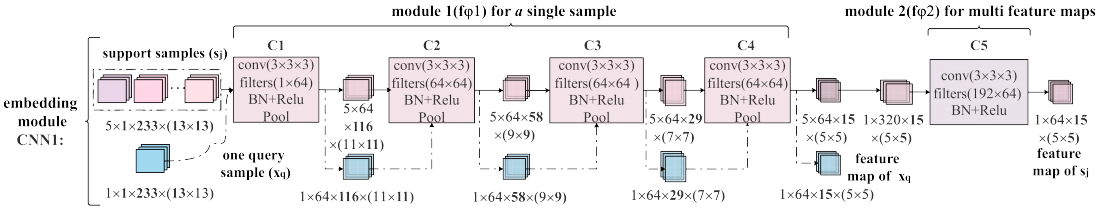


Figure 8.4.: Architecture of the spatial-spectral embedding model, which is composed of five 3D-CNN blocks. The input data here consists of five support samples per class and a query sample from the DFC2018 dataset (where $d \times w \times w = 233 \times 13 \times 13$)

intersection between these two parts. Then we construct a query set for training Q_{train}, a query set for testing Q_{test}, and a support set S_{train} defined as follows:

$$\begin{aligned} Q_{train} &\equiv X_{train} \\ Q_{test} &\equiv X_{test} \\ S_{train} &= \{S_j\}_{j=1}^{C},\ \bigcup_{j=1}^{C} S_j = X_{train} \end{aligned} \tag{8.3}$$

Here S_j contains all labeled samples of the j-th class in X_{train}. Concretely, to construct an input instance M_n^q, we randomly select a query sample x_q from a query set (Q_{train} or Q_{test}) and n support samples per class denoted as $s_j = \{x_{j1}, ..., x_{jn}\}$ from S_j. The formula of M_n^q shows in Equation 8.4 and its class label is same as the selected query sample $Label(M_n^q) = y_q$.

$$\begin{aligned} M_n^q &= [x_{11},\ \ldots,\ x_{1n},\ \ldots,\ x_{C1},\ \ldots,\ x_{Cn},\ x_q], \\ &= [s_1,\ \ldots,\ s_C,\ x_q]. \end{aligned} \tag{8.4}$$

3D Embedding Module for Feature Extraction

After constructing an input instance M_n^q, we feed M_n^q into a three-dimensional convolutional neural network (3D-CNN) as an embedding module to extract spatial-spectral features. As shown in Figure 8.4, the architecture of the embedding module consists of two sub-modules: the first sub-module $f_{\varphi 1}$ is designed to extract features of a single sample x_q or x_{jk} (the k-th support sample of the j-th class). The second sub-module $f_{\varphi 2}$ is dedicated to the extraction of common features (similar to a prototype of each class) of the input support samples per class. The whole embedding module f_{φ} can be defined

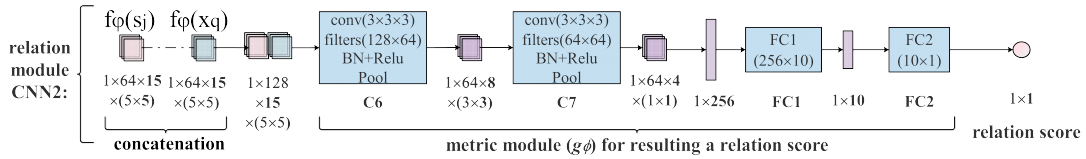


Figure 8.5.: Architecture of the relation model to measure the similarity between deep features $f_\varphi(s_j)$ and $f_\varphi(x_q)$.

by:

$$\begin{aligned} f_\varphi(M_n^q) &= f_\varphi(s_1),\ \ldots,\ f_\varphi(s_C),\ f_\varphi(x_q) \\ f_\varphi(s_j) &= f_{\varphi 2}(f_{\varphi 1}(x_{j1}),\ \ldots,\ f_{\varphi 1}(x_{jn})),\ j = 1, \ldots, C \\ f_\varphi(x_q) &= f_{\varphi 1}(x_q) \end{aligned} \tag{8.5}$$

As shown in Figure 8.4, the input instance contains five support samples each class and a query sample, and the embedding module consists of five 3D-CNN blocks. Taking the j-th class as an example, we feed five support samples s_j with size $5 \times 1 \times 233 \times 13 \times 13$ into the first sub-module $f_{\varphi 1}$, then generate five features for each support sample, thus the size of $f_{\varphi 1}(s_j)$ is $5 \times 64 \times 15 \times 5 \times 5$. The feature size of the query sample s_q extracted by $f_{\varphi 1}$ is $1 \times 64 \times 15 \times 5 \times 5$. To obtain a common feature of s_j, we feed the $f_{\varphi 1}(s_j)$ into the second sub-module $f_{\varphi 2}$, then generate a feature with size $1 \times 64 \times 15 \times 5 \times 5$. The common feature map of each class and the feature map of the query sample will be compared by the relation module.

3D Relation Module for Similarity Measurement

After the embedding module, we obtained the common features $f_\varphi(s_j) = f_{\varphi 2}(f_{\varphi 1}(s_j)),\ j \in \{1, 2, \ldots, C\}$ per class and a feature $f_\varphi(x_q) = f_{\varphi 1}(x_q)$ of the query sample. To determine the label of the query sample, we concatenate the $f_\varphi(x_q)$ with the common feature $f_\varphi(s_j)$ per class, respectively. In a second step, we feed the concatenate feature $\mathscr{C}(f_\varphi(s_j), f_\varphi(x_q))$ into a relation module, which learns to compare the query feature and a common feature per class, respectively. We then define the relation module as

$$\begin{aligned} r_{j,q} &= g_\phi(\mathscr{C}(f_\varphi(s_j), f_\varphi(x_q))) \\ &= g_\phi(\mathscr{C}(f_{\varphi 2}(f_{\varphi 1}(s_j)), f_{\varphi 1}(x_q))),\ j = 1, 2, \ldots, C \end{aligned} \tag{8.6}$$

where the symbol $\mathscr{C}$ represents the operation of feature concatenation and g_ϕ is the deep similarity metric learned by a network.

Taking the output of the embedding module as input, the architecture of SS-RN's relation module is composed of two 3D-CNN blocks and two fully-connected layers, as shown in Figure 8.5. The output of relation module is a scalar (in the range $[0, 1]$) representing the chance that x_q belongs to the j-th class, which is called the relation score. In this setting, by feeding an input instance M_n^q into SS-RN, we obtain C relation scores $r_{j,q}$, $j = 1, 2, \ldots, C$ and the query sample x_q will be classified into the class with the highest relation score.

The loss function of SS-RN is the Mean Square Error (MSE) in eq. (8.7), where M is the total number of query samples, $\{y_i\}_{i=1}^{C}$ is the label of support samples and y_q is the label of the query sample. Adam optimizer (Kingma and Ba 2017) is applied to minimize the MSE error over the training set. Note that SS-RN contains two modules (embedding module and relation module), so two Adam optimizers are employed to train the two modules respectively.

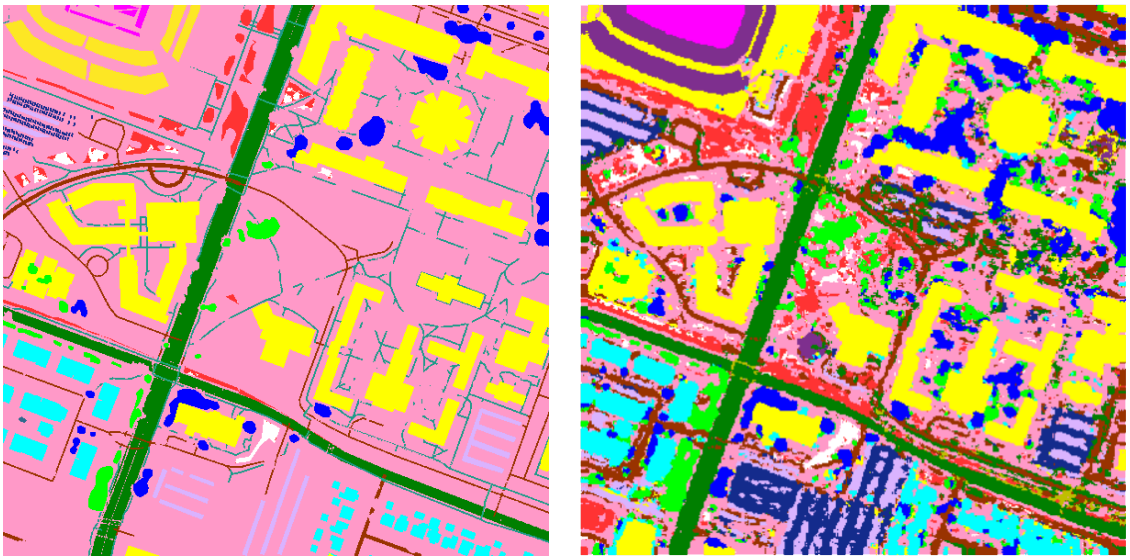
$$\varphi, \phi \leftarrow \arg\min_{\varphi,\phi} \sum_{q=1}^{M} \sum_{j=1}^{C} (r_{j,q} - 1(Label(s_j) == y_q))^2 \tag{8.7}$$

8.4. Experiments

Preliminary experiments have been performed on the dataset from the IEEE Data Fusion Contest (DFC) 2018 (Le Saux et al. 2018). To this end, we sampled the point cloud and the ground truth to $1\,\mathrm{m}^2$ resolution (vs 0.5 in the initial DFC dataset) in order to sample enough points in the vertical columns and obtain interesting ortho-waveforms. The main characteristics of the data are:

- Raw data: one LiDAR tile from DFC 2018 (mono-spectral)
- Spatial grid resolution:
  - Horizontal (x, y): 1 m
  - Vertical (z): 0.15 m
- Labels: 20 classes, some under-represented because of tiling and sub-sampling are removed.
- Train, test: 20% of the points randomly selected to train the model. Validation is performed on the rest of the dataset

Some classes non-present or under-represented in the chosen tile have been removed during the training process (cf. missing scores in Table 8.1).



(a) (b)

Figure 8.6.: Classification results on the DFC 2018 dataset. (a) ground truth and (b) : our results.

Qualitative results are presented in Figure 8.6 and quantitative evaluations are depicted in Tables 8.1 and 8.2. As can be shown, numerical experiments show very high performances despite the low number of training data, which is a good behavior of our network. However on this map, one can observe that we still have some difficulties with thin structures. Nevertheless, the overall map is consistent.

While the reported results show a very high accuracy, they have to be considered with a specific caution. Indeed, even if there is no overlap between pixels in the training and testing sets, the spatial behaviour of the CNN (through the successive increase of the receptive field) makes possible that training and testing pixels share some learnt features. The interested reader is referred to (Audebert, Le Saux, and Lefevre 2019) for an in-depth discussion of this issue that is encountered in many experiments of deep networks in remote sensing. So further experiments would be needed here to draw some final conclusions, including a larger data set and a more reliable split between training and testing sets.

8.5. Conclusion

In this work, we proposed an alternative to rasterization or voxelization strategies for LiDAR data. We suggest to keep the 3D structure of the point cloud and to create *ortho-*

Index	Label	F1 Score
0	Unclassified	–
1	Healthy grass	0.813
2	Stressed grass	0.904
3	Artificial turf	1.000
4	Evergreen trees	0.984
5	Deciduous trees	0.964
6	Bare earth	–
7	Water	–
8	Residential buildings	0.990
9	Non-residential buildings	0.994
10	Roads	0.905
11	Sidewalks	0.904
12	Crosswalks	0.529
13	Major thoroughfares	0.957
14	Highways	–
15	Railways	–
16	Paved parking lots	0.975
17	Unpaved parking lots	–
18	Cars	0.979
19	Trains	–
20	Stadium seats	0.999

Table 8.1.: Classes of the DFC 2018 along with the F1 scores.

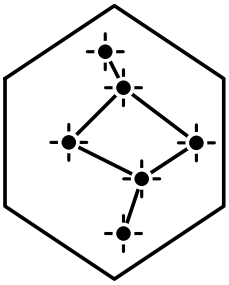
	1	2	3	4	5	8	9	10	11	12	13	16	18	20
1	935	165	0	0	4	0	0	4	53	0	1	0	0	0
2	98	3706	0	1	13	0	1	13	200	0	24	0	0	0
3	0	0	547	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	5040	16	18	37	6	7	0	0	0	0	0
5	0	12	0	2	2310	0	27	0	17	0	6	0	0	0
8	0	0	0	18	1	6374	4	7	26	0	0	1	0	0
9	0	0	0	30	39	25	38227	16	107	1	1	1	0	0
10	46	13	0	14	2	5	18	5016	237	10	243	2	2	0
11	57	216	0	17	33	20	170	143	8985	17	266	3	6	3
12	0	2	0	0	0	0	0	37	16	176	141	0	0	0
13	1	27	0	0	1	0	0	172	238	89	13727	9	0	0
16	0	0	0	0	0	0	1	37	48	0	18	3	124	0
18	0	0	0	0	0	0	0	2	2	0	3	1	855	0
20	0	0	0	0	0	1	8	0	2	0	0	0	0	5448

Table 8.2.: Confusion matrix for the 14 class used on the DFC 2018 dataset.

waveforms, resulting in rasterized data where each pixel is associated with a wavelength in the vertical direction. This has the advantage to keep both the data structure and the spatial organization in a grid. This is somehow similar to hyperspectral data and we demonstrated the efficiency of this procedure on the DFC 2018 dataset using a deep neural network initially tailored for hyperspectral data.

Part IV.

Conclusions & Perspectives



CONCLUSION

Contents

9.1. Overview

In this thesis, we have defined several discretization strategies on LiDAR data. Mapping point clouds into 2D or 3D regular grids allows us to use some of the most successful tools in remote sensing data analysis, namely morphological hierarchies and deep neural networks.

2D rasters In a first part, we re-organised the point clouds in 2D regular raster grids. This setup was used as a baseline for the rest of the thesis. One of the main advantages of this method is the opportunity to use many state-of-the-art computer vision tools suitable for large-scale processing. Notably, we used component trees from mathematical morphology to represent several rasterized LiDAR features. Component trees allow to compute efficiently multi-scale attributes. The most famous LiDAR features, namely DEMs, have shown their full potential when represented with component trees. In comparison to spectral features, which have been used extensively with morphological hierarchies in the literature, the DEMs share a strong connection between the physical elevation of structures and the level used to build the component trees. This results in attributes that were previously abstract when applied on spectral data, such as the spectral height of a node in a tree, which make physical sense on DEMs: the height of

the node in the tree is the topographic height of the physical structure. Therefore, height and volume attributes contained in the tree can be expressed with the metric system.

Component trees are the core of several useful morphological tools. In Part II we used APs for supervised classification and we used the pattern spectra to interactively analyse structures contained in LiDAR data. We also conducted segmentation of the LiDAR data using a deep learning encoder-decoder architecture on the rasterized 2D LiDAR features. However, despite exhaustive vertical description made with position of the first and last echo, the complexity but in the same time the specificity of vertical distribution of the points are substantially lost.

3D voxel grids In a second part, we went one step further and re-organised the point clouds in regular 3D voxel grids. These structures allowed to keep a precise description of the point cloud vertical component. As their 2D counterparts, the regular grids offer a straightforward definition of connectivity required for mathematical morphology. We have outlined a complete workflow to use morphological hierarchies on LiDAR point clouds, from these hierarchies we were able to filter, segment and extract multi-scale features. We then evaluated the reliability of the 3D APs descriptors on a supervised classification task. From a slightly different angle, we used a voxel grid with an increased sampling on the vertical axis, this structure allowed us to sample ortho-waveforms from the LiDAR data. To some extent, this structure is similar to hyperspectral images. We then used a deep neural network originating from hyperspectral imaging to perform a land cover classification task.

LiDAR sources have been applied to variety of challenges. Our global discretization framework for morphological hierarchies and deep neural networks is flexible and extremely efficient. A Python package SAP was developed and distributed[1]. It provides utilities to build morphological hierarchies and includes several morphological applications. SAP aims to facilitate reproducibility and dissemination of morphological tools.

9.2. Future works

This thesis opens several questions and perspectives for future research in relation to LiDAR specific trees, threshold selection, LiDAR point cloud representation and directions for deep learning on point clouds.

[1] `https://gitlab.inria.fr/fguiotte/sap`

Multi-source attributes As noted during this thesis, the potential of component tree is increased when built on DEMs. LiDAR data associate elevation and spectral information (i.e. intensity), but in its current formulation the component tree attributes are calculated on the same information that is used to build them. One can enrich the DSM tree with attribute computed on the spectral information of the LiDAR data. These attributes could be directly used on actual morphological tools: the pattern spectra for example could be used to display a 2D histogram of volume and mean infra-red of objects of the scene. The Feature Profile (FP) from Pham, Aptoula, et al. (2018) would benefit these new multi-source attributes too.

Threshold selection in attribute space APs remain very popular among remote sensing community. However we consider the threshold selection an open question. Related works on this topic only proposed attribute-wise threshold selection (Mahmood et al. 2012; Cavallaro, Falco, et al. 2017; Das et al. 2020). However, according to our work on attribute selection for class separability in attribute space (using pattern spectra Chapter 4), we have deduced that class separability involve two or more attributes. In a supervised classification framework, threshold selection (and attribute selection too) could be automated taking into account the complete attribute space of the morphological hierarchies.

Beyond 3D voxel grids Following the logical development of our approaches (see Figure 1.3 from Introduction), being able to perform morphological operations in point clouds space is appealing. Concerning morphological hierarchies, morphological filters and connected components have been generalized on arbitrary graphs (Cousty, Najman, Dias, et al. 2013). One way to use morphological hierarchies on raw LiDAR point clouds would be to build a connectivity graph of neighboring points. Nevertheless, several issues remain open. The conventional attributes do not transfer well in a continuous space. For example, the famous area attribute is based on pixel count, yet irregular density prohibits to simply count points to assess an area or a volume.

Large scale deep neural networks Concerning deep learning, recent advances achieve very promising results for point cloud classification and segmentation (Thomas et al. 2019). However few studies have been carried on large scale ALS data over forested, mountainous or dense urban area with structures of a wide variety of shapes and sizes. Maybe deep neural networks could take advantage of hierarchical representations to aggregate the global context to minute details of point clouds.

BIBLIOGRAPHY

Aijazi, A., P. Checchin, and L. Trassoudaine (Mar. 28, 2013). "Segmentation Based Classification of 3D Urban Point Clouds: A Super-Voxel Based Approach with Evaluation." In: *Remote Sensing* 5.4, pp. 1624–1650. DOI: 10.3390/rs5041624. URL: http://www.mdpi.com/2072-4292/5/4/1624 (visited on 12/17/2018) (cit. on pp. 79, 95, 105).

Asplund, T., A. Serna, B. Marcotegui, R. Strand, and C. L. Luengo Hendriks (2019). "Mathematical Morphology on Irregularly Sampled Data Applied to Segmentation of 3D Point Clouds of Urban Scenes." In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Ed. by B. Burgeth, A. Kleefeld, B. Naegel, N. Passat, and B. Perret. Vol. 11564. Cham: Springer International Publishing, pp. 375–387. DOI: 10.1007/978-3-030-20867-7_29. URL: http://link.springer.com/10.1007/978-3-030-20867-7_29 (visited on 02/17/2020) (cit. on p. 28).

Audebert, N., B. Le Saux, and S. Lefevre (June 2019). "Deep Learning for Classification of Hyperspectral Data: A Comparative Review." In: *IEEE Geoscience and Remote Sensing Magazine* 7.2, pp. 159–173. DOI: 10.1109/MGRS.2019.2912563. URL: https://ieeexplore.ieee.org/document/8738045/ (visited on 02/06/2020) (cit. on p. 112).

Audebert, N., B. Le Saux, and S. Lefèvre (June 2018). "Beyond RGB: Very High Resolution Urban Remote Sensing with Multimodal Deep Networks." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 140, pp. 20–32. DOI: 10.1016/j.isprsjprs.2017.11.011. URL: https://linkinghub.elsevier.com/retrieve/pii/S0924271617301818 (visited on 10/18/2019) (cit. on pp. 66, 69).

Badrinarayanan, V., A. Kendall, and R. Cipolla (Dec. 1, 2017). "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12, pp. 2481–2495. DOI: 10.1109/TPAMI.2016.2644615. URL: https://ieeexplore.ieee.org/document/7803544/ (visited on 10/15/2020) (cit. on pp. 18, 66).

Bartels, M. and H. Wei (July 2010). "Threshold-Free Object and Ground Point Separation in LIDAR Data." In: *Pattern Recognition Letters* 31.10, pp. 1089–1099. DOI: 10.1016/j.patrec.2010.03.007. URL: http://linkinghub.elsevier.com/retrieve/pii/S0167865510000887 (visited on 12/11/2017) (cit. on p. 21).

Belgiu, M. and L. Drăguţ (Apr. 2016). "Random Forest in Remote Sensing: A Review of Applications and Future Directions." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 114, pp. 24–31. DOI: 10.1016/j.isprsjprs.2016.01.011. URL: https://linkinghub.elsevier.com/retrieve/pii/S0924271616000265 (visited on 11/14/2020) (cit. on p. 17).

Bentley, J. L. (Sept. 1975). "Multidimensional Binary Search Trees Used for Associative Searching." In: *Communications of the ACM* 18.9, pp. 509–517. DOI: 10.1145/361002.361007. URL: https://dl.acm.org/doi/10.1145/361002.361007 (visited on 11/22/2020) (cit. on p. 17).

Bhardwaj, K., S. Patra, and L. Bruzzone (2019). "Threshold-Free Attribute Profile for Classification of Hyperspectral Images." In: *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–12. DOI: 10.1109/TGRS.2019.2916169. URL: https://ieeexplore.ieee.org/document/8734832/ (visited on 08/12/2019) (cit. on p. 27).

Boser, B. E., I. M. Guyon, and V. N. Vapnik (1992). "A Training Algorithm for Optimal Margin Classifiers." In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 144–152 (cit. on p. 17).

Bosilj, P., E. Aptoula, S. Lefèvre, and E. Kijak (Dec. 2, 2016). "Retrieval of Remote Sensing Images with Pattern Spectra Descriptors." In: *ISPRS International Journal of Geo-Information* 5.12, p. 228. DOI: 10.3390/ijgi5120228. URL: http://www.mdpi.com/2220-9964/5/12/228 (visited on 09/21/2017) (cit. on pp. 28, 47).

Bosilj, P., E. Kijak, and S. Lefèvre (Feb. 1, 2018). "Partition and Inclusion Hierarchies of Images: A Comprehensive Survey." In: *Journal of Imaging* 4.2, p. 33. DOI: 10.3390/jimaging4020033. URL: http://www.mdpi.com/2313-433X/4/2/33 (visited on 02/20/2018) (cit. on pp. 24–26, 47).

Breen, E. J. and R. Jones (Nov. 1996). "Attribute Openings, Thinnings, and Granulometries." In: *Computer Vision and Image Understanding* 64.3, pp. 377–389. DOI: 10.1006/cviu.1996.0066. URL: http://linkinghub.elsevier.com/retrieve/pii/S1077314296900661 (visited on 04/04/2019) (cit. on p. 23).

Breiman, L. (2001). "Random Forests." In: *Machine Learning archive* 45.1, pp. 5–32 (cit. on pp. 17, 18).

Briechle, S., P. Krzystek, and G. Vosselman (2020). "Classification of Tree Species and Standing Dead Trees by Fusing UAV-Based LiDAR Data and Multispectral Imagery in the 3d Deep Neural Network Pointnet++." In: 2, p. 8 (cit. on p. 15).

Brodu, N. and D. Lague (2012). "3D Terrestrial Lidar Data Classification of Complex Natural Scenes Using a Multi-Scale Dimensionality Criterion: Applications in Geomorphology." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 68, pp. 121–134. URL: http://www.sciencedirect.com/science/article/pii/S0924271612000330 (visited on 09/06/2017) (cit. on pp. 15, 17, 35, 36, 105).

Calderon, S. and T. Boubekeur (2014). "Point Morphology." In: *ACM Transactions on Graphics (Proc. SIGGRAPH 2014)* (cit. on pp. 28, 77, 79, 95).

Cao, L., N. C. Coops, J. L. Innes, J. Dai, H. Ruan, and G. She (2016). "Tree Species Classification in Subtropical Forests Using Small-Footprint Full-Waveform LiDAR

Data." In: *International Journal of Applied Earth Observation and Geoinformation* 49, pp. 39–51 (cit. on pp. 15, 22).

Carlinet, E., S. Crozet, and T. Geraud (2018). "The Tree of Shapes Turned into a Max-Tree: A Simple and Efficient Linear Algorithm." In: *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 1488–1492 (cit. on p. 25).

Cavallaro, G., M. Dalla Mura, J. A. Benediktsson, and L. Bruzzone (2015). "Extended Self-Dual Attribute Profiles for the Classification of Hyperspectral Images." In: *IEEE Geoscience and Remote Sensing Letters* 12.8, pp. 1690–1694 (cit. on p. 36).

Cavallaro, G., N. Falco, M. Dalla Mura, and J. A. Benediktsson (Apr. 2017). "Automatic Attribute Profiles." In: *IEEE Transactions on Image Processing* 26.4, pp. 1859–1872. doi: 10.1109/TIP.2017.2664667. url: http://ieeexplore.ieee.org/document/7842555/ (visited on 10/11/2018) (cit. on pp. 27, 39, 119).

Chase, A. F., D. Z. Chase, J. F. Weishampel, J. B. Drake, R. L. Shrestha, K. C. Slatton, J. J. Awe, and W. E. Carter (Feb. 2011). "Airborne LiDAR, Archaeology, and the Ancient Maya Landscape at Caracol, Belize." In: *Journal of Archaeological Science* 38.2, pp. 387–398. doi: 10.1016/j.jas.2010.09.018. url: https://linkinghub.elsevier.com/retrieve/pii/S0305440310003286 (visited on 11/27/2020) (cit. on p. 16).

Chehata, N., L. Guo, and C. Mallet (2009). "Airborne Lidar Feature Selection for Urban Classification Using Random Forests." In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 38 (Part 3), W8 (cit. on pp. 13, 16–18, 105).

Chun, B. and J.-M. Guldmann (May 2014). "Spatial Statistical Analysis and Simulation of the Urban Heat Island in High-Density Central Cities." In: *Landscape and Urban Planning* 125, pp. 76–88. doi: 10.1016/j.landurbplan.2014.01.016. url: https://linkinghub.elsevier.com/retrieve/pii/S0169204614000243 (visited on 11/27/2020) (cit. on p. 14).

Cousty, J., L. Najman, F. Dias, and J. Serra (Apr. 2013). "Morphological Filtering on Graphs." In: *Computer Vision and Image Understanding* 117.4, pp. 370–385. doi: 10.1016/j.cviu.2012.08.016. url: https://linkinghub.elsevier.com/retrieve/pii/S107731421200183X (visited on 02/17/2020) (cit. on p. 119).

Cousty, J., L. Najman, and B. Perret (2013). "Constructive Links between Some Morphological Hierarchies on Edge-Weighted Graphs." In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Ed. by C. L. L. Hendriks, G. Borgefors, and R. Strand. Red. by D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, and G. Weikum. Vol. 7883. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 86–97. doi: 10.1007/978-3-642-38294-9_8. url: http://link.springer.com/10.1007/978-3-642-38294-9_8 (visited on 11/30/2020) (cit. on p. 24).

Dalla Mura, M., J. A. Benediktsson, and L. Bruzzone (2011). "Self-Dual Attribute Profiles for the Analysis of Remote Sensing Images." In: *International Symposium*

on Mathematical Morphology and Its Applications to Signal and Image Processing. Springer, pp. 320–330 (cit. on pp. 27, 36).

Dalla Mura, M., J. A. Benediktsson, B. Waske, and L. Bruzzone (2010). "Morphological Attribute Profiles for the Analysis of Very High Resolution Images." In: *IEEE Transactions on Geoscience and Remote Sensing* 48.10, pp. 3747–3762 (cit. on pp. 27, 36, 94).

Damodaran, B. B., J. Höhle, and S. Lefèvre (Mar. 1, 2017). "Attribute Profiles on Derived Features for Urban Land Cover Classification." In: *Photogrammetric Engineering & Remote Sensing* 83.3, pp. 183–193. DOI: 10.14358/PERS.83.3.183. URL: http://www.ingentaconnect.com/content/10.14358/PERS.83.3.183 (visited on 12/15/2017) (cit. on pp. 29, 35, 41).

Das, A., K. Bhardwaj, S. Patra, and L. Bruzzone (2020). "A Novel Threshold Detection Technique for the Automatic Construction of Attribute Profiles in Hyperspectral Images." In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 13, pp. 1374–1384. DOI: 10.1109/JSTARS.2020.2981164. URL: https://ieeexplore.ieee.org/document/9050483/ (visited on 12/04/2020) (cit. on pp. 27, 119).

Demir, B. and L. Bruzzone (Apr. 2016). "Histogram-Based Attribute Profiles for Classification of Very High Resolution Remote Sensing Images." In: *IEEE Transactions on Geoscience and Remote Sensing* 54.4, pp. 2096–2107. DOI: 10.1109/TGRS.2015.2496167. URL: http://ieeexplore.ieee.org/document/7358126/ (visited on 12/02/2020) (cit. on p. 27).

Dufour, A., O. Tankyevych, B. Naegel, H. Talbot, C. Ronse, J. Baruthio, P. Dokládal, and N. Passat (Feb. 2013). "Filtering and Segmentation of 3D Angiographic Data: Advances Based on Mathematical Morphology." In: *Medical Image Analysis* 17.2, pp. 147–164. DOI: 10.1016/j.media.2012.08.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S1361841512001119 (visited on 12/13/2018) (cit. on pp. 80, 95).

Eitel, J. U., B. Höfle, L. A. Vierling, A. Abellán, G. P. Asner, J. S. Deems, C. L. Glennie, P. C. Joerg, A. L. LeWinter, T. S. Magney, G. Mandlburger, D. C. Morton, J. Müller, and K. T. Vierling (Dec. 2016). "Beyond 3-D: The New Spectrum of Lidar Applications for Earth and Ecological Sciences." In: *Remote Sensing of Environment* 186, pp. 372–392. DOI: 10.1016/j.rse.2016.08.018. URL: https://linkinghub.elsevier.com/retrieve/pii/S0034425716303212 (visited on 11/05/2020) (cit. on pp. 13, 15, 105).

Ferdosi, B. J., H. Buddelmeijer, S. Trager, M. H. F. Wilkinson, and J. B. T. M. Roerdink (Oct. 2010). "Finding and Visualizing Relevant Subspaces for Clustering High-Dimensional Astronomical Data Using Connected Morphological Operators." In: *2010 IEEE Symposium on Visual Analytics Science and Technology*. 2010 IEEE Symposium on Visual Analytics Science and Technology (VAST). Salt Lake City, UT, USA: IEEE, pp. 35–42. DOI: 10.1109/VAST.2010.5652450. URL: http://ieeexplore.ieee.org/document/5652450/ (visited on 12/13/2018) (cit. on pp. 79, 81, 95).

Fernandez-Diaz, J., W. Carter, C. Glennie, R. Shrestha, Z. Pan, N. Ekhtari, A. Singhania, D. Hauser, and M. Sartori (Nov. 10, 2016). "Capability Assessment and Performance Metrics for the Titan Multispectral Mapping Lidar." In: *Remote Sensing* 8.11, p. 936. DOI: 10.3390/rs8110936. URL: http://www.mdpi.com/2072-4292/8/11/936 (visited on 01/19/2018) (cit. on p. 5).

Ferraz, A., F. Bretar, S. Jacquemoud, G. Gonçalves, L. Pereira, M. Tomé, and P. Soares (2012). "3-D Mapping of a Multi-Layered Mediterranean Forest Using ALS Data." In: *Remote Sensing of Environment* 121, pp. 210–223 (cit. on p. 14).

Friedman, J. H., J. L. Bentley, and R. A. Finkel (1977). "An Algorithm for Finding Best Matches in Logarithmic Expected Time." In: *ACM Transactions on Mathematical Software* 3.3, pp. 209–226 (cit. on p. 17).

Géraud, T., E. Carlinet, S. Crozet, and L. Najman (2013). "A Quasi-Linear Algorithm to Compute the Tree of Shapes of nD Images." In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Ed. by C. L. L. Hendriks, G. Borgefors, and R. Strand. Red. by D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, and G. Weikum. Vol. 7883. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 98–110. URL: http://link.springer.com/10.1007/978-3-642-38294-9_9 (visited on 12/03/2018) (cit. on pp. 25, 80).

Ghamisi, P., J. A. Benediktsson, and S. Phinn (July 3, 2015). "Land-Cover Classification Using Both Hyperspectral and LiDAR Data." In: *International Journal of Image and Data Fusion* 6.3, pp. 189–215. DOI: 10.1080/19479832.2015.1055833. URL: http://www.tandfonline.com/doi/full/10.1080/19479832.2015.1055833 (visited on 10/17/2018) (cit. on pp. 35, 36, 41).

Ghamisi, P. and B. Hofle (May 2017). "LiDAR Data Classification Using Extinction Profiles and a Composite Kernel Support Vector Machine." In: *IEEE Geoscience and Remote Sensing Letters* 14.5, pp. 659–663. DOI: 10.1109/LGRS.2017.2669304. URL: http://ieeexplore.ieee.org/document/7873288/ (visited on 05/31/2020) (cit. on p. 29).

Ghamisi, P., R. Souza, J. A. Benediktsson, X. X. Zhu, L. Rittner, and R. A. Lotufo (Oct. 2016). "Extinction Profiles for the Classification of Remote Sensing Data." In: *IEEE Transactions on Geoscience and Remote Sensing* 54.10, pp. 5631–5645. DOI: 10.1109/TGRS.2016.2561842. URL: http://ieeexplore.ieee.org/document/7514921/ (visited on 10/17/2018) (cit. on p. 27).

Gil, A. L., L. Núñez-Casillas, M. Isenburg, A. A. Benito, J. R. Bello, and M. Arbelo (2013). "A Comparison between LiDAR and Photogrammetry Digital Terrain Models in a Forest Area on Tenerife Island." In: *Canadian Journal of Remote Sensing*, p. 15 (cit. on p. 13).

Gorte, B. and N. Pfeifer (2004). "Structuring Laser-Scanned Trees Using 3D Mathematical Morphology." In: *International Archives of Photogrammetry and Remote Sensing* 35.B5, pp. 929–933 (cit. on pp. 79, 95, 105).

Gotz, M., G. Cavallaro, T. Geraud, M. Book, and M. Riedel (Nov. 1, 2018). "Parallel Computation of Component Trees on Distributed Memory Machines." In: *IEEE Transactions on Parallel and Distributed Systems* 29.11, pp. 2582–2598. DOI: 10.1109/TPDS.2018.2829724. URL: https://ieeexplore.ieee.org/document/8360392/ (visited on 12/01/2020) (cit. on p. 25).

Grossiord, E., H. Talbot, N. Passat, M. Meignan, P. Terve, and L. Najman (Apr. 2015). "Hierarchies and Shape-Space for Pet Image Segmentation." In: *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*. 2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI 2015). Brooklyn, NY, USA: IEEE, pp. 1118–1121. DOI: 10.1109/ISBI.2015.7164068. URL: http://ieeexplore.ieee.org/document/7164068/ (visited on 12/03/2018) (cit. on pp. 80, 95).

Gueguen, L. and G. K. Ouzounis (Sept. 2012). "Hierarchical Data Representation Structures for Interactive Image Information Mining." In: *International Journal of Image and Data Fusion* 3.3, pp. 221–241. DOI: 10.1080/19479832.2012.697924. URL: http://www.tandfonline.com/doi/abs/10.1080/19479832.2012.697924 (visited on 02/06/2020) (cit. on p. 47).

Guinard, S. and L. Landrieu (May 31, 2017). "Weakly Supervised Segmentation-Aided Classification of Urban Scenes from 3d LiDAR Point Clouds." In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-1/W1, pp. 151–157. DOI: 10.5194/isprs-archives-XLII-1-W1-151-2017. URL: http://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-1-W1/151/2017/ (visited on 06/25/2019) (cit. on p. 14).

Guo, L., N. Chehata, C. Mallet, and S. Boukir (Jan. 2011). "Relevance of Airborne Lidar and Multispectral Image Data for Urban Scene Classification Using Random Forests." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 66.1, pp. 56–66. DOI: 10.1016/j.isprsjprs.2010.08.007. URL: http://linkinghub.elsevier.com/retrieve/pii/S0924271610000705 (visited on 09/11/2017) (cit. on p. 18).

Guo, Y., H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun (June 23, 2020). *Deep Learning for 3D Point Clouds: A Survey*. arXiv: 1912.12033 [cs, eess]. URL: http://arxiv.org/abs/1912.12033 (visited on 11/16/2020) (cit. on pp. vii, 7).

Guyot, A., L. Hubert-Moy, and T. Lorho (Feb. 1, 2018). "Detecting Neolithic Burial Mounds from LiDAR-Derived Elevation Data Using a Multi-Scale Approach and Machine Learning Techniques." In: *Remote Sensing* 10.2, p. 225. DOI: 10.3390/rs10020225. URL: http://www.mdpi.com/2072-4292/10/2/225 (visited on 02/03/2020) (cit. on p. 16).

Haas, A., G. Matheron, and J. Serra (1967). "Morphologie Mathématique et Granulométries En Place." In: *Annales Des Mines*. Vol. 11. 736-753, pp. 7–3 (cit. on p. 23).

Hakala, T., O. Nevalainen, S. Kaasalainen, and R. Mäkipää (Mar. 12, 2015). "Technical Note: Multispectral Lidar Time Series of Pine Canopy Chlorophyll Content." In: *Biogeosciences* 12.5, pp. 1629–1634. DOI: 10.5194/bg-12-1629-2015. URL: https://www.biogeosciences.net/12/1629/2015/ (visited on 06/05/2019) (cit. on p. 13).

Hartsell, D., P. E. LaRocque, and J. Tripp (Oct. 26, 2016). "Galaxy: A New State of the Art Airborne Lidar System." In: SPIE Remote Sensing. Ed. by T. Erbertseder, T. Esch, and N. Chrysoulakis. Edinburgh, United Kingdom, 100080U. DOI: 10.1117/12.2241841. URL: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2241841 (visited on 12/06/2020) (cit. on p. 5).

Hernández, J. (Dec. 2009). "Analyse Morphologique d'images Pour La Modélisation d'environnements Urbains" (cit. on p. 83).

Hu, H., D. Munoz, J. A. Bagnell, and M. Hebert (May 2013). "Efficient 3-D Scene Analysis from Streaming Data." In: *2013 IEEE International Conference on Robotics and Automation*. 2013 IEEE International Conference on Robotics and Automation (ICRA). Karlsruhe, Germany: IEEE, pp. 2297–2304. DOI: 10.1109/ICRA.2013.6630888. URL: http://ieeexplore.ieee.org/document/6630888/ (visited on 11/23/2020) (cit. on p. 22).

Hu, M.-K. (Feb. 1962). "Visual Pattern Recognition by Moment Invariants." In: *IEEE Transactions on Information Theory* 8.2, pp. 179–187. DOI: 10.1109/TIT.1962.1057692. URL: http://ieeexplore.ieee.org/document/1057692/ (visited on 04/17/2020) (cit. on p. 58).

Jang, J.-D., V. Payan, A. A. Viau, and A. Devost (Mar. 2008). "The Use of Airborne Lidar for Orchard Tree Inventory." In: *International Journal of Remote Sensing* 29.6, pp. 1767–1780. DOI: 10.1080/01431160600928591. URL: https://www.tandfonline.com/doi/full/10.1080/01431160600928591 (visited on 11/12/2020) (cit. on p. 15).

Jayathunga, S., T. Owari, and S. Tsuyuki (Dec. 2018). "The Use of Fixed–Wing UAV Photogrammetry with LiDAR DTM to Estimate Merchantable Volume and Carbon Stock in Living Biomass over a Mixed Conifer–Broadleaf Forest." In: *International Journal of Applied Earth Observation and Geoinformation* 73, pp. 767–777. DOI: 10.1016/j.jag.2018.08.017. URL: https://linkinghub.elsevier.com/retrieve/pii/S0303243418302587 (visited on 11/25/2020) (cit. on p. 13).

Jones, R. (Sept. 1999). "Connected Filtering and Segmentation Using Component Trees." In: *Computer Vision and Image Understanding* 75.3, pp. 215–228. DOI: 10.1006/cviu.1999.0777. URL: https://linkinghub.elsevier.com/retrieve/pii/S1077314299907774 (visited on 11/30/2020) (cit. on p. 24).

Kazemier, J. J., G. K. Ouzounis, and M. H. F. Wilkinson (2017). "Connected Morphological Attribute Filters on Distributed Memory Parallel Machines." In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Ed. by J. Angulo, S. Velasco-Forero, and F. Meyer. Vol. 10225. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 357–368. DOI: 10.1007/978-3-319-57240-6_29. URL: http://link.springer.com/10.1007/978-3-319-57240-6_29 (visited on 12/01/2020) (cit. on p. 25).

Keqi Zhang, Shu-Ching Chen, D. Whitman, Mei-Ling Shyu, Jianhua Yan, and Chengcui Zhang (Apr. 2003). "A Progressive Morphological Filter for Removing Nonground Measurements from Airborne LIDAR Data." In: *IEEE Transactions on Geoscience and Remote Sensing* 41.4, pp. 872–882. DOI: 10.1109/TGRS.2003.810682.

URL: http://ieeexplore.ieee.org/document/1202973/ (visited on 11/14/2020) (cit. on pp. 20, 28).

Khodadadzadeh, M., J. Li, S. Prasad, and A. Plaza (June 2015). "Fusion of Hyperspectral and LiDAR Remote Sensing Data Using Multiple Feature Learning." In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 8.6, pp. 2971–2983. DOI: 10.1109/JSTARS.2015.2432037. URL: http://ieeexplore.ieee.org/document/7115053/ (visited on 02/27/2018) (cit. on pp. 35, 36).

Kingma, D. P. and J. Ba (Jan. 29, 2017). *Adam: A Method for Stochastic Optimization.* arXiv: 1412.6980 [cs]. URL: http://arxiv.org/abs/1412.6980 (visited on 11/05/2020) (cit. on p. 111).

Kiwanuka, F. N., G. K. Ouzounis, and M. H. Wilkinson (2009). "Surface-Area-Based Attribute Filtering in 3d." In: *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, pp. 70–81 (cit. on pp. 80, 95).

Kiwanuka, F. N. and M. H. F. Wilkinson (2012). "Radial Moment Invariants for Attribute Filtering in 3D." In: *Applications of Discrete Geometry and Mathematical Morphology*. Ed. by U. Köthe, A. Montanvert, and P. Soille. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 68–81 (cit. on pp. 80, 95).

Kwan, C., D. Gribben, B. Ayhan, S. Bernabe, A. Plaza, and M. Selva (2020). "Improving Land Cover Classification Using Extended Multi-Attribute Profiles (EMAP) Enhanced Color, Near Infrared, and LiDAR Data." In: p. 28 (cit. on p. 29).

Landrieu, L. and M. Simonovsky (2018). "Large-Scale Point Cloud Semantic Segmentation with Superpoint Graphs." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4558–4567 (cit. on pp. 18, 63, 105).

Landrieu, L., H. Raguet, B. Vallet, C. Mallet, and M. Weinmann (Oct. 2017). "A Structured Regularization Framework for Spatially Smoothing Semantic Labelings of 3D Point Clouds." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 132, pp. 102–118. DOI: 10.1016/j.isprsjprs.2017.08.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S0924271617302988 (visited on 06/25/2019) (cit. on p. 18).

Le Saux, B., N. Yokoya, R. Hansch, and S. Prasad (Mar. 2018). "2018 IEEE GRSS Data Fusion Contest: Multimodal Land Use Classification [Technical Committees]." In: *IEEE Geoscience and Remote Sensing Magazine* 6.1, pp. 52–54. DOI: 10.1109/MGRS.2018.2798161. URL: http://ieeexplore.ieee.org/document/8328995/ (visited on 10/29/2018) (cit. on p. 111).

Li, W., Q. Guo, M. K. Jakubowski, and M. Kelly (Jan. 1, 2012). "A New Method for Segmenting Individual Trees from the Lidar Point Cloud." In: *Photogrammetric Engineering & Remote Sensing* 78.1, pp. 75–84. DOI: 10.14358/PERS.78.1.75. URL: http://openurl.ingenta.com/content/xref?genre=article&issn=0099-1112&volume=78&issue=1&spage=75 (visited on 11/14/2020) (cit. on p. 14).

Liao, W., J. Chanussot, M. D. Mura, X. Huang, R. Bellens, S. Gautama, and W. Philips (2017). "Taking Optimal Advantage of Fine Spatial Information." In: *ieee Geoscience and remote sensing magazine*, p. 21 (cit. on p. 29).

Liaw, A., M. Wiener, et al. (2002). "Classification and Regression by randomForest." In: *R news* 2.3, pp. 18–22 (cit. on p. 40).

Lodha, S. K., E. J. Kreps, D. P. Helmbold, and D. Fitzpatrick (June 2006). "Aerial LiDAR Data Classification Using Support Vector Machines (SVM)." In: *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*. Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06). Chapel Hill, NC, USA: IEEE, pp. 567–574. doi: 10.1109/3DPVT.2006.23. url: http://ieeexplore.ieee.org/document/4155775/ (visited on 10/17/2018) (cit. on pp. 21, 22, 35, 105).

Magnussen, S., T. Nord-Larsen, and T. Riis-Nielsen (June 2018). "Lidar Supported Estimators of Wood Volume and Aboveground Biomass from the Danish National Forest Inventory (2012–2016)." In: *Remote Sensing of Environment* 211, pp. 146–153. doi: 10.1016/j.rse.2018.04.015. url: https://linkinghub.elsevier.com/retrieve/pii/S0034425718301615 (visited on 12/07/2020) (cit. on pp. v, 3).

Mahmood, Z., G. Thoonen, and P. Scheunders (July 2012). "Automatic Threshold Selection for Morphological Attribute Profiles." In: *2012 IEEE International Geoscience and Remote Sensing Symposium*. IGARSS 2012 - 2012 IEEE International Geoscience and Remote Sensing Symposium. Munich, Germany: IEEE, pp. 4946–4949. doi: 10.1109/IGARSS.2012.6352502. url: http://ieeexplore.ieee.org/document/6352502/ (visited on 06/08/2020) (cit. on p. 119).

Mallet, C. (2010). "Analyse Des Données Lidar Aéroportées à Retour d'Onde Complète Pour La Cartographie Des Milieux Urbains." Télécom ParisTech (cit. on p. 12).

Mallet, C. and F. Bretar (2009). "Full-Waveform Topographic Lidar: State-of-the-Art." In: *ISPRS Journal of photogrammetry and remote sensing* 64.1, pp. 1–16 (cit. on p. 13).

Mallet, C., F. Bretar, M. Roux, U. Soergel, and C. Heipke (Dec. 2011). "Relevance Assessment of Full-Waveform Lidar Data for Urban Area Classification." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 66.6, S71–S84. doi: 10.1016/j.isprsjprs.2011.09.008. url: http://linkinghub.elsevier.com/retrieve/pii/S0924271611001055 (visited on 10/17/2018) (cit. on pp. 13, 16, 35, 36, 105).

Maragos, P. (July 1989). "Pattern Spectrum and Multiscale Shape Representation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11.7, pp. 701–716. doi: 10.1109/34.192465. url: https://ieeexplore.ieee.org/document/192465/ (visited on 02/06/2020) (cit. on pp. 28, 47).

Matheron, G. (1964). "Étude Théorique Des Granulométries." In: *en français, École des Mines de Paris, Note Géostatistique* (cit. on p. 23).

Mathews, A. J. and J. L. R. Jensen (Aug. 20, 2012). "An Airborne LiDAR-Based Methodology for Vineyard Parcel Detection and Delineation." In: *International Journal of Remote Sensing* 33.16, pp. 5251–5267. doi: 10.1080/01431161.2012.663114. url: https://www.tandfonline.com/doi/full/10.1080/01431161.2012.663114 (visited on 11/12/2020) (cit. on p. 15).

Matikainen, L., M. Lehtomäki, E. Ahokas, J. Hyyppä, M. Karjalainen, A. Jaakkola, A. Kukko, and T. Heinonen (Sept. 2016). "Remote Sensing Methods for Power Line

Corridor Surveys." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 119, pp. 10–31. DOI: 10.1016/j.isprsjprs.2016.04.011. URL: https://linkinghub.elsevier.com/retrieve/pii/S0924271616300697 (visited on 11/25/2020) (cit. on pp. 13, 21).

MATURANA, D. and S. SCHERER (Sept. 2015). "VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition." In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Hamburg, Germany: IEEE, pp. 922–928. DOI: 10.1109/IROS.2015.7353481. URL: http://ieeexplore.ieee.org/document/7353481/ (visited on 09/03/2019) (cit. on p. 22).

MERCIOL, F., T. BALEM, and S. LEFÈVRE (2017). "Efficient and Large-Scale Land Cover Classification Using Multiscale Image Analysis." In: p. 5 (cit. on pp. 25, 43).

MERCIOL, F., L. FAUCQUEUR, B. DAMODARAN, P.-Y. RÉMY, B. DESCLÉE, F. DAZIN, S. LEFÈVRE, A. MASSE, and C. SANNIER (Jan. 18, 2019). "GEOBIA at the Terapixel Scale: Toward Efficient Mapping of Small Woody Features from Heterogeneous VHR Scenes." In: *ISPRS International Journal of Geo-Information* 8.1, p. 46. DOI: 10.3390/ijgi8010046. URL: http://www.mdpi.com/2220-9964/8/1/46 (visited on 12/03/2020) (cit. on p. 5).

MONASSE, P. and F. GUICHARD (May 2000a). "Fast Computation of a Contrast-Invariant Image Representation." In: *IEEE Transactions on Image Processing* 9.5, pp. 860–872. DOI: 10.1109/83.841532. URL: http://ieeexplore.ieee.org/document/841532/ (visited on 02/03/2020) (cit. on pp. 25, 49).

MONASSE, P. and F. GUICHARD (June 2000b). "Scale-Space from a Level Lines Tree." In: *Journal of Visual Communication and Image Representation* 11.2, pp. 224–236. DOI: 10.1006/jvci.1999.0441. URL: https://linkinghub.elsevier.com/retrieve/pii/S1047320399904418 (visited on 03/11/2020) (cit. on p. 25).

MONGUS, D., N. LUKAČ, and B. ŽALIK (July 2014). "Ground and Building Extraction from LiDAR Data Based on Differential Morphological Profiles and Locally Fitted Surfaces." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 93, pp. 145–156. DOI: 10.1016/j.isprsjprs.2013.12.002. URL: http://linkinghub.elsevier.com/retrieve/pii/S0924271613002840 (visited on 12/11/2017) (cit. on p. 29).

MONGUS, D. and B. ZALIK (Jan. 2014). "Computationally Efficient Method for the Generation of a Digital Terrain Model From Airborne LiDAR Data Using Connected Operators." In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 7.1, pp. 340–351. DOI: 10.1109/JSTARS.2013.2262996. URL: http://ieeexplore.ieee.org/document/6521401/ (visited on 04/04/2019) (cit. on pp. 21, 28).

MONGUS, D. and B. ŽALIK (Oct. 2015). "An Efficient Approach to 3D Single Tree-Crown Delineation in LiDAR Data." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 108, pp. 219–233. DOI: 10.1016/j.isprsjprs.2015.08.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S0924271615001951 (visited on 04/08/2019) (cit. on p. 14).

Muja, M. and D. G. Lowe (2009). "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration." In: *International Conference on Computer Vision Theory and Application VISSAPP'09*, pp. 331–340 (cit. on p. 17).

Najafi, M., S. Taghavi Namin, M. Salzmann, and L. Petersson (2014). "Non-Associative Higher-Order Markov Networks for Point Cloud Classification." In: *Computer Vision – ECCV 2014*. Ed. by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars. Vol. 8693. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 500–515. doi: 10.1007/978-3-319-10602-1_33. url: http://link.springer.com/10.1007/978-3-319-10602-1_33 (visited on 11/22/2020) (cit. on p. 18).

Najman, L. and M. Couprie (Nov. 2006). "Building the Component Tree in Quasi-Linear Time." In: *IEEE Transactions on Image Processing* 15.11, pp. 3531–3539. doi: 10.1109/TIP.2006.877518. url: http://ieeexplore.ieee.org/document/1709995/ (visited on 04/08/2019) (cit. on p. 25).

Ni, H., X. Lin, and J. Zhang (Mar. 18, 2017). "Classification of ALS Point Cloud with Improved Point Cloud Segmentation and Random Forests." In: *Remote Sensing* 9.3, p. 288. doi: 10.3390/rs9030288. url: http://www.mdpi.com/2072-4292/9/3/288 (visited on 09/25/2018) (cit. on p. 18).

Niemeyer, J., F. Rottensteiner, and U. Soergel (Jan. 2014). "Contextual Classification of Lidar Data and Building Object Detection in Urban Areas." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 87, pp. 152–165. doi: 10.1016/j.isprsjprs.2013.11.001. url: http://linkinghub.elsevier.com/retrieve/pii/S0924271613002359 (visited on 12/14/2017) (cit. on pp. 14, 16–18, 35, 36, 105).

Nord-Larsen, T. and J. Schumacher (Apr. 2012). "Estimation of Forest Resources from a Country Wide Laser Scanning Survey and National Forest Inventory Data." In: *Remote Sensing of Environment* 119, pp. 148–157. doi: 10.1016/j.rse.2011.12.022. url: https://linkinghub.elsevier.com/retrieve/pii/S0034425712000107 (visited on 11/14/2020) (cit. on p. 14).

Ouzounis, G. K. and P. Soille (2011). "Pattern Spectra from Partition Pyramids and Hierarchies." In: *Mathematical Morphology and Its Applications to Image and Signal Processing*. Ed. by P. Soille, M. Pesaresi, and G. K. Ouzounis. Vol. 6671. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 108–119. doi: 10.1007/978-3-642-21569-8_10. url: http://link.springer.com/10.1007/978-3-642-21569-8_10 (visited on 02/06/2020) (cit. on p. 47).

Padilla, F. J. A., B. Romaniuk, B. Naegel, S. Servagi-Vernat, D. Morland, D. Papathanassiou, and N. Passat (Apr. 2018). "Hierarchical Forest Attributes for Multimodal Tumor Segmentation on FDG-PET/Contrast-Enhanced CT." In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018). Washington, DC: IEEE, pp. 163–167. doi: 10.1109/ISBI.2018.8363546. url: https://ieeexplore.ieee.org/document/8363546/ (visited on 12/03/2018) (cit. on pp. 80, 95).

Passalacqua, P., P. Belmont, D. M. Staley, J. D. Simley, J. R. Arrowsmith, C. A. Bode, C. Crosby, S. B. DeLong, N. F. Glenn, S. A. Kelly, D. Lague,

H. Sangireddy, K. Schaffrath, D. G. Tarboton, T. Wasklewicz, and J. M. Wheaton (Sept. 2015). "Analyzing High Resolution Topography for Advancing the Understanding of Mass and Energy Transfer through Landscapes: A Review." In: *Earth-Science Reviews* 148, pp. 174–193. DOI: 10.1016/j.earscirev.2015.05.012. URL: https://linkinghub.elsevier.com/retrieve/pii/S0012825215300015 (visited on 11/27/2020) (cit. on pp. v, 3, 15).

Pedergnana, M., P. R. Marpu, M. Dalla Mura, J. A. Benediktsson, and L. Bruzzone (Nov. 2012). "Classification of Remote Sensing Optical and LiDAR Data Using Extended Attribute Profiles." In: *IEEE Journal of Selected Topics in Signal Processing* 6.7, pp. 856–865. DOI: 10.1109/JSTSP.2012.2208177. URL: http://ieeexplore.ieee.org/document/6237479/ (visited on 02/27/2018) (cit. on pp. 29, 35, 36, 41).

Pelletier, J. D. and C. A. Orem (Oct. 2014). "How Do Sediment Yields from Post-Wildfire Debris-Laden Flows Depend on Terrain Slope, Soil Burn Severity Class, and Drainage Basin Area? Insights from Airborne-LiDAR Change Detection: POST-FIRE SEDIMENT YIELDS MEASURED BY AIRBORNE LIDAR." In: *Earth Surface Processes and Landforms* 39.13, pp. 1822–1832. DOI: 10.1002/esp.3570. URL: http://doi.wiley.com/10.1002/esp.3570 (visited on 12/07/2020) (cit. on pp. v, 3).

Perret, B., G. Chierchia, J. Cousty, S. F. Guimarães, Y. Kenmochi, and L. Najman (2019). "Higra: Hierarchical Graph Analysis." In: *SoftwareX* 10, pp. 1–6. DOI: 10.1016/j.softx.2019.100335 (cit. on p. 52).

Pesaresi, M. and J. Benediktsson (Feb. 2001). "A New Approach for the Morphological Segmentation of High-Resolution Satellite Imagery." In: *IEEE Transactions on Geoscience and Remote Sensing* 39.2, pp. 309–320. DOI: 10.1109/36.905239. URL: http://ieeexplore.ieee.org/document/905239/ (visited on 06/12/2019) (cit. on p. 26).

Peternell, M. and T. Steiner (May 2007). "Minkowski Sum Boundary Surfaces of 3D-Objects." In: *Graphical Models* 69.3-4, pp. 180–190. DOI: 10.1016/j.gmod.2007.01.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S1524070307000021 (visited on 01/16/2019) (cit. on pp. 77, 79).

Pham, M.-T., E. Aptoula, and S. Lefevre (Jan. 2018). "Feature Profiles from Attribute Filtering for Classification of Remote Sensing Images." In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 11.1, pp. 249–256. DOI: 10.1109/JSTARS.2017.2773367. URL: http://ieeexplore.ieee.org/document/8118175/ (visited on 03/25/2019) (cit. on pp. 27, 119).

Pham, M.-T., S. Lefevre, and E. Aptoula (Feb. 2018). "Local Feature-Based Attribute Profiles for Optical Remote Sensing Image Classification." In: *IEEE Transactions on Geoscience and Remote Sensing* 56.2, pp. 1199–1212. DOI: 10.1109/TGRS.2017.2761402. URL: http://ieeexplore.ieee.org/document/8082117/ (visited on 03/13/2018) (cit. on pp. 27, 43).

Pham, M.-T., S. Lefèvre, E. Aptoula, and L. Bruzzone (Mar. 27, 2018). *Recent Developments from Attribute Profiles for Remote Sensing Image Classification*. arXiv:

1803.10036 [cs]. URL: http://arxiv.org/abs/1803.10036 (visited on 11/12/2019) (cit. on p. 94).

Pingel, T. J., K. C. Clarke, and W. A. McBride (Mar. 2013). "An Improved Simple Morphological Filter for the Terrain Classification of Airborne LIDAR Data." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 77, pp. 21–30. DOI: 10.1016/j.isprsjprs.2012.12.002. URL: http://linkinghub.elsevier.com/retrieve/pii/S0924271613000026 (visited on 12/11/2017) (cit. on pp. 20, 28).

Pontil, M. and A. Verri (1998). "Properties of Support Vector Machines." In: *Neural Computation* 10.4, pp. 955–974 (cit. on p. 17).

Qi, C. R., H. Su, K. Mo, and L. J. Guibas (2017). "Pointnet: Deep Learning on Point Sets for 3d Classification and Segmentation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652–660 (cit. on pp. 18, 105).

Qi, C. R., L. Yi, H. Su, and L. J. Guibas (2017). "Pointnet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space." In: *Advances in Neural Information Processing Systems*, pp. 5099–5108 (cit. on pp. 18, 105).

Rao, M., P. Tang, and Z. Zhang (Dec. 2019). "Spatial–Spectral Relation Network for Hyperspectral Image Classification with Limited Training Samples." In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12.12, pp. 5086–5100. DOI: 10.1109/JSTARS.2019.2957047 (cit. on pp. 106, 107).

Roynard, X., J.-E. Deschaud, and F. Goulette (Nov. 2017). "Paris-Lille-3D: A Large and High-Quality Ground Truth Urban Point Cloud Dataset for Automatic Segmentation and Classification." In: *ArXiv e-prints* (cit. on pp. 84, 97, 100).

Salembier, P., A. Oliveras, and L. Garrido (Apr. 1998). "Antiextensive Connected Operators for Image and Sequence Processing." In: *IEEE Transactions on Image Processing* 7.4, pp. 555–570. DOI: 10.1109/83.663500. URL: http://ieeexplore.ieee.org/document/663500/ (visited on 12/17/2018) (cit. on pp. 23–25, 48, 58, 83, 95).

Salembier, P. and J. Serra (1995). "Flat Zones Filtering, Connected Operators, and Filters by Reconstruction." In: *IEEE Transactions on Image Processing* 4.8, pp. 1153–1160. DOI: 10.1109/83.403422. URL: http://ieeexplore.ieee.org/document/403422/ (visited on 11/13/2020) (cit. on p. 23).

Salembier, P. and M. Wilkinson (Nov. 2009). "Connected Operators." In: *IEEE Signal Processing Magazine* 26.6, pp. 136–157. DOI: 10.1109/MSP.2009.934154. URL: http://ieeexplore.ieee.org/document/5230812/ (visited on 12/13/2018) (cit. on p. 83).

Serna, A. and B. Marcotegui (July 2014). "Detection, Segmentation and Classification of 3D Urban Objects Using Mathematical Morphology and Supervised Learning." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 93, pp. 243–255. DOI: 10.1016/j.isprsjprs.2014.03.015. URL: http://linkinghub.elsevier.com/retrieve/pii/S0924271614000872 (visited on 06/29/2018) (cit. on pp. 29, 77, 79, 105).

Serna, A., B. Marcotegui, and J. Hernández (2016). "Segmentation of Façades from Urban 3D Point Clouds Using Geometrical and Morphological Attribute-Based Operators." In: *ISPRS International Journal of Geo-Information*. ISPRS International

Journal of Geo-Information 5.1, p. 6. DOI: 10.3390/ijgi5010006. URL: https://hal-mines-paristech.archives-ouvertes.fr/hal-01254284 (cit. on pp. 29, 79, 95).

Shan, J. and S. Aparajithan (2005). "Urban DEM Generation from Raw LiDAR Data." In: *Photogrammetric Engineering & Remote Sensing* 71.2, pp. 217–226 (cit. on p. 105).

Shendryk, I., M. Broich, M. G. Tulbure, and S. V. Alexandrov (Feb. 2016). "Bottom-up Delineation of Individual Trees from Full-Waveform Airborne Laser Scans in a Structurally Complex Eucalypt Forest." In: *Remote Sensing of Environment* 173, pp. 69–83. DOI: 10.1016/j.rse.2015.11.008. URL: https://linkinghub.elsevier.com/retrieve/pii/S0034425715301966 (visited on 10/20/2020) (cit. on pp. 14, 15).

Shendryk, I., M. Broich, M. G. Tulbure, A. McGrath, D. Keith, and S. V. Alexandrov (Dec. 2016). "Mapping Individual Tree Health Using Full-Waveform Airborne Laser Scans and Imaging Spectroscopy: A Case Study for a Floodplain Eucalypt Forest." In: *Remote Sensing of Environment* 187, pp. 202–217. DOI: 10.1016/j.rse.2016.10.014. URL: https://linkinghub.elsevier.com/retrieve/pii/S0034425716303868 (visited on 09/16/2019) (cit. on pp. v, 3, 15).

Simonyan, K. and A. Zisserman (Apr. 10, 2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv: 1409.1556 [cs]. URL: http://arxiv.org/abs/1409.1556 (visited on 10/15/2020) (cit. on p. 66).

Soille, P. and M. Pesaresi (2002). "Advances in Mathematical Morphology Applied to Geoscience and Remote Sensing." In: *IEEE Transactions on Geoscience and Remote Sensing* 40.9, pp. 2042–2055 (cit. on p. 26).

Souza, R., L. Rittner, R. Machado, and R. Lotufo (2015). "A Comparison Between Extinction Filters and Attribute Filters." In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Ed. by J. A. Benediktsson, J. Chanussot, L. Najman, and H. Talbot. Vol. 9082. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 63–74. DOI: 10.1007/978-3-319-18720-4_6. URL: http://link.springer.com/10.1007/978-3-319-18720-4_6 (visited on 11/13/2020) (cit. on p. 24).

Sukhanov, S., D. Budylskii, I. Tankoyeu, R. Heremans, and C. Debes (July 2018). "Fusion of Lidar, Hyperspectral and RGB Data for Urban Land Use and Land Cover Classification." In: *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*. IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium. Valencia: IEEE, pp. 3864–3867. DOI: 10.1109/IGARSS.2018.8517333. URL: https://ieeexplore.ieee.org/document/8517333/ (visited on 06/03/2019) (cit. on pp. 21, 22, 64, 65).

Teo, T.-A. and H.-M. Wu (June 28, 2017). "Analysis of Land Cover Classification Using Multi-Wavelength LiDAR System." In: *Applied Sciences* 7.7, p. 663. DOI: 10.3390/app7070663. URL: http://www.mdpi.com/2076-3417/7/7/663 (visited on 10/31/2018) (cit. on pp. 21, 22).

Thomas, H., C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas (Aug. 19, 2019). *KPConv: Flexible and Deformable Convolution for Point*

Clouds. arXiv: 1904.08889 [cs]. URL: http://arxiv.org/abs/1904.08889 (visited on 03/30/2020) (cit. on pp. 19, 119).

Tooke, T. R., N. C. Coops, and J. Webster (Jan. 2014). "Predicting Building Ages from LiDAR Data with Random Forests for Building Energy Modeling." In: *Energy and Buildings* 68, pp. 603–610. DOI: 10.1016/j.enbuild.2013.10.004. URL: http://linkinghub.elsevier.com/retrieve/pii/S0378778813006506 (visited on 12/14/2017) (cit. on p. 14).

Urbach, E., J. Roerdink, and M. Wilkinson (Feb. 2007). "Connected Shape-Size Pattern Spectra for Rotation and Scale-Invariant Classification of Gray-Scale Images." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.2, pp. 272–285. DOI: 10.1109/TPAMI.2007.28. URL: http://ieeexplore.ieee.org/document/4042702/ (visited on 04/17/2020) (cit. on p. 28).

Urbach, E. and M. Wilkinson (Jan. 2002). "Shape-Only Granulometries and Grey-Scale Shape Filters." In: *Proc. Int. Symp. Math. Morphology (ISMM) 2002*, pp. 305–314 (cit. on pp. 58, 83).

Urien, H., I. Buvat, N. Rougon, M. Soussan, and I. Bloch (2017). "Brain Lesion Detection in 3D PET Images Using Max-Trees and a New Spatial Context Criterion." In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Ed. by J. Angulo, S. Velasco-Forero, and F. Meyer. Vol. 10225. Cham: Springer International Publishing, pp. 455–466. DOI: 10.1007/978-3-319-57240-6_37. URL: http://link.springer.com/10.1007/978-3-319-57240-6_37 (visited on 12/03/2018) (cit. on pp. 80, 95).

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory* (cit. on p. 17).

Wang, A., X. He, P. Ghamisi, and Y. Chen (2018). "LiDAR Data Classification Using Morphological Profiles and Convolutional Neural Networks." In: *IEEE Geoscience and Remote Sensing Letters* 15.5, pp. 774–778 (cit. on p. 29).

Wang, C.-K., Y.-H. Tseng, and H.-J. Chu (Jan. 8, 2014). "Airborne Dual-Wavelength LiDAR Data for Classifying Land Cover." In: *Remote Sensing* 6.1, pp. 700–715. DOI: 10.3390/rs6010700. URL: http://www.mdpi.com/2072-4292/6/1/700 (visited on 10/31/2018) (cit. on p. 64).

Wang, H. and C. Glennie (Oct. 2015). "Fusion of Waveform LiDAR Data and Hyperspectral Imagery for Land Cover Classification." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 108, pp. 1–11. DOI: 10.1016/j.isprsjprs.2015.05.012. URL: https://linkinghub.elsevier.com/retrieve/pii/S0924271615001495 (visited on 11/26/2018) (cit. on p. 22).

Weinmann, M., B. Jutzi, S. Hinz, and C. Mallet (July 2015). "Semantic Point Cloud Interpretation Based on Optimal Neighborhoods, Relevant Features and Efficient Classifiers." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 105, pp. 286–304. DOI: 10.1016/j.isprsjprs.2015.01.016. URL: https://linkinghub.elsevier.com/retrieve/pii/S0924271615000349 (visited on 10/18/2018) (cit. on pp. 36, 105).

Westenberg, M. A., J. B. T. M. Roerdink, and M. H. F. Wilkinson (Dec. 2007). "Volumetric Attribute Filtering and Interactive Visualization Using the Max-Tree Representation." In: *IEEE Transactions on Image Processing* 16.12, pp. 2943–2952.

DOI: 10.1109/TIP.2007.909317. URL: http://ieeexplore.ieee.org/document/4376245/ (visited on 12/13/2018) (cit. on pp. 80, 95).

Wichmann, V., M. Bremer, J. Lindenberger, M. Rutzinger, C. Georges, and F. Petrini-Monteferri (Aug. 19, 2015). "EVALUATING THE POTENTIAL OF MULTISPECTRAL AIRBORNE LIDAR FOR TOPOGRAPHIC MAPPING AND LAND COVER CLASSIFICATION." In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* II-3/W5, pp. 113–119. DOI: 10.5194/isprsannals-II-3-W5-113-2015. URL: http://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/II-3-W5/113/2015/ (visited on 06/05/2019) (cit. on p. 64).

Wilkinson, M., H. Gao, W. Hesselink, J.-E. Jonker, and A. Meijster (Oct. 2008). "Concurrent Computation of Attribute Filters on Shared Memory Parallel Machines." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.10, pp. 1800–1813. DOI: 10.1109/TPAMI.2007.70836. URL: http://ieeexplore.ieee.org/document/4407727/ (visited on 11/19/2020) (cit. on p. 25).

Wilkinson, M. and M. Westenberg (2001). "Shape Preserving Filament Enhancement Filtering." In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2001*. Ed. by W. Niessen and M. Viergever. Red. by G. Goos, J. Hartmanis, and J. van Leeuwen. Vol. 2208. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 770–777. DOI: 10.1007/3-540-45468-3_92. URL: http://link.springer.com/10.1007/3-540-45468-3_92 (visited on 12/13/2018) (cit. on pp. 79, 95).

Witharana, C., W. B. Ouimet, and K. M. Johnson (Mar. 4, 2018). "Using LiDAR and GEOBIA for Automated Extraction of Eighteenth–Late Nineteenth Century Relict Charcoal Hearths in Southern New England." In: *GIScience & Remote Sensing* 55.2, pp. 183–204. DOI: 10.1080/15481603.2018.1431356. URL: https://www.tandfonline.com/doi/full/10.1080/15481603.2018.1431356 (visited on 11/05/2020) (cit. on p. 105).

Xu, Y., B. Du, L. Zhang, D. Cerra, M. Pato, E. Carmona, S. Prasad, N. Yokoya, R. Hansch, and B. Le Saux (June 2019). "Advanced Multi-Sensor Optical Remote Sensing for Urban Land Use and Land Cover Classification: Outcome of the 2018 IEEE GRSS Data Fusion Contest." In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12.6, pp. 1709–1724. DOI: 10.1109/JSTARS.2019.2911113. URL: https://ieeexplore.ieee.org/document/8727489/ (visited on 05/29/2020) (cit. on p. 67).

Yan, W., A. Shaker, and P. LaRocque (Apr. 4, 2019). "Scan Line Intensity-Elevation Ratio (SLIER): An Airborne LiDAR Ratio Index for Automatic Water Surface Mapping." In: *Remote Sensing* 11.7, p. 814. DOI: 10.3390/rs11070814. URL: https://www.mdpi.com/2072-4292/11/7/814 (visited on 12/06/2020) (cit. on p. 4).

Yan, W. Y., A. Shaker, and N. El-Ashmawy (Mar. 2015). "Urban Land Cover Classification Using Airborne LiDAR Data: A Review." In: *Remote Sensing of Environment* 158, pp. 295–310. DOI: 10.1016/j.rse.2014.11.001. URL: http://linkinghub.

elsevier.com/retrieve/pii/S0034425714004374 (visited on 09/01/2018) (cit. on pp. v, 3).

Zhang, M., P. Ghamisi, and W. Li (Oct. 3, 2017). "Classification of Hyperspectral and LIDAR Data Using Extinction Profiles with Feature Fusion." In: *Remote Sensing Letters* 8.10, pp. 957–966. DOI: 10.1080/2150704X.2017.1335902. URL: https://www.tandfonline.com/doi/full/10.1080/2150704X.2017.1335902 (visited on 06/10/2020) (cit. on p. 29).

Zhang, W., J. Qi, P. Wan, H. Wang, D. Xie, X. Wang, and G. Yan (June 15, 2016). "An Easy-to-Use Airborne LiDAR Data Filtering Method Based on Cloth Simulation." In: *Remote Sensing* 8.6, p. 501. DOI: 10.3390/rs8060501. URL: http://www.mdpi.com/2072-4292/8/6/501 (visited on 01/16/2020) (cit. on p. 21).

LIST OF PUBLISHED CONTRIBUTIONS

Guiotte, F., G. Etaix, S. Lefèvre, and T. Corpetti (2020). "Interactive Digital Terrain Model Analysis in Attribute Space." In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLIII-B2-2020, pp. 1203–1209. DOI: 10.5194/isprs-archives-XLIII-B2-2020-1203-2020. URL: https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLIII-B2-2020/1203/2020/.

Guiotte, F., M. B. Rao, S. Lefèvre, P. Tang, and T. Corpetti (2020). "Relation Network for Full-Waveforms LiDAR Classification." In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLIII-B3-2020, pp. 515–520. DOI: 10.5194/isprs-archives-XLIII-B3-2020-515-2020. URL: https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLIII-B3-2020/515/2020/.

Guiotte, F., S. Lefèvre, and T. Corpetti (May 2019a). "Attribute Filtering of Urban Point Clouds Using Max-Tree on Voxel Data." In: *Mathematical Morphology and Its Applications to Signal and Image Processing*, pp. 391–402. DOI: 10.1007/978-3-030-20867-7_30. URL: https://hal.archives-ouvertes.fr/hal-02343890 (cit. on p. 105).

– (2019b). "Filtrage et classification de nuage de points sur la base d'attributs morphologiques." In: *Journées francophones des jeunes chercheurs en vision par ordinateur (ORASIS)*, p. 9.

– (2019c). "Rasterization Strategies for Airborne LiDAR Classification Using Attribute Profiles." In: *2019 Joint Urban Remote Sensing Event (JURSE)*. IEEE, pp. 1–4. DOI: 10.1109/JURSE.2019.8808945. URL: https://hal.archives-ouvertes.fr/hal-02343901/document (cit. on pp. 46, 63, 64, 71, 77, 105).

– (2019d). "Stratégies de rastérisation pour la classification de données LiDAR aéroportées par profils d'attributs morphologiques." In: *Colloque GRETSI sur le Traitement du Signal et des Images*, p. 5.

– (2019e). "Voxel-Based Attribute Profiles on Lidar Data for Land Cover Mapping." In: *IEEE International Geosciences and Remote Sensing Symposium (IGARSS)*. Yokohama, Japan. DOI: 10.1109/IGARSS.2019.8899129. URL: https://hal.archives-ouvertes.fr/hal-02343963.

Guiotte, F., M.-T. Pham, R. Dambreville, T. Corpetti, and S. Lefèvre (Jan. 2020). "Semantic Segmentation of LiDAR Points Clouds: Rasterisation beyond Digital

Elevation Models." In: *IEEE Geoscience and Remote Sensing Letters*. DOI: 10.1109/LGRS.2019.2958858. URL: https://hal.archives-ouvertes.fr/hal-02399410 (cit. on p. 105).

ACRONYMS

AGB Above Ground Biomass. 14

ALS Airborne Laser Scanning. 13–15, 22, 29, 119

AP Attribute Profile. 27–29, 35, 36, 40, 41, 43, 91, 96–100, 118, 119, 147

CNN Convolutional Neural Network. 22

CRF Conditional Random Field. 17, 18

CSF Cloth Simulation Filter. 21

DAP Derivative of Attribute Profile. 97

DEM Digital Elevation Model. 19, 21, 29, 33, 35, 36, 38, 41, 58, 61, 63, 65, 67, 69, 71, 79, 95, 117, 119, 145

DSDAP Derivative of Self-Dual Attribute Profile. 36, 38, 41–43

DSM Digital Surface Model. 19, 21, 29, 40, 41, 119

DTM Digital Terrain Model. 19–21, 29, 39, 45, 47–52, 54, 56, 58, 59, 146

EP Extinction Profile. 27, 29

ESDAP Extended Self-Dual Attribute Profile. 36, 40

FIFO First-In-First-Out. 25

FLANN Fast Library for Approximate Nearest Neighbors. 17

FP Feature Profile. 27, 119, 147

FWF Full-Waveforms. 12, 13, 16, 22

GIS Geographic Information System. 145

GPS Global Positioning System. 12

HAP Histogram-based Attribute Profile. 27

IGN *Institut National de l'Information Géographique et Forestière.* 4

IMU Inertial Measurement Unit. 12

IRISA *Institut de Recherche en Informatique et Systèmes Aléatoires.* 5

LETG *Laboratoire Environnement Télédétection et Géomatique.* 5

LFAP Local Feature-Based Attribute Profile. 27, 43, 147

LFSDAP Local Feature-Based Self-Dual Attribute Profile. 147

MLS Mobile Laser Scanning. 14, 22, 28, 29, 84

MP Morphological Profile. 26, 27, 29

MRF Markov Random Field. 18

NDBI Normalized Difference Built-up Index. 21

nDSM Normalized Digital Surface Model. 21

NDVI Normalized Difference Vegetation Index. 21

OSU Observatories of Universe Sciences. 4

PCA Principal Component Analysis. 17

RF Random Forest. 17, 18, 64

SAP Simple Attribute Profiles. 8, 118, 145, 147

SDAP Self-Dual Attribute Profile. 27, 36, 38, 40–43, 147

SE Structuring Element. 21, 23, 26, 28

SMRF Simple Morphological Filter. 20

SVM Support Vector Machine. 17, 21

TLS Terrestrial Laser Scanning. 14

UAV Unmanned Aerial Vehicle. 15

Appendices

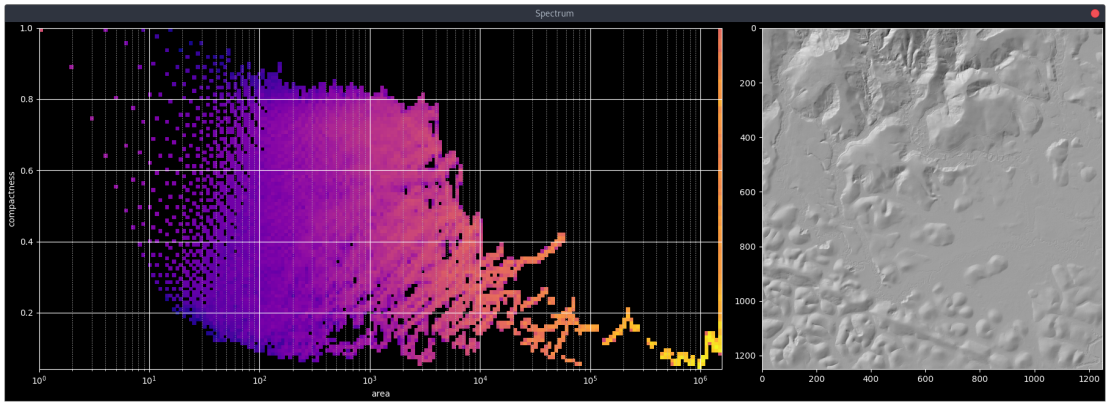
APPENDIX A.

INTERACTIVE DTM ANALYSIS IN ATTRIBUTE SPACE: A USE CASE

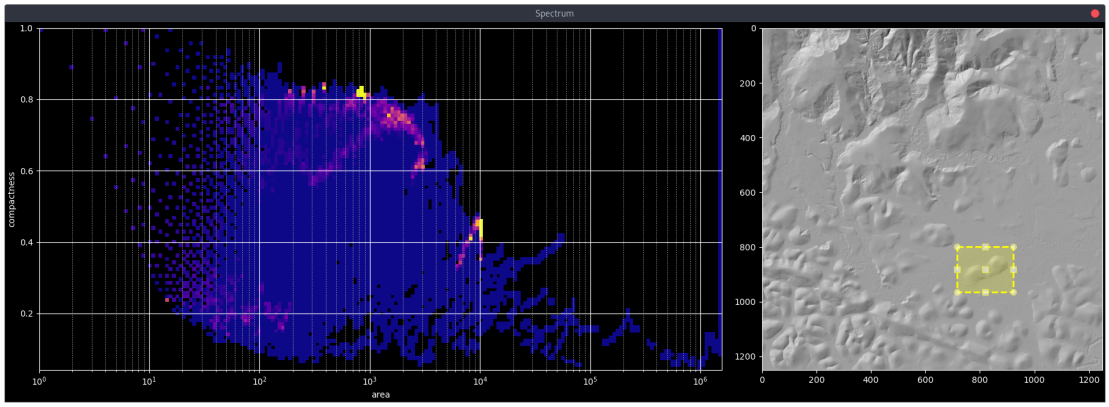
This appendix aims to describe a real-world application of morphological hiearchies and LiDAR data. It was conducted through a collaboration with the TELLUS Environment SME. The proposed workflow allows us to build an attribute space (of shapes and sizes) from any tree representation of a DEM (or any other LiDAR feature). From this attribute space, we then compute and display the 2D histogram of probability distribution of structures (a.k.a. the pattern spectrum), see Chapter 4 for more in-depth details. Let us point out that some interactive analysis can be performed from the attribute space to the DEMs space, and from the DEMs space to the attribute space, as shown in Figure A.1 Page 146.

The software backend needed to build the trees and compute the density probabilities is entirely managed by the package SAP. This current user interface is a proof of concept realized with the Python package *Matplotlib*[1] and *JupyterLab*[2]. Nevertheless, this proof of concept is already fast and fully interactive. The final user interface should be implemented to a more professional Geographic Information System (GIS) software. TELLUS Environment has planned to release the software "when it's ready" as an open source add-on for *QGIS*[3].

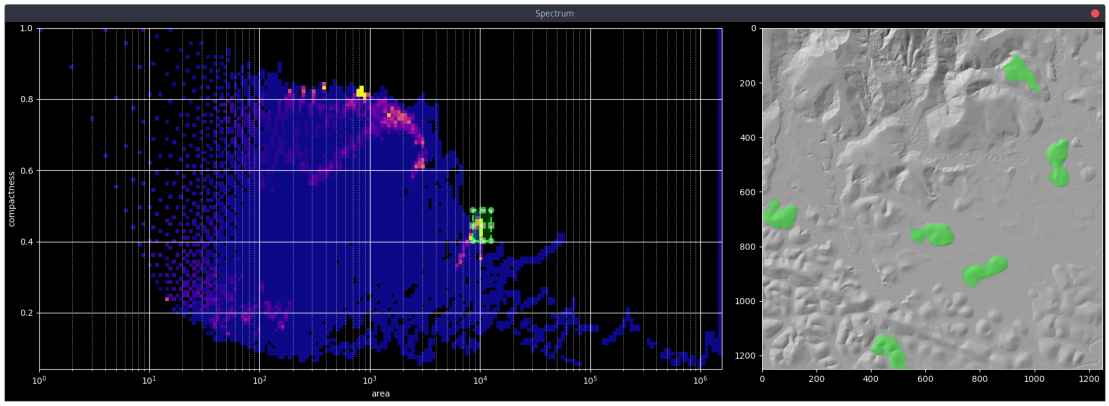
[1] https://matplotlib.org/
[2] https://jupyter.org/
[3] https://qgis.org/



(a) First, the user select the LiDAR feature to display, the tree (max/min-tree, tree of shapes, …) and two attributes in drop-down menus (not shown here). The LiDAR feature is displayed (with a hill-shade rendering) on the right. The pattern spectrum of the two attributes is computed and displayed on the left (area and compactness in this example).



(b) Here the user choose to make a rectangular selection on the DTM (right, in yellow), the corresponding nodes then activates in the pattern spectrum (left): blue means that 0% of the node bin is in the selection, yellow means that 100% of the node bin is in the selection.



(c) The user finally make a rectangular selection on the spectrum (left, in green) to select structures of area around 10 000 m^2 and compactness between 0.4 and 0.5. The corresponding structures are highlighted in green over the DTM (right).

Figure A.1.: Reproduction of user interactions with the application.

CLASSIFICATION OF REMOTE SENSING DATA WITH MORPHOLOGICAL ATTRIBUTES PROFILES: A DECADE OF ADVANCES

This appendix consists in a survey looking back over the last ten years with more than 100 APs related contributions on remote sensing data. A complete experimental comparison of AP extensions is provided.

This paper was written in collaboration with Deise Santana Maia, Minh-Tan Pham, Erchan Aptoula and Sébastien Lefèvre. The two main authors are Deise Santana Maia and Minh-Tan Pham. The experiments where carried with the SAP package for which I gave support on the experimental part. For this work I have implemented APs (and max/min APs), SDAPs, LFAPs, Local Feature-Based Self-Dual Attribute Profiles (LFSDAPs), FPs and I have especially added some partition tree profiles, namely α-APs and ω-APs with the help of Deise.

All these descriptors are currently available in SAP. This survey is currently under revision in a major journal in remote sensing.

Classification of remote sensing data with morphological attributes profiles: a decade of advances

Deise Santana Maia, Minh-Tan Pham, Erchan Aptoula, Florent Guiotte, Sébastien Lefèvre

***Abstract*—Morphological attribute profiles (APs) are among the most prominent methods for the spatial-spectral pixel analysis of remote sensing images. Since their introduction a decade ago to tackle land-cover classification, many studies have been contributed to the state-of-the-art, focusing not only on their application to a wider range of tasks, but on their performance improvement and extension to more complex earth observation data as well. Despite the overwhelming proliferation of deep learning-based methods in the past five years, APs are far from obsolete, due mainly to their high flexibility, low computational cost, lower training data requirement, and rigorous mathematical foundation. In this survey, an entire decade of more than 100 AP related contributions to the field of remote sensing have been compiled, providing an extensive panorama of this robust and effective tool. Moreover, a collective experimental comparison of the reviewed AP variations is provided as well, not only in terms of classification performance, but for the first time in terms of their generalization capacity too.**

***Index Terms*—Mathematical morphology, attribute profiles, multilevel image description, image classification, remote sensing.**

I. Introduction

Classification constitutes one of the paramount tasks of remote sensing image analysis for earth observation. Its performance is critical for the success of land-use and land-cover mapping and monitoring. The rapid development of high resolution (HR) and very high resolution (VHR) image acquisition technologies, has led to increasingly more complex images and higher levels of detail. Consequently, the importance of the joint exploitation of spatially contextual information along with spectral pixel characteristics has become clear early on [1].

Morphological profiles (MPs) [2] were introduced almost two decades ago to address this exact issue. In essence, they produce multi-scale descriptions of their input, through the application of a sequence of morphological reconstruction based filters using structuring elements (SEs) of various sizes (and shapes) [2]. Their rigorous mathematical foundation, and inherent ability to capture spatial-spectral information, has led to the development of several variants [3]–[6].

However, as spatial resolutions and image sizes progressively increased, their relatively high computational cost has been drastically accentuated. Moreover, their initially celebrated capacity of capturing size and shape variations, became eventually extremely limited, when confronted with thematic classes differing in terms of alternative properties, such as contrast, homegeneity, etc. In an effort to overcome the aforementioned shortcomings, MPs have been generalized into morphological attribute profiles (APs) through the seminal work of M. Dalla Mura, J. A. Benediktsson, B. Waske and L. Bruzzone [7].

On the contrary of MPs, APs employ attribute filters (AFs) [8], a powerful class of connected morphological filters, capable of removing entire connected components w.r.t. arbitrarily defined attributes (e.g. geometric, statistical, etc), thus eliminating the size/shape limitation of MPs. Furthermore, APs can be generated efficiently through quasi-linear [9] and parallel [10] algorithms, through their input's tree-based hierarchical representation [11], thus equipping them with a high level of scalability; an invaluable property in remote sensing, where gigapixel images are becoming the norm. In fact, [12] presents an application of AP at a terapixel scale!

It is thus not surprising that since their introduction ten years ago, a great number of AP related publications have appeared (4540 at Google Scholar as of June 2020), tackling various aspects of remote sensing image analysis. Many among them have been dedicated to improving further APs through a rich variety of extensions, focusing on every single stage of their calculation.

Even though recent years have witnessed the overwhelming proliferation of deep learning [13], overshadowing performance-wise most non-deep feature extraction methods, APs continue to withstand the test of time. Besides their aforementioned invaluable properties, this is additionally due to their ability to perform even with limited amounts of training data as well as their capacity to accommodate arbitrary modalities and challenging types of images, so often encountered in the remote sensing domain.

This article[1] presents a survey on APs and contributes to the state of the art in the following ways:

1) We provide a comprehensive review of ten years of advances on APs, by decomposing their calculation into four stages, and grouping the reviewed studies accordingly. Evidently, this is not the first survey on APs [14], [15]. Contrary to [14] from 5 years ago, this survey is not limited with the vanilla definition of APs and their focus on hyperspectral data, and compared to [15], it additionally addresses APs on partition tree structures, threshold selection techniques and AP post-processing.
2) We present the results of an extensive series of classification experiments with multiple real datasets, intended

[1]A short version of this article has appeared at ICPRAI 2018

to measure the performance of the various reviewed AP variants. However, the experiments have been conducted for the first time in terms of connectivity type as well as spectral quantization level.

3) More importantly, we present the results of pixel classification experiments intended to measure not only the performance of the various reviewed AP variants, but their generalization capacity as well. Being in the Big Data era, generalization is a core and sought after property of any content description tool, and one for which AP have been criticized for, as an image's tree representation contains both training and testing elements [16]. Mixing training/testing sets is unfortunately not uncommon with deep learning either [17]. We underline this validation malpractice, often encountered in the state of the art and propose a solution through an image's spatial subdivision and independent tree construction.
4) In order to promote reproducible research and ease generalization, we also provide the first available open-source library for APs (SAP - Simple Attribute Profiles)[2].

In the remainder of this article, we first recall the theoretical background of APs and highlight the key stages of AP construction (Sec. II). Then we present and discuss the plethora of AP oriented developments that have taken place in the last ten years (Sec. III). Next, we present the results of our experimental study (Sec. IV) providing an extensive evaluation of some of the reviewed AP extensions, compared to the original APs. To this end, we have employed two publicly available real remote sensing image datasets and used an open source library, thus guaranteeing reproducibility. Then, we present a critical analysis of the experimental settings commonly used by AP-based classification approaches (Sec. V), underline a validation malpractice involving the mixing of training and testing data and propose an amendment. Sec. VII is devoted to concluding remarks and future research directions.

II. Principle of APs

APs are multilevel image description tools obtained by successively applying a set of morphological attribute filters (AFs) [7]. Unlike usual image filtering operators which are directly performed on pixel level, AFs work on connected component (CC) level based on the concept of image connectivity. In particular, AFs are applied on CCs with regard to a predicate based on an arbitrary statistical or geometric property thereof. Consequently, they exhibit a higher level of flexibility w.r.t. operators by reconstruction, that are severely limited with characterizing only the size and shape of their input. This advantage naturally extends to APs vs MPs as well [2], [7].

The generation of APs [7] from an input image can be summarized as a four-step process (see Fig. 1):

1) Construction of the image's hierarchical tree representation, where CCs are denoted as nodes.
2) Computation of one or more relevant attributes describing the geometrical and statistical features from each tree node.
3) Filtering the tree by preserving/removing nodes according to their attribute values compared against predefined thresholds.
4) Reconstruction of the image from the filtered tree.

Step (1) can be performed using different pixel connectivity rules. For 2D images, 4 and 8-connectivity are the most common. Steps (3) and (4) can be implemented for different attributes (with different threshold values) to finally produce a set of filtered images (by stacking them) forming the APs.

More formally, according to the seminal work of [7], given a grayscale image $X : E \to \mathbb{Z},\ E \subseteq \mathbb{Z}^2$, the calculation of APs on X is achieved by applying a sequence of AFs based on a min-tree (i.e. attribute thickening operators $\{\phi_k\}_{k=1}^{K}$) and on a max-tree (i.e. attribute thinning operators $\{\gamma_k\}_{k=1}^{K}$) as follows:

$$\begin{aligned}\mathrm{AP}(X) = \Big\{&\phi_K(X), \phi_{K-1}(X), \ldots, \phi_1(X), X,\\ &\gamma_1(X), \ldots, \gamma_{K-1}(X), \gamma_K(X)\Big\},\end{aligned} \tag{1}$$

where $\phi_k(X)$ denotes the filtered image obtained by applying the attribute thickening ϕ with regard to the threshold k. Similar explanation is made for $\gamma_k(X)$. As observed, the resulting $\mathrm{AP}(X)$ is a stack of $(2K+1)$ images including the original image, K filtered images from the thickening profiles and the other K from the thinning profiles.

A toy example of AFs is presented in Figure 2. Given the grayscale image $X : E \to [0, 1, 2]$ of Figure 2(a), we first obtain the max-tree T of X using 4-connectivity. Then we compute the area (number of pixels) of the nodes of T. Subsequently, we prune the nodes of T with area less than a given parameter k. In our case, k is equal to 8 and the nodes composed of less than 8 pixels are pruned from the tree. Finally, we reconstruct the image from the pruned tree, resulting in X'. Hence, X' is the area thinning of X for $k = 8$.

III. Recent advances from APs

Each of the aforementioned four AP construction stages have received various forms of extensions and contributions from the scientific community. Moreover, AP based image analysis pipelines include often some form of pre-processing, usually in order to adapt multi-band input (since Eq. (1) expects single-band data) as well as post-processing steps to increase description capability.

We now revisit the recently proposed developments that have provided significant contributions to the AP framework for remote sensing image classification (Sec. III-A to Sec. III-E). Here, our study will focus on the following key concepts:

- the adaptation of APs to various modalities besides single-band images, and in particular to multi-band data (Sec. III-A);
- the construction of APs using various hierarchical image representations (Sec. III-B);
- the determination of attributes and thresholds (Sec. III-C);

[2]https://gitlab.inria.fr/fguiotte/sap

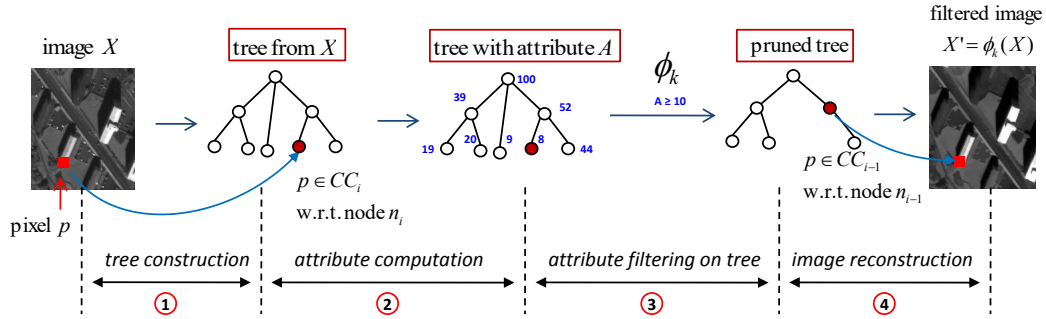
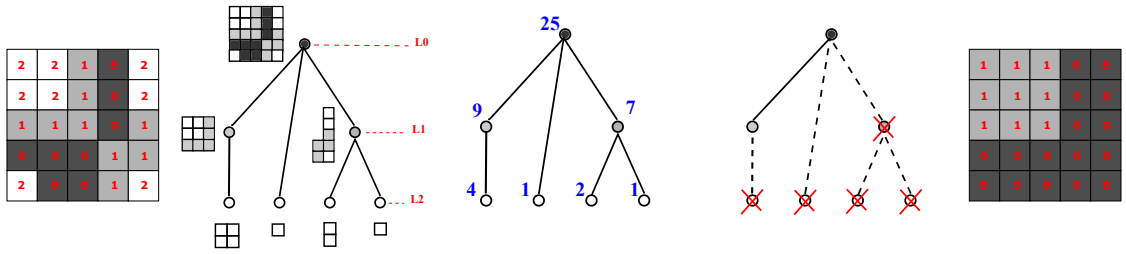


Fig. 1: The AP generation framework which involves four main stages: tree construction, attribute computation, tree-based attribute filtering (pruning) and image reconstruction from filtered (pruned) tree. This figure is adopted from [18].



(a) Image (b) Max-tree (c) Area attribute (d) Area filtering (e) Reconstruction

Fig. 2: (a) Original grayscale image $X : E \rightarrow [0, 1, 2]$. (b) Max-tree T of X computed using 4-connectivity. (c) Area (number of pixels) of the nodes of T. (d) Area filtering on T with regard to the threshold $k = 8$. (e) Image X' reconstructed from the pruned tree.

- the tree filtering rules used in the construction of APs (Sec. III-D);
- post-processing techniques intended for feature enhancement (Sec. III-E).

For other AP related notions, including profiles computed with different morphological filters and image reconstruction techniques, we refer readers to the related references for additional details (Sec. III-F).

A. Input data

Since APs were originally proposed to deal with only single-band images [7], their adaptation to other kinds of remote sensing data became necessary early on. In particular, their adaptation to multi-channel images (multispectral and hyperspectral) became an intensive research topic. The main idea in this regard has been to reduce the various and often correlated image bands into fewer components through some dimension reduction tool, and extract APs independently from each of them, followed by their subsequent merging.

The initial extension of APs to hyperspectral images was proposed in [19], namely Extended AP (EAP), which consists in employing the principal component analysis (PCA) to this end; an unsupervised yet often sub-optimal tool due to its linear nature. Alternatives to PCA that have been additionally studied include the independent component analysis (ICA) [20], [21], the kernel PCA (KPCA) [22], the feature space discriminant analysis (FSDA) [23] and other supervised methods such as the discriminant analysis feature extraction (DAFE) [24], the non-parametric weighted feature extraction (NWFE) [25], the sparse Hilbert-Schmidt independence criterion and Surrogate kernel (HSIC) [26] among others.

Given that independent AP calculation from each band (or each image component, assuming dimension reduction has been applied to the image bands), ignores any and all correlational information among them, [27] have proposed a vector alternative calculating APs collectively and simultaneously from all available bands/components. The core idea relies on representing all bands through a single hierarchical representation, leading to vector APs (VAPs).

It would be also possible to compute APs on derived features of optical remote sensing data, such as edge/contour information obtained by Sobel gradient filtering of panchromatic images [10], or NDVI (Normalized difference vegetation index) from multispectral images [28] for urban and crop field classification. Then, to deal with VHR optical images where highly textural information becomes significant, another adaptation of APs was proposed in [29]. The raw input data is replaced by its textural features, thus considerably improving classification performance on the tested textured images. Furthermore, although APs are usually explored for supervised classification of optical images (either panchromatic,

multispectral or hyperspectral), some studies have investigated them within unsupervised scenarios. For example in [30], the authors exploited the Differential APs (i.e. which compute the difference between successive APs to form differential profiles) for unsupervised anomaly detection in hyperspectral images , where they observed that the anomalies and the background of an image are enhanced in the thinning and thickening profiles, respectively. Some other examples are the retrieval of building height using panchromatic angular images [31] and the change detection in temporal panchromatic images [32].

While the application of APs to optical remote sensing data has been strongly focused on, alternative remote sensing image types have received far less attention. One may witness some tentative works on SAR (Synthetic Aperture Radar) and polarimetric SAR images for segmentation [33], building detection [34], crop field and land-cover classification [35], [36] and change detection [37]–[39] using the original APs and the Differential Attribute Profiles; on passive microwave remote sensing image analysis [40]; on LiDAR data for building detection [41] and land cover classification [16], [28], [42]–[44]; on satellite image time-series classification using Sentinel-2 data [45], [46]; on the fusion of APs and Extinction Profiles (a variant of AP that will be discussed in Sec. III-F) of hyperspectral and LiDAR data using composite kernel SVM [47], [48] and deep learning approaches [49], [50] for land cover classification. This is still an open topic for on-going and future research.

B. Tree construction

Although attribute filters have been introduced more than 20 years ago [8], the relatively late popularity of APs is mainly due to computational issues revolving around the efficient calculation of connected components, that were resolved up to a significant degree through the tree representation of images [11]. They are of paramount importance for the computation of APs, since the trees need to be computed only once and then multiple filtering outputs can be derived easily from them.

Even though the seminal work of Dalla Mura et al. [7] relies on component trees for the implementation of APs, the type of the tree is independent from the rest of the procedure, thus component trees can be replaced by alternative tree types. Consequently, and not surprisingly, there exists a number of reported works exploring such options; e.g. tree of shapes, alpha-trees and omega-trees, each with its own set of properties. Despite a plethora of tree representations for modeling connected components in mathematical morphology (of which not all have yet been implemented with the purpose of AP construction), they can be all categorized into *inclusion* and *partitioning* hierarchies. We invite interested readers to refer to the recent comprehensive survey of partition and inclusion hierarchies of images conducted by Bosilj et al. [51].

Inclusion hierarchies constitute partial partitions of a given image with nested supports and their components are formed by creating, inflating and merging image blocks [52]. Moreover, inclusion trees require the presence of a total ordering relation imposed on the set of image pixel values, which evidently renders their extension to multivariate images challenging, since ordering vectors is not straightforward.

Partitioning hierarchies on the other hand constitute full partitions of a given image, where the leaves of the hierarchy form the finest partition and are iteratively merged until a single root node is formed. In addition, partitioning hierarchies, or simply partitioning trees, most often require only a similarity metric for determining the merging or not between neighboring components, hence making them particularly suitable for processing multivariate images.

The vast majority of the reported work on APs rely on *max-* and *min-trees* that belong both to the category of inclusion trees. More formally, given a grayscale image $X : E \rightarrow \mathbb{Z}, E \subseteq \mathbb{Z}^2$, its upper-level sets are defined as $\{X \geq t\}$ with $t \in \mathbb{Z}$ (resp. lower-level sets $\{X \leq t\}$), i.e. the set of images obtained by thresholding an image at all possible values of their pixels. The connected components ($CC \subseteq E$) composing the upper or lower level sets are referred to as peak components. These two tree types (that are dual w.r.t. complementation) model the inclusion relations between these peak components, thus max-trees are excellent for modeling regions that are brighter than their surrounding, and min-trees respectively for regions that are darker. That is why besides we employ both attribute thinnings (i.e. max-tree filtering) and thickenings (i.e. min-tree filtering) during the construction of APs (Eq. (1)).

As the construction of two trees per image is both memory-wise and computationally inefficient, and results in longer feature vectors per image, thus also affecting classification performance and complexity, self-dual APs (SDAPs) were introduced to target these issues [53]. More specifically, they rely on the use of the *Tree of Shapes* (ToS) [54], which has been designed in order to provide a unified representation for both bright and dark image structures. The ToS is constructed by filling the holes of the aforementioned peak components, and the shapes represented by the nodes do not intersect and are either disjoint or nested. Consequently, ToS is an inclusion hierarchy, which on the contrary to the component tree is also contrast-invariant and self-dual. SDAPs have been empirically shown to outperform APs consistently in terms of classification performance, while also producing shorter feature vectors per pixel [25], [53], [55].

Motivated by the success of ToS, as well as by the useful qualities of partitioning trees in this context aforementioned, α- and ;ω-trees were recently applied to AP implementation [56]. In particular, the α-tree is constructed based on the local range, where every tree node corresponds to an α-connected component (α-CC) [57]. For instance, for $\alpha > 0$, an α-CC is defined as the CC of the maximal size such that only the neighboring pixels with gray level difference less or equal to α are considered connected.

Although α-trees can lead efficiently to a complete and self-dual image representation, due to however the locality of the metric used, gray level variations within α-CCs can be much higher than α (i.e. "the chaining effect" [57]). This undesirable effect can be mitigated through the use of ω-trees [58] which constitute a subset of α-trees, constraining every α-CC with their global range (i.e. the maximal dissimilarity between

any two pixels belonging to that component). The hierarchy remains self-dual, complete and capable of capturing regions of low, intermediate and high gray levels, but global range provides better grouping per level than just a local measure. Lefèvre et al. [59] have focused on the similarity metric requirement of partitioning trees for extension to multiband images, and have used metric learning in order to adapt them to hyperspectral images. Bosilj et al. [56], [60] on the other hand, have empirically tested APs, SDAPs, α- and ω-APs against each other, and established the superiority of SDAPs in noise-free conditions, while partitioning tree based profiles outperformed their inclusion tree counterparts in terms of noise robustness.

C. Attribute and threshold selection

APs possess undoubtedly many desirable practical and theoretical properties that render them particularly suitable for the task of spatial-spectral description in our context. However, they are not without flaws, and the main source of their criticism so far has been in terms of their sensitivity to parameter selection [61]–[63], and by parameters we mean particularly the *attributes* employed to characterize every tree node, and most importantly *the set of associated threshold values*. The attribute and threshold selection parameters will be discussed in the subsections III-C1 and III-C2, respectively.

1) Attributes: From a theoretical point of view, any function $a : \mathcal{P}(E) \rightarrow \mathbb{R}$, where $\mathcal{P}(E)$ denotes the powerset of E, computable on an arbitrary collection of pixels, can be in fact employed as an attribute for AP construction. In practice, it is used during filtering by comparing a given connected component's ($CC \subseteq E$) attribute value against a predetermined threshold in the form of a binary predicate (e.g. in case of area, "is the connected component's area greater than 300 pixels?"). It thus provides a great degree of freedom as far as the object based analysis of an image is concerned. The pioneering paper of APs [7] has introduced four such attributes: *area*, *moment of inertia*, *diagonal length of bounding box* and *standard deviation* where the first three describe a geometric property related to the shape of the tree node under study and the last, its statistical pixel intensity distribution.

Although the aforementioned four attributes are by far the most widely encountered in the state of the art, APs can accommodate (from a theoretical point of view) a vast pool of attributes. Examples include entropy, homogeneity [7], as well as the diameter of equivalent circle and area of convex hull for automatic threshold selection [64]; complexity (perimeter over area) [65]; perimeter and area of bounding box used to evaluate threshold-free APs [66]; solidity (area over area of convex hull) and orientation (between the major axis of the convex hull and the x-axis) [67]; Cov (Coefficient of variation) and NRCS (Normalized Radar Cross Section) tailored for SAR images [33], [39], where Cov is the ratio of the standard deviation divided by the mean value of pixel intensities, and NRCS, expressed in decibels, is the radar cross section per unit area of surface. Furthermore, in [68], it has been observed that when dealing with multiband input, one can extend the pool of attribute measures to include multi-dimensional functions exploiting all available bands simultaneously and two new attributes have been proposed: higher-dimensional spread and dispersion.

As far as the selection of attributes is concerned, there is no straightforward rule nor a limitation. Formally, one should use the attributes that are "meaningful" for the data under consideration; or in other words, the attributes with respect to which the objects of interest differentiate themselves from the rest. In cases where this cannot be determined or known a priori, it is "customary" to most often combine attributes that are expected to provide complementary information, e.g. some geometrical (such as area or moment of inertia) and some statistical (such as standard deviation), known as Extended Multi Attribute Profiles (EMAP) [7]. Nevertheless, in order to tackle the potentially long feature vectors resulting from the combination of various attributes, the application of dimension reduction methods is not uncommon [62].

2) Thresholds: For optimal performance of any given attribute, the set of corresponding thresholds is supposed to span the range of values between which lie the attribute values of the tree nodes representing the objects of interest. In the presence of objects of interest of varying scales however, as it is often the case in practice, the size of the threshold set (or the subdivision level of the said range) also becomes of paramount importance so as to capture their response through attribute filters. An example for illustrating this principle is the case of distinguishing minerals of various sizes with sieves; and evidently, if you only possess two sieves (one large and one very small), you cannot expect to detect minerals of in-between sizes. Consequently, it is no surprise that the selection of the threshold set has a profound effect on the description performance of APs [27]. Although using a wide threshold range with a very fine subdivision level, might at first attempt seem as an intuitive counter-measure against this issue, it is to be avoided. Since, it will not only trigger the Hughes phenomenon, but also provides no guarantee of capturing the differences among the various scales of the objects of interest.

During the early years of APs, threshold sets were determined exclusively manually based on expert knowledge, usually with four thresholds per attribute [7], [20], [53]. Even though some attributes such as moment of inertia possess many desirable invariance properties (against scale, rotation), rendering them and their manual threshold sets robust against content variations, unfortunately most attributes are heavily affected by the spatial, spectral resolution as well as size of the objects and regions of interest.

Consequently, it was not long before attempts started being made on solving this issue. One line of research focused on developing analytical approaches based on expert knowledge, in the form of equations for producing thresholds. For instance [61], [63] have proposed such equations for area depending on the input's spatial resolution as well as for standard deviation based on the image's mean pixel intensity, where the number of thresholds however is still user-defined.

Further approaches relying on (semi-)supervised learning were put forward in [69], that proposed clustering the attributes of a given tree presentation into a user-defined number of

clusters, while [70] investigated the evolutionary optimization of a very large set of thresholds. Then, [71], [72] presented a framework based on granulometric characteristic functions, where they employ morphological filters in order to first assess the input image's content and determine the corresponding thresholds adaptively.

More recently, unsupervised threshold selection methods [64], [73] and threshold-free APs [66], [74], [75] have been proposed. In [73], from the AP computed from all possible threshold values of an attribute, a genetic algorithm selects the subset of reconstructed images which convey the highest amount of information. A disadvantage of this technique is the expensive computation of APs for all possible threshold values. Then, a more efficient threshold selection method is proposed in [64]: first, the set of attribute values of all nodes of a component tree are sorted in non-decreasing order; then, an increasing curve is built from the sorted attribute values; finally, the points in the curve with the largest gradient values are the selected threshold values. In [74], [75] a threshold-free approach was introduced, where every tree node is described based on simple statistical properties of the sequence of attribute values belonging to the nodes in the path connecting the node under study to the root of the tree. Properties such as the highest change of attribute value have shown promise. This strategy removes the need for thresholds. Moreover, by no longer employing the same global thresholds for every node, it instead uses a node-adaptive description strategy.

D. Tree filtering

In terms of tree filtering, relatively few developments have taken place since the inception of APs. Nevertheless, this subsection provides an overview of filtering rules for the sake of completeness.

Tree filtering is the stage where attribute filtering is performed to a given image using its tree representation. More specifically, given an attribute and a corresponding threshold value, one removes certain nodes (either single nodes or entire branches) of the tree that do not satisfy the threshold according to some predefined strategy; such as *Max*, *Min*, *Direct*, *Viterbi* and *Subtractive* [7].

The effect of the selected filtering strategy depends on the increasingness (or not) of the attribute under study. Formally, an attribute $a(\cdot)$ assessed on region CC_i is said to be increasing if the following property holds [76]:

$$\forall\, CC_1 \subseteq CC_2, \Rightarrow a(CC_1) \leq a(CC_2) \tag{2}$$

Common increasing attributes include area and diagonal length of the bounding box. When the attribute is increasing, filtering is straightforward. More specifically, if a node does not satisfy the underlying predicate, it is removed along with all its descendants, since increasingness guarantees that they do not satisfy the predicate either. After removal, a node's pixel values become those of its highest ancestor node that satisfies the predicate. In this case, *all of the aforementioned filtering strategies lead to the same exact outcome*.

On the other hand, if the attribute is not increasing (e.g. moment of inertia, standard deviation, etc), then filtering is no longer straightforward, as whether a node's descendants need to be removed or not, can no longer be determined by the node under study alone. At this point one of the following is used [11]:

1) *Max*: prunes the nodes along a branch starting from the leaves up until the first node that satisfies the predicate and needs to be preserved.
2) *Min*: prunes the nodes along a branch starting from the leaves up until the last node that does not satisfy the predicate and needs to be removed.
3) *Viterbi*: relies on dynamic programming through the Viterbi algorithm. It formulates filtering as an optimization problem in terms of node removal and preservation costs, which solves for minimal cost.
4) *Direct*: consists simply in removing the nodes that do not satisfy the underlying predicate. Its eventual descendants are transferred to the first ancestor node that satisfies the predicate and thus needs to be preserved. Although simple, the direct filtering strategy is notoriously known [77] for its difficulty in dealing with shape based object analysis. This is mostly due to contrast loss related issues, which also constitutes the main motivation behind the design of the following subtractive strategy.
5) *Subtractive*: it behaves almost identically to the direct strategy, with the only difference being that after removing a node not satisfying the underlying predicate, an additional propagation step is performed on the descendant nodes. In particular, the pixel intensity associated with the descendant nodes is lowered in the case of max-tree (and increased in the case of min-tree), so that their contrast with respect to the local background will remain consistent once removal takes effect. In the case of tree of shapes, besides lowering the pixel intensity of the remaining descendant nodes, the subtractive rule can introduce new intensity values that were not present in the original image [55], [78].
   As far as partitioning trees are concerned, the reader is referred to [56]. The subtractive strategy has been shown to outperform its alternatives when dealing with non-increasing attributes, especially with moment of inertia [77].

The reader is referred to [55], [78] for the results of empirical comparison between filtering strategies.

E. Post-processing of output profiles

APs, i.e. the sequence of filtered images in Eq. (1), can be directly fed into supervised classifiers such as SVM or Random Forest for classification on a pixel basis. Such direct application has provided better performance compared to MPs [7] in terms of classification accuracy as well as computational cost. However, since APs often lead to feature vectors with a relatively high redundancy level, depending on the number of employed thresholds, [24], the post-processing of these features has been addressed in several studies. Many among them have proposed to apply different feature selection techniques to extract more informative features and reduce their dimension. In [24], [61], both linear (PCA, ICA) and

nonlinear methods (ICA, KPCA, DAFE, DBFE, NWFE, etc.) have been investigated. A general framework as well as a systematic survey on spatial-spectral approaches combining APs with these feature selection techniques has been presented in [14].

Other works have focused on extra spatial processing of APs for better characterization of structural and textural information from the image content. Recent studies [79], [80] claim that when dealing with VHR remote sensing images where regions and objects appear more heterogeneous, APs may not provide a complete spatial characterization of pixels. Therefore, some efforts have been realized to improve APs through the histogram or some first-order statistical features of the local patch around the pixel under study. As a result, the local histogram-based APs (HAPs) [79], [81] and the local feature-based APs (LFAPs) [80], [82] have been proposed and proved to be more efficient for better dealing with local textures. The extensions of these extra spatial processing methods to self-dual profiles as well as to hyperspectral images have been studied in [80].

Some further notable extensions to APs include their sparse representation in an attempt to increase their description capacity [83]. In detail, through the collection of representative samples of low-dimensional class-dependent structures, any sample can then be sparsely and more effectively represented and classified. Moreover, the combination of APs with classifier ensembles has been also investigated intensively in [84], [85].

Last but not least, we have observed an increasing tendency to combine APs with convolutional neural networks (CNN) in the classification of satellite images. While that CNNs require large training sets in order to provide optimal features from the raw data, APs can produce effective features from scratch thanks to their inherent expert knowledge. CNNs can exploit APs to produce even stronger features from them without the need for large training sets. Hence, APs simplifies the learning process of CNNs by reducing the number of training samples required for a satisfactory classification result which, consequently, reduces training time. On the downside, combining CNNs with APs increases the design and computational complexity of the classification task. For instance, we refer readers to deep learning approaches on APs [86], [87], Extinction Profiles (to be discussed in Sec. III-F) [88] and SDAPs [89]. In those works, spatial features are extracted in two phases: first using APs (and their variants) and then using CNNs. More precisely, pixel features obtained from APs are fed into CNNs, leading to better classification results when compared to the AP and CNN methods individually, while increasing considerably the test time [86], [90].

F. Extensions and generalization of APs

As mentioned previously, the main advantages of APs in comparison to MPs are the efficient computation of APs through hierarchical image representations and the possibility of extracting information other than the ones constrained by the size and shape of structuring elements. However, using APs in remote sensing images also has a few limitations.

First, apart from threshold-free APs [66], [74], [75], the quality of an AP depends on the selected set of thresholds. A bad selection of thresholds can lead to redundant information in the AP [61], [62]. An alternative solution to alleviate this redundancy problem is to replace the attribute filters used in the computation of APs by extinction filters, resulting in *Extinction Profiles (EPs)* [91]. An extinction filter acts on the regional extrema (minima or maxima) of an image: each extremum is either completely preserved or pruned. Let X be a grayscale image. The extinction value (with respect to a given attribute) of any maximum M of X is the maximum attribute value k such that M is still included in a maximum of $\gamma_k(X)$, where γ_k is the thinning operator with parameter k. Similarly, the extinction value of any minimum M of X is the maximum attribute value k such that M is still included in a minimum of $\phi_k(X)$, where ϕ_k is the thickening operator with parameter k. This way, to compute an EP, the filtering parameter is the number of minima or maxima to be preserved instead of a threshold value, which makes EPs less sensitive to image resolution [91].

Since [91], EPs and its extension to hyperspectral images (Extended Extinction Profiles) have been successfully applied to the land cover classification of hyperspectral data [49], [92] and to the fusion of hyperspectral and Lidar data [30]. To further reduce redundancy, composite kernels are used to fuse the spatial information of EPs with hyperspectral data in [93] and with Lidar data in [47].

Another limitation of APs is that, very often, clusters of pixels associated to distinct semantic objects, such as roads and buildings, are connected by narrow paths of similar intensity value. This leads to pixels of different semantic classes being connected throughout several levels of a component tree. Consequently, the attribute values of several connected components describe the union of objects of different classes instead of an object of a single class. In [94], the authors address those problems in the context of hyperspectral image classification. To overcome those issues, attribute connected filters are replaced by partial reconstruction filters, which allows to disconnect regions connected by narrow paths and improve the overall classification accuracy. Then, this idea is explored in [43], which shows the interest of using partial reconstruction in the classification of hyperspectral and Lidar image in comparison to attribute connected filters.

Another related approach, called *Invariant Attribute Profiles (IAPs)*, has been recently proposed in [95] to overcome other limitations of APs, including the sensibility of APs to geometric transformations, like rotation, and to the surrounding of pixels of a same material. Different from APs, IAPs are not computed from a hierarchical representation of the input data. Instead, hierarchical information is indirectly extracted from the original image by performing convolutions of different sizes and by computing the Fourrier transform for different values of Fourrier order. Then, IAPs are obtaining by stacking spatially invariant profiles, obtained from the segmentations of the convoluted images, and frequency invariant profiles, obtained from the histogram of oriented gradients (HOG) of the Fourrier transforms.

Finally, Pham *et al* [96] propose a generalization of APs

called *Feature Profiles (FPs)*. They generalize the step 4) of the generation of APs, which consists in reconstructing an image from a filtered tree. To build an AP, this reconstruction is originally performed by projecting the gray valu nodes of the filtered tree onto the image pixels. In reconstruction step is extended by taking into cons not only the gray values of the nodes, but also other and geometrical features. The resulting images com so called FP. The experiments with remote sensing i [96], [97] show the interest of projecting attributes and moment of inertia in the context of image class Hence, we also consider FPs later in the experimenta

IV. Experimental study

This section describes our experimental study to the performance of the standard APs as well as som recent variants. The contributions of this section are the evaluation of newer variants of APs, and of the connectivity and quantization parameters on the per of APs. This is the first study addressing the connect quantization parameters' effect in this context.

Experiments were mostly performed in Python us licly available libraries [3]. First, APs and some of t ants were computed with the Simple Attribute Profi package[4]. To the best of our knowledge, this is the first available open-source library for computing APs and some of their extensions. The SAP package relies on the Higra [98] library[5], which provides efficient implementation and post-processing of morphological trees in C++. Then, classification was performed with the scikit-learn Python library.

Supervised classification has been conducted on both grayscale and hyperspectral images for the sake of comprehensiveness. In this section, we introduce the datasets, the experimental setup commonly encountered in the state of the art [7], [14], and the classification results that have been obtained.

In more detail, the standard setup involves calculating the hierarchical tree representation (as explained in detail at Section II) from the entire input image, computing the APs (or its variants) from it, and then subdividing the resulting features based on the locations of training and validation/testing pixels. The next section will elaborate on the reasons why this approach constitutes a validation malpractice and propose an alternative strategy for better assessing the generalization capacity of APs.

A. Data description

The experiments have been conducted with two publicly available datasets for the sake of reproducibility. In order to show performance variations depending on the number of spectral bands, one hyperspectral and one panchromatic dataset have been selected.

[3]Source codes are available in https://gitlab.inria.fr/dsantana/attributes-profiles-survey-source-codes

[4]The documentation and source codes of the SAP package are provided in https://gitlab.inria.fr/fguiotte/sap

[5]The documentation and source codes of the Higra package are provided in https://github.com/higra

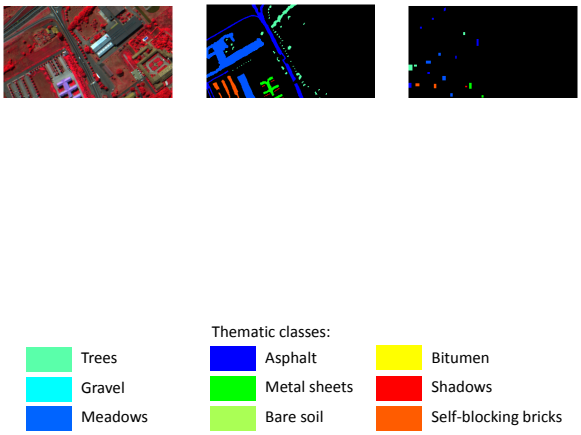


Fig. 3: The 610×340 Pavia University data (left to right: false-color image made by bands 31-56-102, ground truth including nine thematic classes and training set).

1) Pavia University dataset: The Pavia dataset is a hyperspectral image acquired by the ROSIS airborne sensor with 1.3-m spatial resolution over the region of Pavia University, Italy [6]. The image consists of 610×340 pixels with 103 spectral bands (from 0.43 to 0.86 μm) after the noisy bands are removed. The ground truth covers nine thematic classes: trees, asphalt, bitumen, gravel, metal sheets, shadows, meadows, self-blocking bricks and bare soil. For this image, the standard training set[7], composed of 3,921 pixels (see Tab. I), was adopted for the classification task. The test set was composed of all the remaining 40,002 pixels in the ground truth that are not in the training set. In the remainder of this paper, this partition into training/test set of Pavia will be denoted as $Pavia_1$. The reader may note that some works in the literature consider all ground truth pixels as testing pixels, which is not done here in order to provide fair classification results with a non-void intersection between training and testing pixels. The false-color image (made by combining the bands 31, 56 and 102), the ground truth map and the training set are shown in Fig. 3. Following the standard approach of handling hyperspectral images [7], we first performed the PCA on this dataset and the first four PCs (involving more than 99% of the total variance) were preserved for our experiments. The APs and their extensions are computed independently on each selected PCs and are concatenated, leading to EAPs. This way, we assess the family of EAPs and its variants on the Pavia dataset.

2) Gray-Potsdam dataset: The Potsdam dataset is composed of 38 aerial high resolution images, of 6000×6000 pixels, with 5cm spatial resolution over the city of Pots-

[6]The Pavia dataset and its ground truth can be downloaded in http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes

[7]Training set provided by the IEEE Geoscience and Remote Sensing Society (GRSS) Data and Algorithm Standard Evaluation (DASE) website: http://dase.grss-ieee.org/

TABLE I: Number of training and test samples of the standard ($Pavia_1$) and new ($Pavia_2$) partitions of the Pavia dataset.

Class	Number of samples		
	Training set	Test set	Total
Asphalt	548	6,304	6,852
Meadows	540	18,146	18,686
Gravel	392	1,815	2,207
Trees	524	2,912	3,436
Metal sheets	265	1,113	1,378
Bare Soil	532	4,572	5,104
Bitumen	375	981	1,356
Bricks	514	3,364	3,878
Shadows	231	795	1,026

dam, Germany [8]. For each image, the red, green, blue and infrared bands are available. The ground-truth annotations of this dataset cover six thematic classes including impervious surfaces, building, low vegetation, tree, car, and clutter/background (water bodies, tennis courts, swimming pools, etc). Experiments were performed on one image of this dataset, namely *top_potsdam_7_7.tiff*, whose ground truth is composed of several connected components of each of the six classes. To highlight the strength of the spatial information extracted from APs in the context of image classification, we considered a grayscale version of the original RGB image. The original RGB image was converted into a grayscale image using the formula $0.3R+0.59G+0.11B$, which gives an approximation of the luminance in the NTSC color space [99]. The input grayscale image together with its thematic ground truth map are shown in Fig. 4(a) and 4(b), respectively. To prevent biased results towards the majority class, ten training sets were obtained by random sampling the same number of pixels from each of the six thematic classes (see Table II). In total, for each random split, $360,000$ pixels (1% of the ground-truth samples) were selected for training and the remaining $35,640,000$ pixels were used for testing. In the remainder of this paper, those random partitions into training/test set of Gray-Potsdam will be denoted as $GrayPotsdam_1$. Due to the dimensions of Gray-Potsdam, the random training pixels of $GrayPotsdam_1$ cannot be visualized when the image is downsized. The reader can refer to the source codes of our experiments [9] to visualize the train/test splits of the Gray-Potsdam dataset.

TABLE II: Number of training and test samples per class of the partitions $GrayPotsdam_1$ and $GrayPotsdam_2$ of the Gray-Potsdam dataset.

Class	Number of samples		
	Training set	Test set	Total
Background	60,000	1,565,250	1,625,250
Trees	60,000	5,785,203	5,845,203
Cars	60,000	631,810	691,810
Buildings	60,000	12,362,473	12,422,473
Low vegetation	60,000	8,639,455	8,699,455
Impervious surfaces	60,000	6,655,809	6,715,809

[8]The Potsdam dataset and its ground truth can be downloaded from http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html

[9]Source codes are available in https://gitlab.inria.fr/dsantana/attributes-profiles-survey-source-codes

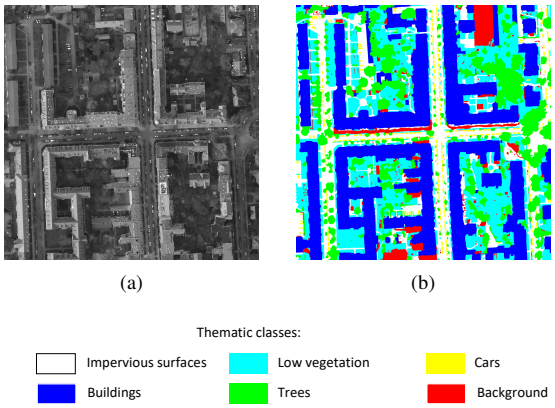


Fig. 4: Grayscale version of the image *top_potsdam_7_7.tiff* of the Potsdam dataset (a) and its thematic ground truth (b).

B. Experimental setup

The experiments have been conducted using a wide variety of parameters, shown in Table III. Some of which have been employed for the first time in the state of the art. To ease the readability of the results presented in this section and in the following section, we present a summary of our experimental settings in tables IV and V. Tab. IV shows the parameter settings used for each experiment, and Tab. V contains the description of each dataset split employed in our experiments.

The popular choices of classifiers for APs are SVM and Random Forest (RF) [14]. As suggested in [14], RFs usually perform better than SVMs in this context. Moreover, RFs require lower training and prediction time [80]. Hence, our supervised classification was conducted on the two datasets using the RF classifier [100] with 100 trees, as employed in other studies [19], [96]. In the scikit-learn implementation of random forests, a random shuffling of the data is applied before training. Hence, the classification output for each run is not reproducible unless a fixed value is assigned to the $random\_state$ parameter of the $RandomForestClassifier$ class. To take this into consideration, the classification results reported in this section and in the following section are the average and the standard deviation of the scores obtained for ten runs on each dataset. The number of variables involved in the training was set to the square root of the feature vector length. In order to evaluate and compare classification accuracy of different approaches, overall accuracy (OA), average accuracy (AA), and kappa coefficient (κ) have been taken into account [7]. As far as attribute filtering is concerned, for the sake of design simplicity and computational cost we have limited our experiments with two attributes, *area* and the *moment of inertia*. The former is increasing and the latter is not. For Pavia dataset, fourteen area thresholds were computed automatically using the formula proposed in [61]:

TABLE III: Choice of methods and parameters employed in the reported tests.

Datasets	Attributes	Tree Types	Quantization	Connectivity	Post-processing	Thresholds
PaviaU	Area	Max-Min	8 bits	4	HAPs	manual
Gray-Potsdam	Moment of inertia	Tree of shapes	16 bits	8	FPs	automatic
		α-tree	64 bits		LFAPs	without
		ω-tree				
		Max-tree				
		Min-tree				

TABLE IV: Experimental settings employed in the reported tests.

	Dataset split	Attributes	Tree types	Quantization	Connectivity	Post-Processing	Thresholds
Sec. IV Tab. VI	$Pavia_1$	All	All	8 bits	4	All	All
Sec. IV Tab. VIII	$GrayPotsdam_1$	All	All	8 bits	4	FPs	manual
						LFAPs	without
Sec. IV Tab. X	$GrayPotsdam_1$	All	All	8 bits	4	FPs	manual
					8	LFAPs	without
Sec. IV Tab. XI	$Pavia_1$	All	All	8 bits	4	All	All
					8		
Sec. IV Tab. XII	$Pavia_1$	All	All	8 bits	4	All	All
				16 bits			
				32 bits			
Sec. V Tab. XIII	$Pavia_1$	All	All	8 bits	4	FPs	All
	$Pavia_2$						
Sec. V Tab. XIV	$Pavia_2$	All	All	8 bits	4	FPs	All
Sec. V Tab. XV	$GrayPotsdam_1$	All	All	8 bits	4	FPs	manual
	$GrayPotsdam_2$						without
Sec. V Tab. XVI	$Potsdam_2$	All	All	8 bits	4	FPs	manual
							without
Sec. V Tab. XVII	$Potsdam_3$	All	All	8 bits	4	FPs	manual
Sec. V Tab. XVIII	$Potsdam_3$	All	All	8 bits	4	FPs	manual

TABLE V: Pavia and Gray-Potsdam dataset splits employed in the reported tests.

Dataset split	Description	Objective
$Pavia_1$ (Fig. 3)	⋆ Standard split obtained from http://dase.grss-ieee.org/. ⋆ Commonly used in the literature.	⋆ Provide evaluation scores of AP extensions which are comparable with the results in the literature.
$Pavia_2$ (Fig. 9(b))	⋆ Ten sets of training/test samples randomly extracted from a restricted region of the Pavia data. ⋆ Lower variability of training pixels when compared to $Pavia_1$. ⋆ Same number of training samples per class as $Pavia_1$.	⋆ Evaluate the impact of having large ground-truth regions not contributing to the training set. ⋆ Generalize APs to datasets with lower levels of 'leakage' of training features.
$GrayPotsdam_1$	⋆ Ten sets of training/test samples randomly extracted from all of the ground-truth connected components of Gray-Potsdam. ⋆ Same number of training samples per semantic class.	⋆ Evaluate APs and its extensions on Gray-Potsdam using the training/test splitting method commonly used in the literature.
$GrayPotsdam_2$ (Fig. 8)	⋆ Ten sets of training/test samples randomly extracted from a restricted set of ground-truth connected components of Gray-Potsdam. ⋆ Same number of traning and test samples as $GrayPotsdam_1$.	⋆ Similar to $Pavia_2$: introduce connected components in the test set that do not include any training pixels. Test if the features extracted from APs really reflect the geometrical characteristics of the objects belonging to a certain class, or if the success of APs are mainly due to the leakage of training/testing features.
$GrayPotsdam_3$ (Fig. 11)	⋆ Dataset divided in half. ⋆ Ten sets of training samples randomly extracted from the upper half. ⋆ Test set composed of all pixels from the lower half. ⋆ Same number of training pixels per class as $GrayPotsdam_1$ and $GrayPotsdam_2$, but different number of test pixels.	⋆ Generalize APs to multiple-image datasets. ⋆ Compute training and testing features from different trees obtained from the two halves of the image.

$$\lambda_a, X = \frac{1000}{v}\{a_{min}, a_{min}+\delta_a, a_{min}+2\delta_a, \ldots, a_{max}\} \quad (3)$$

that has received wide acclaim in related works [14], [27], [96], [101]. In Eq. (3) a_{min} and a_{max} are initialized by 1 and 14 respectively, with a step increase δ_a equal to 1 and v represents the spatial resolution of the input data. The resulting thresholds for the Pavia dataset follow:

$$\lambda_{a,Pav} = \{770, 1538, 2307, 3076, 3846, 4615, 5384, 6153, 6923, 7692, 8461, 9230, 10000, 10769\}.$$

For the Gray-Potsdam dataset, the thresholds obtained with the previous formula were not sufficient to cover the size variations of the targeted classes. To obtain a set of thresholds spanning a larger range of values without increasing the number of thresholds, we considered the fourteenth first values of the geometric sequence whose n-th term is given by 200×2^n. The resulting thresholds for the Gray-Potsdam follow:

$$\lambda_{a,Pot} = \{200, 400, 800, 1600, 3200, 6400, 12800, 25600, 51200, 102400, 204800, 409600, 819200, 1638400\}.$$

As far as moment of inertia is concerned, the manually set thresholds used in several studies [19], [25], [27] were adopted here as well:

$$\lambda_{i,Pot} = \lambda_{i,Pav} = \{0.2, 0.3, 0.4, 0.5\}.$$

Moreover, as mentioned in Tab. III, we consider two additional experimentation parameters, namely the input image's connectivity and quantization level, the effect of which, to the best of our knowledge, has not been previously studied in the state of the art. For instance in [49], [91], [102], the authors use 4-connectivity, but in most published studies these two parameters are seldom mentioned.

For the sake of simplicity and to avoid introducing new acronyms, EAP will be referred to as AP in the remainder of this article. Hence, whenever we discuss results on the Pavia dataset, it should be understood that the APs and its variants corresponds to EAPs on this dataset.

In more detail, we compare APs and EAPs generated from different kinds of trees including: max-tree (AP-maxT), min-tree (AP-minT), a max-tree along with a min-tree (as in standard APs) [7], the SDAPs [53] from tree of shapes, the α-APs and ω-APs from α- and ω-trees respectively [56]. We also provide the results of some effective post-processing techniques including the HAPs/HSDAPs [79], [80], LFAPs/LFSDAPs [80] and of some extensions of APs, including Threshold-Free (TF) APs [74] and FPs [96]. A far more limited comparative study of these parameters is reported in [97]. We obtained HAPs and HSDAPs using histograms of 7 bins and a window size of 7×7 pixels, which are the optimal parameter settings according to the experiments of [96]. LFAPs and LFSDAPs were computed by using the mean and the standard deviation of 7×7 sized windows as the local features. As shown in [96], feature and histogram profiles are fairly robust to the choice of window size, but the experiments of [96] with 7×7 windows provided superior results. FPs were obtained by projecting the average gray levels (FP_μ), the area (FP_a) and both average gray levels and area ($\text{FP}_{\mu+a}$) of tree nodes during the reconstruction step, as done in [96]. Among all tested methods, TF AP [74] is the only one not implemented in the SAP library.

C. Results

This subsection will start with the presentation and discussion of results obtained from experiments employing the commonly encountered 8-bit quantization and 4-connectivity [49], [91], [102] in the state of the art. Then, it will continue with an evaluation of the effect of the aforementioned two parameters on classification performance.

1) Classification results for Pavia dataset: The overall classification results, the classification results per class and the classification maps for the Pavia dataset obtained using the partition $Pavia_1$ are presented in Tab. VI, Tab. VII and Fig. 5, respectively.

For this dataset, the tree type underlying the APs appears to have an important influence on performance. More specifically, the α-APs and ω-APs outperformed both APs and SDAPs. In particular, by using the ω-tree, one can achieve an average OA of 96.33%, i.e. 5.76% and 2.23% better than standard APs and SDAPs, respectively. However, this improvement is not observed for all classes. While the accuracy of ω-AP for the gravel class increases by 20.58% with respect to APs, the accuracy for the asphalt class decreases by 5.98%. Then, by post-processing the AP and the SDAP with histogram and local feature profiles, we improved AP (resp. SDAP) by more than 4% (resp. 1%) in terms of overall accuracy. Regarding the feature profiles, FP_μ and $\text{FP}_{\mu+a}$ provided much better results than FP_a, as already shown in the original paper on FPs [96]. Finally, the threshold-free profile, which has less dimensions than all other tested profiles, outperformed AP by 3.41%, 3.13% and 4.51% in terms of OA, AA and κ, respectively. Among all tested methods, the best classification accuracy was achieved by F_μ, with $\text{OA} = 96.76$ and $\kappa \times 100 = 95.65$. Compared to the standard APs, an enhancement of 6.19% in OA and 8.12% in κ was adopted.

Furthermore, our classification results on the Pavia dataset are competitive with respect to some recent deep learning approaches discussed in [17]. In [17], the authors compare several CNN architectures from the literature for hyperspectral image classification. Among the tested architectures, the best results for Pavia ($OA = 84.32 \pm 0.72$ and $\kappa = 0.799 \pm 0.009$), obtained with the same dataset split ($Pavia_1$) employed here, achieved with a 3D CNN [103], are still inferior to most of our results given in Tab. VI.

The reader may note that our classification results differ from the ones presented in the original papers of (extended) APs [19], SDAPs [25], LFAPS [80], $\alpha-$ and ω-APs [56]. More precisely, the differences can be explained by the various sets of attributes and thresholds employed in some of the papers [19], [25], [80], by the use of pre- and post-processing techniques other than PCA [25], and by number of RF trees [25]. Moreover, even using the same set of thresholds and the

TABLE VI: Classification result of Pavia dataset obtained by different methods using the default 4-connectivity and 1-byte quantization.

Method	**Dimension**	**Classification result**		
		OA (%)	AA (%)	$\kappa \times 100$
4 PC	4	65.27 ± 0.25	74.88 ± 0.20	56.93 ± 0.27
AP-maxT	80	89.21 ± 0.63	87.37 ± 0.19	85.42 ± 0.80
AP-minT	80	87.11 ± 2.00	92.62 ± 0.61	83.23 ± 2.41
AP	152	90.57 ± 2.60	93.13 ± 0.67	87.53 ± 3.32
SDAP	80	94.10 ± 0.21	93.85 ± 0.37	92.16 ± 0.27
α-AP	80	95.46 ± 0.59	95.25 ± 1.23	93.91 ± 0.79
ω-AP	80	96.33 ± 0.40	**97.17 ± 0.89**	95.08 ± 0.54
HAP	1064	94.73 ± 0.30	92.96 ± 0.31	92.84 ± 0.42
HSDAP	340	95.35 ± 0.29	94.00 ± 0.49	93.79 ± 0.39
LFAP	304	94.75 ± 0.29	94.12 ± 0.41	92.97 ± 0.39
LFSDAP	160	96.34 ± 0.16	92.81 ± 0.20	95.06 ± 0.22
FP_μ	152	**96.76 ± 0.15**	97.05 ± 0.19	**95.65 ± 0.21**
FP_a	152	85.48 ± 0.98	93.94 ± 0.55	81.36 ± 1.16
$FP_{\mu+a}$	304	96.15 ± 0.14	96.74 ± 0.32	94.82 ± 0.19
TF-AP	72	93.98 ± 0.53	96.26 ± 0.29	92.04 ± 0.68

TABLE VII: Classification results per class of Pavia dataset obtained by different methods using the default 4-connectivity and 1-byte quantization.

Method	**Dimension**	**Accuracy per class (%)**								
		Asphalt	Meadow	Gravel	Tree	Metal	Soil	Bitumen	Brick	Shadow
4 PC	4	71.25 ± 0.44	53.20 ± 0.45	38.39 ± 0.53	98.28 ± 0.10	98.79 ± 0.17	67.34 ± 0.58	66.74 ± 0.96	83.28 ± 0.44	96.64 ± 0.34
AP-maxT	80	92.33 ± 0.14	93.66 ± 1.41	43.32 ± 0.67	95.61 ± 0.77	99.63 ± 0.08	71.07 ± 0.24	97.83 ± 0.14	95.28 ± 0.13	97.61 ± 0.46
AP-minT	80	92.81 ± 0.08	80.28 ± 4.42	86.02 ± 1.65	98.97 ± 0.08	99.90 ± 0.08	84.19 ± 0.08	99.93 ± 0.12	98.02 ± 0.13	93.47 ± 3.01
AP	152	95.71 ± 0.21	87.38 ± 5.81	73.17 ± 3.00	99.09 ± 0.29	99.65 ± 0.05	85.64 ± 0.14	**100.0 ± 0.00**	99.24 ± 0.18	98.25 ± 2.03
SDAP	80	97.15 ± 0.28	92.51 ± 0.47	77.15 ± 0.29	93.21 ± 0.70	99.83 ± 0.06	99.08 ± 0.00	98.94 ± 0.25	98.15 ± 0.50	88.62 ± 3.07
α-AP	80	89.39 ± 0.21	96.63 ± 0.43	76.27 ± 11.05	99.73 ± 0.07	99.61 ± 0.07	98.93 ± 0.06	99.49 ± 0.00	99.35 ± 0.10	97.82 ± 0.44
ω-AP	80	89.73 ± 0.56	96.70 ± 0.42	**93.75 ± 7.84**	**99.75 ± 0.09**	99.64 ± 0.04	98.95 ± 0.06	99.49 ± 0.00	99.36 ± 0.11	97.12 ± 0.40
HAP	1064	**99.86 ± 0.16**	97.68 ± 0.42	63.83 ± 1.71	97.74 ± 0.11	99.96 ± 0.06	80.73 ± 2.22	99.99 ± 0.03	98.13 ± 0.07	98.73 ± 0.33
HSDAP	340	99.70 ± 0.18	94.49 ± 0.46	68.59 ± 2.44	95.25 ± 0.19	**99.99 ± 0.03**	**99.68 ± 0.39**	99.55 ± 0.10	99.08 ± 0.06	89.70 ± 2.48
LFAP	304	90.33 ± 0.90	96.23 ± 0.36	72.68 ± 3.79	96.79 ± 0.10	99.20 ± 0.24	98.08 ± 0.09	99.36 ± 0.17	97.28 ± 0.10	97.12 ± 0.79
LFSDAP	160	99.42 ± 0.18	**98.42 ± 0.12**	65.82 ± 0.51	92.88 ± 0.80	99.52 ± 0.18	97.06 ± 0.56	99.16 ± 0.14	98.75 ± 0.13	84.26 ± 1.25
FP_μ	152	94.97 ± 0.13	97.10 ± 0.21	88.65 ± 0.91	97.07 ± 0.19	99.89 ± 0.04	98.43 ± 1.57	**100.0 ± 0.00**	97.43 ± 0.53	**99.92 ± 0.06**
FP_a	152	95.51 ± 0.25	73.87 ± 2.20	92.31 ± 4.14	99.31 ± 0.20	99.93 ± 0.04	86.98 ± 1.18	99.98 ± 0.04	**99.72 ± 0.10**	97.80 ± 1.82
$FP_{\mu+a}$	304	96.03 ± 0.23	96.61 ± 0.21	92.43 ± 1.28	97.93 ± 0.43	99.89 ± 0.04	91.00 ± 0.96	99.97 ± 0.05	98.57 ± 0.38	98.23 ± 1.78
TF-AP	72	95.79 ± 0.26	90.31 ± 1.14	82.42 ± 2.41	99.64 ± 0.08	99.75 ± 0.10	99.23 ± 0.19	**100.0 ± 0.00**	99.61 ± 0.06	99.56 ± 0.06

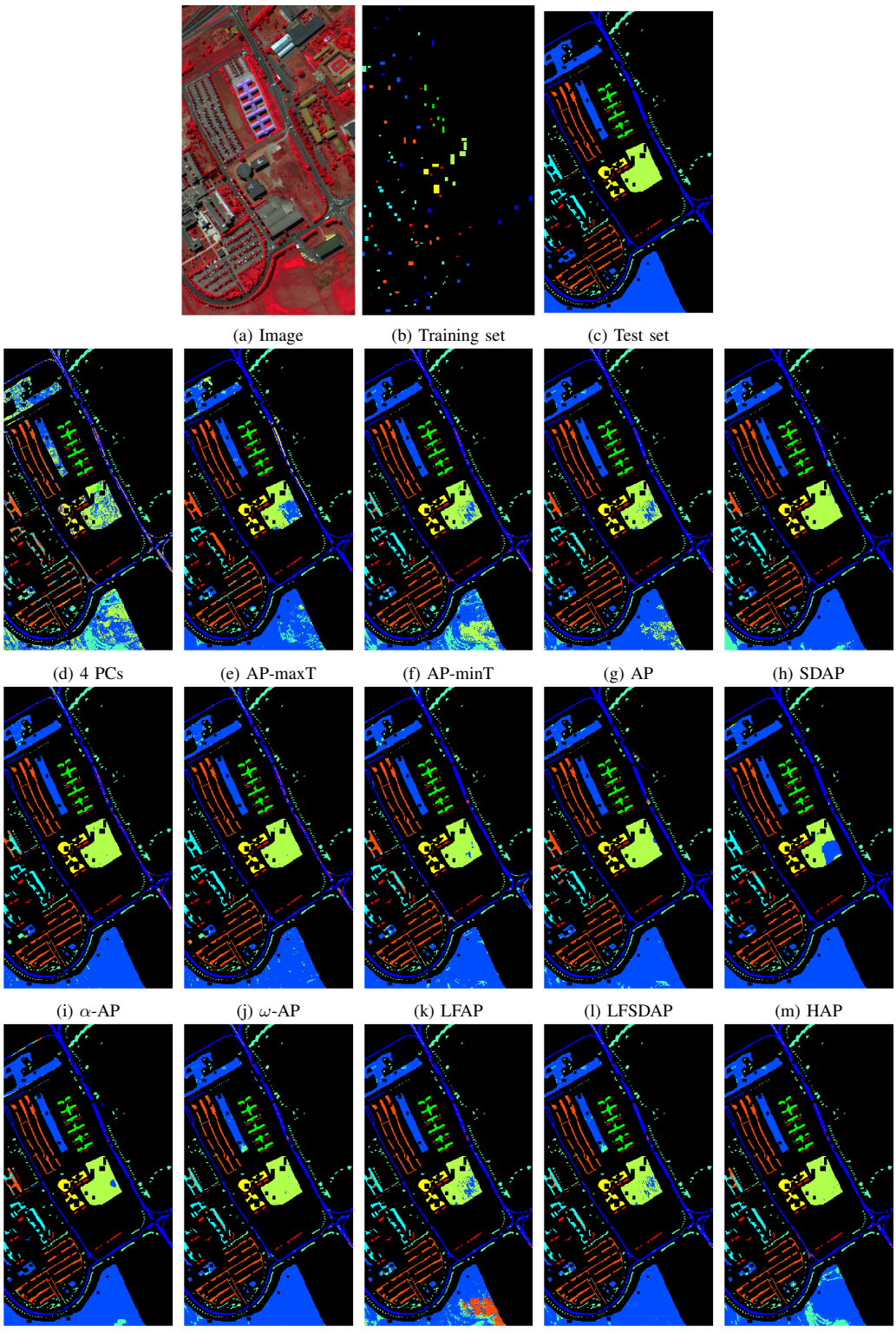
same number of RF trees, as done in [56], the quantization and connectivity parameters, which are not explicitly given in those papers, may play a role in the final results, as discussed later in Sec. IV-D. Nevertheless, in term of conclusion, there is no incoherence between our paper and the papers cited in the beginning of this paragraph.

2) Classification results for Gray-Potsdam dataset: The overall and per-class classification results for the Gray-Potsdam dataset using $GrayPotsdam_1$ are presented in Tables VIII and IX, respectively. As already mentioned previously, the reported results are the average scores over ten runs on the different random training-test splits of $GrayPotsdam_1$. Since LFAPs perform better than HAPs in general, as attested by [80], and due to the expensive computation of HAPs, we consider only LFAPs in our experiments with the Gray-Potsdam dataset.

In the case of the Gray-Potsdam dataset, it can be observed that AP variants boost accuracy consistently at various degrees. In particular, the α-APs and ω-APs could outperform APs on each single max-tree or min-tree but still falls below the standard APs. On the other hand, SDAP performed better than AP, α-AP and ω-AP. Then, by post-processing the output profiles, LFAP and LFSDAP outperformed AP and SDAP by 3.52% and 4.61%, respectively, in terms of overall accuracy. Among the feature profiles, the best result was achieved by $FP_{\mu+a}$, which outperforms the APs by more than 2% in terms of OA, AA and κ. Finally, the threshold-free AP presented lower scores than APs, but it outperformed AP-maxT, AP-minT, α-AP and ω-AP despite having the smallest number of dimensions among all methods. In conclusion, the best classification result was obtained by LFSDAP with 80.80% of overall accuracy, which represents an improvement of 5.70% with respect to the standard AP.

Fig. 6(a) and (b) present a crop of the Gray-Potsdam dataset and of its ground-truth, respectively, which are composed of the first 500 lines (from the top to the bottom) and 1200 columns (from left to right) of the original data. Fig. 6(c)-(o) illustrates the classification maps obtained on the crop of Fig. 6(a) using the aforementioned methods. We can see that the classification based solely on pixels gray value is very noisy in most regions of the image. By incorporating spatial information from attribute profiles, we see a more structured result, with a clearer separation between the regions of different classes. Furthermore, the post-processing of APs and SDAPs with local features successfully reduces the noise in all classes, especially the regions containing trees (in green) and buildings (in blue).

In terms of future research directions, it would be interesting to investigate the combination potential of the AP variants and extensions, in an effort to discover whether they provide com-



(a) Image (b) Training set (c) Test set

(d) 4 PCs (e) AP-maxT (f) AP-minT (g) AP (h) SDAP

(i) α-AP (j) ω-AP (k) LFAP (l) LFSDAP (m) HAP

(n) HSDAP (o) FP_{μ} (p) FP_{a} (q) $FP_{\mu+a}$ (r) TF-AP

Fig. 5: Classification results of Pavia data corresponding to the results of table VI. : trees, : gravel, : meadows, : asphalt, : metal, : bare soil, : bitumen, : shadows, : bricks.

TABLE VIII: Classification results of Gray-Potsdam dataset obtained by different methods using the default 4-connectivity and 1-byte quantization.

Method	Dimension	Classification result		
		OA (%)	AA (%)	$\kappa \times 100$
Gray-values	1	46.56 ± 0.42	34.30 ± 0.02	30.84 ± 0.32
AP-maxT	20	63.65 ± 0.14	59.68 ± 0.02	52.65 ± 0.15
AP-minT	20	58.00 ± 0.23	56.55 ± 0.02	47.13 ± 0.21
AP	38	75.10 ± 0.05	77.56 ± 0.02	67.83 ± 0.05
SDAP	20	76.19 ± 0.08	78.25 ± 0.02	69.20 ± 0.09
α-AP	20	68.32 ± 0.07	67.94 ± 0.04	59.47 ± 0.08
ω-AP	20	68.10 ± 0.06	67.78 ± 0.03	59.23 ± 0.06
LFAP	76	78.62 ± 0.04	81.21 ± 0.04	72.25 ± 0.05
LFSDAP	40	**80.80 ± 0.03**	**83.44 ± 0.02**	**75.05 ± 0.04**
FP_μ	38	77.14 ± 0.05	79.41 ± 0.02	70.42 ± 0.06
FP_a	38	77.19 ± 0.05	79.46 ± 0.02	70.47 ± 0.05
$FP_{\mu+a}$	76	77.94 ± 0.04	80.17 ± 0.03	71.43 ± 0.05
TF-AP	18	72.34 ± 0.05	74.93 ± 0.04	64.33 ± 0.06

TABLE IX: Classification results per class of Gray-Potsdam dataset obtained by different methods using the default 4-connectivity and 1-byte quantization.

Method	Dimension	Accuracy per class					
		Background	Trees	Cars	Buildings	Low vegetation	Impervious surfaces
Gray-values	1	10.24 ± 1.34	16.23 ± 1.23	11.45 ± 0.22	49.82 ± 1.51	70.86 ± 1.51	47.21 ± 1.12
AP-maxT	20	53.82 ± 0.19	31.80 ± 0.63	59.40 ± 0.54	68.53 ± 0.29	77.42 ± 0.58	67.08 ± 0.62
AP-minT	20	54.74 ± 0.80	51.14 ± 1.22	58.37 ± 0.19	61.34 ± 0.83	67.38 ± 0.30	46.32 ± 1.28
AP	38	85.43 ± 0.21	63.49 ± 0.17	92.02 ± 0.08	84.15 ± 0.16	72.42 ± 0.21	67.85 ± 0.22
SDAP	20	82.57 ± 0.48	64.89 ± 0.22	92.82 ± 0.09	83.93 ± 0.14	74.43 ± 0.23	70.87 ± 0.48
α-AP	20	71.70 ± 0.14	50.39 ± 0.30	75.38 ± 0.21	80.68 ± 0.18	62.65 ± 0.13	66.86 ± 0.24
ω-AP	20	71.07 ± 0.14	50.26 ± 0.42	75.73 ± 0.24	80.24 ± 0.21	62.62 ± 0.17	66.76 ± 0.22
LFAP	76	85.18 ± 0.11	70.54 ± 0.17	96.00 ± 0.05	84.38 ± 0.09	75.87 ± 0.09	75.30 ± 0.07
LFSDAP	40	**87.98 ± 0.09**	**74.25 ± 0.12**	**97.30 ± 0.03**	86.15 ± 0.10	**78.00 ± 0.08**	**76.94 ± 0.08**
FP_μ	38	86.55 ± 0.22	66.47 ± 0.17	93.06 ± 0.05	85.73 ± 0.15	74.66 ± 0.17	69.97 ± 0.18
FP_a	38	86.96 ± 0.22	66.42 ± 0.18	92.98 ± 0.06	85.90 ± 0.13	74.70 ± 0.19	69.81 ± 0.21
$FP_{\mu+a}$	76	87.39 ± 0.16	67.60 ± 0.14	93.40 ± 0.05	**86.41 ± 0.13**	75.47 ± 0.19	70.74 ± 0.18
TF-AP	18	82.76 ± 0.18	59.09 ± 0.16	90.47 ± 0.10	81.70 ± 0.14	69.24 ± 0.14	66.32 ± 0.24

plementary information and eventually higher performances.

D. Assessment of connectivity and quantization

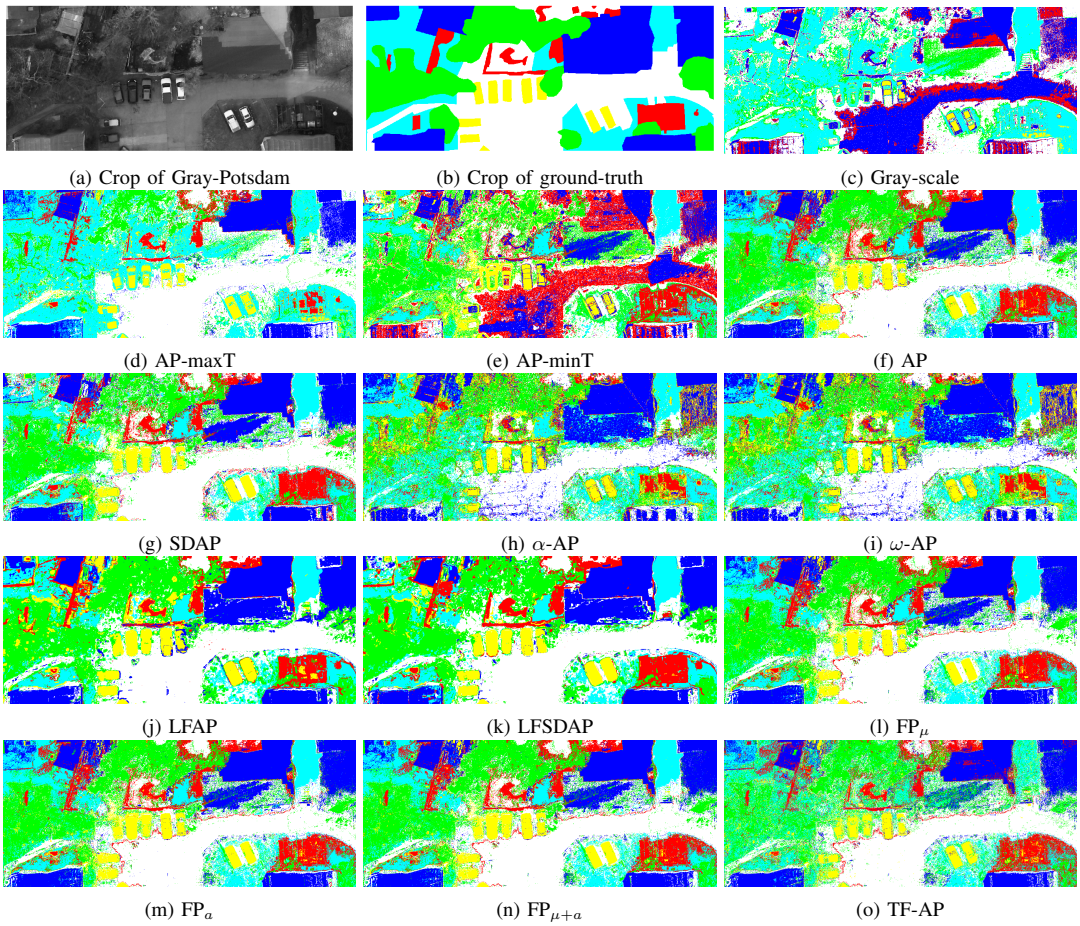
In this section, we discuss the influence of the connectivity parameters (4 versus 8) and of the quantization parameters (64 bits, 16 bits and 8 bits) in the classification results obtained with different methods.

As previously stated, the connectivity parameter is rarely mentioned in published works, though it can have a non-negligible impact in the construction of tree representations. In general, trees computed with 4-connectivity are "finer" than the ones obtained with 8-connectivity. In other words, given the trees T_4 and T_8 obtained from the same image using 4- and 8-connectivity, respectively, every node of T_4 is a subset of a node of T_8. For instance, in Fig. 7(b) and (c), we show the max-trees of the image of Fig. 7(a) computed with 4- and 8-connectivity, respectively. It can be verified that every node of Max-T_4 is a subset of a node of Max-T_8. As another example, the max-tree of the 4-connected Gray-Potsdam image is composed of approximately 31% more nodes than the max-tree of the 8-connected image ($4,725,207$ *vs* $3,606,550$ nodes). Concerning the max-trees computed on the first principal component of Pavia, the relative difference in the number of nodes is even higher: using 4-connectivity lead to approximately 43.7% more nodes than using 8-connectivity ($25,030$ *vs* $17,413$ nodes). Those observations raise the question of whether the connectivity has as much of an impact on the APs as it has on the number of tree nodes.

Similarly to the connectivity, the quantization parameter affects heavily the trees' depth and number of nodes. For the Pavia dataset [10], the value of a pixel at every band is represented as a 16-bit unsigned integer. However, as aforementioned, the APs are computed on the four principal components of the Pavia dataset. Those components are obtained with the *PCA.fit()* method of the scikit-learn Python library, which returns real-valued (64-bit float) components. In general, rounding those real values to 16 or 8-bit integers reduce the number of distinct values in the components and, consequently, the time and space complexity to compute their respective trees. Though the computation time and space complexity is not critical for the small sized Pavia dataset, it is of great importance for larger datasets. Hence, we will study the effect of approximating the 64-bit values to 16-bit and to 8-bit values in the classification of Pavia.

Since the post-processing techniques LFAP and HAP do not depend directly on the connectivity and quantization parameters, they will not be included in this set of experiments. Moreover, as the tree-of-shapes is not yet implemented with 8-connectivity in the Higra package, it will not be considered in this section.

[10]downloaded from http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes



(a) Crop of Gray-Potsdam (b) Crop of ground-truth (c) Gray-scale

(d) AP-maxT (e) AP-minT (f) AP

(g) SDAP (h) α-AP (i) ω-AP

(j) LFAP (k) LFSDAP (l) FP_{μ}

(m) FP_{a} (n) $\text{FP}_{\mu+a}$ (o) TF-AP

Fig. 6: Classification results of a crop of Gray-Potsdam corresponding to the results of Table VIII. : impervious surfaces, : buildings, : low vegetation, : trees, : cars, : background.

In Tables X and XI, we present the classification results of Gray-Potsdam and Pavia data, respectively, with 4- and 8-connectivity.

For the Potsdam data, the connectivity parameter had little but consistent influence on the classification results with different methods. The experiments with 4-connectivity provided better scores in general. We attribute this result to the larger number of tree nodes obtained using 4- instead of 8-connectivity. Hence, we conclude that the finer regions obtained with 4-connectivity provide valuable attributes for performing classification on this dataset.

Regarding the Pavia data, changing the connectivity parameter had a different impact on each method. All methods perform better on the 4-connected Pavia data, except for FP_a. Notably, changing the connectivity parameter of the AP-minT from 8 to 4 lead to an improvement of 8.89% and 11.02% in terms of OA and κ, respectively. On the other hand, changing the connectivity parameter of the FP_a from 4 to 8 lead to an improvement of 4.09% and 4.72% in terms of OA and κ, respectively.

Overall though, 4-connectivity can be observed to outperform almost consistently 8-connectivity. In terms of future research directions, advanced connectivity concepts such as hyper-connectivity [104] and mask based connectivity [105] appear as promising options.

We now focus on the assessment of the quantization parameter in the classification of the Pavia dataset. Since the RGB values of the Potsdam dataset are already provided as 8-bit values, the quantization parameter will not be assessed on Gray-Potsdam.

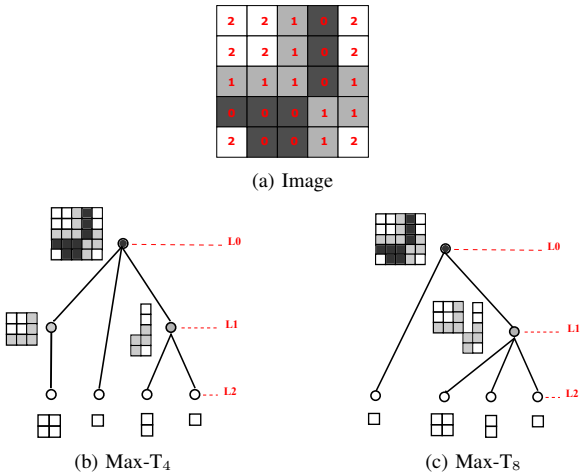
In table XII, we present the classification results of the Pavia dataset with 8-bit (default), 16-bit and 64-bit quantization. Regarding the classification based solely on the four principal components, approximating the 64-bit float values to 16-bit values had little influence on the results. Whereas, approximating the 64-bit/16-bit to 8-bit values had a larger impact on the classification results, with a decrease of more than 3% in terms of accuracy and κ scores. A more significant loss is observed in the classification results with AP-minT and AP, with a decrease of more than 5% in terms of accuracy and

TABLE X: Comparison between the classification results of Gray-Potsdam for different connectivity parameters.

Method	Dimension	Classification result (4-connectivity)			Classification result (8-connectivity)		
		OA (%)	AA (%)	$\kappa \times 100$	OA (%)	AA (%)	$\kappa \times 100$
AP-maxT	20	**63.65 ± 0.14**	**59.68 ± 0.02**	**52.65 ± 0.15**	63.03 ± 0.16	58.90 ± 0.02	51.84 ± 0.17
AP-minT	20	**58.00 ± 0.23**	**56.55 ± 0.02**	**47.13 ± 0.21**	57.36 ± 0.23	55.75 ± 0.03	46.30 ± 0.21
AP	38	**75.10 ± 0.05**	**77.56 ± 0.02**	**67.83 ± 0.05**	73.91 ± 0.07	76.42 ± 0.03	66.36 ± 0.07
α-AP	20	**68.32 ± 0.07**	**67.94 ± 0.04**	**59.47 ± 0.08**	67.67 ± 0.08	66.72 ± 0.04	58.64 ± 0.08
ω-AP	20	**68.10 ± 0.06**	**67.78 ± 0.03**	**59.23 ± 0.06**	67.40 ± 0.08	66.46 ± 0.03	58.34 ± 0.08
FP_μ	38	**77.14 ± 0.05**	**79.41 ± 0.02**	**70.42 ± 0.06**	75.95 ± 0.07	78.23 ± 0.02	68.91 ± 0.08
FP_a	38	**77.19 ± 0.05**	**79.46 ± 0.02**	**70.47 ± 0.05**	75.98 ± 0.09	78.28 ± 0.03	68.96 ± 0.10
$FP_{\mu+a}$	76	**77.94 ± 0.04**	**80.17 ± 0.03**	**71.43 ± 0.05**	76.70 ± 0.06	78.96 ± 0.03	69.86 ± 0.07
TF-AP	18	**72.34 ± 0.05**	**74.93 ± 0.04**	**64.33 ± 0.06**	71.90 ± 0.08	74.49 ± 0.03	63.81 ± 0.09

TABLE XI: Comparison between the classification results of Pavia for different connectivity parameters.

Method	Dimension	Classification result (4-connectivity)			Classification result (8-connectivity)		
		OA (%)	AA (%)	$\kappa \times 100$	OA (%)	AA (%)	$\kappa \times 100$
AP-maxT	80	**89.21 ± 0.63**	87.37 ± 0.19	**85.42 ± 0.80**	87.88 ± 1.16	**89.56 ± 0.30**	84.17 ± 1.43
AP-minT	80	**87.11 ± 2.00**	**92.62 ± 0.61**	**83.23 ± 2.41**	78.22 ± 0.50	88.26 ± 0.14	72.21 ± 0.57
AP	152	**90.57 ± 2.60**	93.13 ± 0.67	**87.53 ± 3.32**	90.54 ± 1.55	**93.80 ± 0.40**	87.31 ± 1.99
α-AP	80	**95.46 ± 0.59**	95.25 ± 1.23	**93.91 ± 0.79**	94.73 ± 0.40	**95.66 ± 0.27**	92.97 ± 0.52
ω-AP	80	**96.33 ± 0.40**	**97.17 ± 0.89**	**95.08 ± 0.54**	94.71 ± 0.34	95.69 ± 0.17	92.94 ± 0.44
FP_μ	152	**96.76 ± 0.15**	**97.05 ± 0.19**	**95.65 ± 0.21**	93.35 ± 0.15	93.55 ± 0.16	90.90 ± 0.20
FP_a	152	85.48 ± 0.98	93.94 ± 0.55	81.36 ± 1.16	**89.57 ± 1.18**	**94.03 ± 0.32**	**86.08 ± 1.50**
$FP_{\mu+a}$	304	**96.15 ± 0.14**	**96.74 ± 0.32**	**94.82 ± 0.19**	94.21 ± 0.32	95.00 ± 0.28	92.09 ± 0.44
TF-AP	72	**93.98 ± 0.53**	**96.26 ± 0.29**	**92.04 ± 0.68**	92.65 ± 0.24	95.26 ± 0.17	90.32 ± 0.31



(a) Image

(b) Max-T_4 (c) Max-T_8

Fig. 7: (a) Original grayscale image $X : E \rightarrow [0, 1, 2]$. (b) Max-tree of X computed using 4-connectivity. (c) Max-tree of X computed using 8-connectivity.

κ scores when the 64-bit float values are approximated to 8-bit values. On the other hand, we have opposite results for SDAP, α-AP and ω-AP: using 8-bit for quantization provides the best scores. Consequently, we could draw different conclusions regarding the performance of APs compared to SDAPs, α-AP and ω-AP depending on the quantization settings: if 8-bit, then SDAP, α-AP and ω-AP outperform AP; otherwise, AP outperform those three methods. Regarding feature profiles, the best FP_μ and $FP_{\mu+a}$ results are achieved with 64-bit quantization. Whereas, the best FP_a is achieved with 8-bit quantization. It is noteworthy that, as discussed previously, FP_μ and FP_a also presented opposite results with respect to the connectivity parameters. This may suggest that having finer (resp. coarser) components trees, obtained with 4-connectivity and 64-bit quantization (resp. 8-connectivity and 8-bit quantization) for example, lead to better FP_μ (resp. FP_a) scores.

Our experiments show that besides the choice of attributes and threshold values, the connectivity and quantization parameters have a great impact as well on the performance of APs with different tree representations. More importantly, this effect varies greatly depending on the underlying tree type. In summary, these two parameters that are almost always silently set to (unmentioned in published studies) default values, appear to merit the same level of attention and care that threshold/attribute selection enjoys.

V. Discussion on the Generalization of APs

The experiments described in the previous section follow the same approach of other experiments in the literature: the training and testing features are obtained from the same tree computed on the whole data (or on the principal components of the data). This approach is reasonable when the aim is to completely classify the pixels of an image whose annotated pixels are evenly spread across this image [12]. That was the case of the training sets considered previously, which allowed us to obtain an improvement of more than 30% in terms of classification accuracy using APs with respect to spectral pixel values. However, this technique raises doubts in case of situations where one encounters distinct images for training and testing, or alternatively when the training pixels are not evenly spread across the data. This issue has already been discussed in [17], where the authors show that having training samples evenly spread across all ground-truth connected components is not a realistic scenario for evaluating classification methods in the remote sensing context.

In this section, we discuss the generalization of APs in those two scenarios. We first address in Sec. V-A the problem of

TABLE XII: Comparison between the classification results of Pavia for different quantization parameters.

Method	Dimension	Classification result (8-bit)			Classification result (16-bit)			Classification result (64-bit)		
		OA (%)	AA (%)	$\kappa \times 100$	OA (%)	AA (%)	$\kappa \times 100$	OA (%)	AA (%)	$\kappa \times 100$
4 PC	4	65.27 ± 0.25	74.88 ± 0.20	56.93 ± 0.27	**68.60 ± 0.10**	**78.08 ± 0.09**	**60.93 ± 0.11**	68.51 ± 0.24	**78.08 ± 0.22**	60.81 ± 0.29
AP-maxT	80	**89.21 ± 0.63**	87.37 ± 0.19	**85.42 ± 0.80**	88.97 ± 0.20	87.39 ± 0.15	84.99 ± 0.27	89.07 ± 0.37	**87.42 ± 0.17**	85.13 ± 0.49
AP-minT	80	87.11 ± 2.00	92.62 ± 0.61	83.23 ± 2.41	91.58 ± 1.64	94.55 ± 0.65	88.89 ± 2.06	**92.42 ± 1.07**	**94.77 ± 0.83**	**89.97 ± 1.35**
AP	152	90.57 ± 2.60	93.13 ± 0.67	87.53 ± 3.32	95.43 ± 0.21	94.01 ± 0.47	93.83 ± 0.28	**95.50 ± 0.19**	**94.25 ± 0.52**	**93.93 ± 0.26**
SDAP	80	**94.10 ± 0.21**	**93.85 ± 0.37**	**92.16 ± 0.27**	92.29 ± 0.32	91.36 ± 0.42	89.66 ± 0.42	92.49 ± 0.47	91.34 ± 0.61	89.92 ± 0.61
α-AP	80	**95.46 ± 0.59**	**95.25 ± 1.23**	**93.91 ± 0.79**	93.87 ± 0.72	93.85 ± 0.22	91.82 ± 0.93	94.08 ± 0.17	93.96 ± 0.09	92.10 ± 0.22
ω-AP	80	**96.33 ± 0.40**	**97.17 ± 0.89**	**95.08 ± 0.54**	93.75 ± 0.54	92.97 ± 0.76	91.64 ± 0.71	93.75 ± 0.44	92.78 ± 0.98	91.64 ± 0.59
FP_{μ}	152	96.76 ± 0.15	97.05 ± 0.19	95.65 ± 0.21	97.09 ± 0.70	97.43 ± 0.47	96.10 ± 0.93	**97.10 ± 1.00**	**97.64 ± 0.33**	**96.11 ± 1.31**
FP_{a}	152	**85.48 ± 0.98**	**93.94 ± 0.55**	**81.36 ± 1.16**	82.72 ± 0.74	92.64 ± 0.63	77.99 ± 0.95	83.09 ± 0.42	93.05 ± 0.29	78.50 ± 0.55
$FP_{\mu+a}$	304	96.15 ± 0.14	96.74 ± 0.32	94.82 ± 0.19	97.48 ± 0.31	**98.30 ± 0.19**	96.62 ± 0.41	**97.55 ± 0.29**	98.23 ± 0.26	**96.71 ± 0.39**

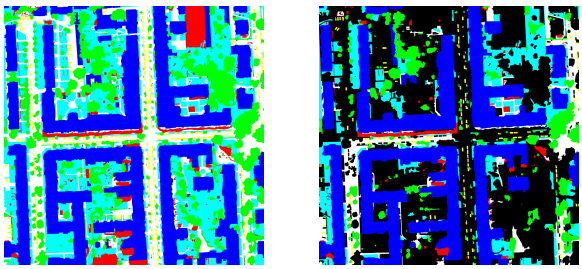
having data composed of a single image, but with a better separation between training and testing pixels. To do so, we perform experiments using a new split of the Gray-Potsdam and Pavia datasets. Then, in Sec. V-B, we approach the case where training and testing sets belong to different images. We split the Gray-Potsdam image so that APs can be computed separately for training and testing pixels.

A. Generalization of APs to other partially annotated images

The "standard" training set of the Pavia University dataset, *i.e.* the training set of $Pavia_1$, is composed of pixels belonging to most of the ground-truth (and testing) connected components. Hence, the training set of each semantic class accounts for most of the variability in terms of spectral signatures and geometric properties of the connected components of the said class. However, the availability of training samples from every connected component of each semantic class in a real-world scenario, dealing commonly with remote sensing datasets representing geographically large areas, is evidently unrealistic. The data splitting procedure used in this section aims to simulate, at a limited degree, the aforementioned real-world conditions.

We compare the classification results presented in the previous section with the results based on a new partition of the Pavia and Gray-Potsdam datasets[11]. In the new partition of Pavia (see Fig. 9(b) and (c)), ten sets of training samples were randomly extracted from a restricted region (composed of 85 connected components) of the ground-truth, which is composed of 229 connected components in total. In order to provide comparable results with the standard partition of this dataset, we selected the same number of training samples per class, resulting in 3921 training samples (see Table I) and $40,002$ testing samples for each of the ten random splits. Similarly, ten sets of training samples of Gray-Potsdam were extracted from nearly half of the ground-truth connected components. In total, 726 ground-truth connected components (see Fig. 8) contributed to the new training sets of Gray-Potsdam. The same number of training and test samples given in Table II were obtained. These new partitions of the Pavia and Gray-Potsdam datasets will be denoted as $Pavia_2$ and $GrayPotsdam_2$, respectively. Both training and testing sets include pixels belonging to all classes of those datasets. The experiments were performed using the default settings given

[11]The proposed split of Pavia and Gray-Potsdam datasets are available in https://gitlab.inria.fr/dsantana/attributes-profiles-survey-source-codes



(a) (b)

Fig. 8: (a) Original Gray-Potsdam ground-truth. (b) Connected components that contributed to the training set of the partition $GrayPotsdam_2$.

in Sec. IV: 4-connectivity and 8-bit quantization. We applied the same set of thresholds for the area and moment of inertia attributes used in the previous experiments on Gray-Potsdam and Pavia.

The main challenge of using those new dataset partitions is that a sample of a given semantic class is not always representative of other samples of the same class in different regions the image. This is particularly true for the bare soil and meadows classes of Pavia. In the classification results based only on pixel intensities, several meadow samples are classified as bare soil and vice-versa, as shown in Figure 5)(d). A similar observation can be drawn for the building and impervious surface classes of Gray-Potsdam (see Figure 6(c)).

Tables XIII and XIV present the classification results computed on the partitions $Pavia_1$ (Fig. 3) and $Pavia_2$ (Fig. 9). First, in Table XIII, we see a degradation in the baseline classification results (4 PC) using the new partition, which is due to the training set not including diversified samples of every class. Then, we can observe an even larger degradation in terms of AP results. With $Pavia_1$, all APs improve the baseline results by 20.21-31.49% in terms of overall accuracy while that, with $Pavia_2$, the improvement ranges from 1.65 to 15.99%. Moreover, with $Pavia_2$, as shown in Tab. XIV, none of the AP-based methods outperform the baseline with respect to the brick class, and we observe a less remarkable improvement for the meadow class. On the other hand, the best scores achieved with $Pavia_2$ for the other seven classes are comparable with the best scores achieved with $Pavia_1$. For the Pavia dataset, we can conclude that the classification

results with AP-based methods presented in Sec. IV cannot be generalized to more realistic scenarios (with better separation between training and test pixels). In fact, it appears that APs are much less useful when the training and test samples belong to regions with dissimilar spatial and geometric properties. For instance, the training and test pixels of the asphalt class all belong to thin and elongated regions, which may be the reason why the results for this class are improved by all AP-based methods with respect to the baseline. On the other hand, the pixels in the training and test sets of the meadow class belong to components of very different shapes, which may partially explain the worse results on this class.

Fig. 9(d)-(n) illustrates the classification results of Table XIII. We can observe that all AP-based methods improve considerably the classification results of the bare soil class (in light green) with respect to the baseline results (Fig. 9(d)). This can be due to the training and test pixels sharing nodes/features at higher levels of the trees. On the other hand, the classification results of the shadow and meadow classes are worsened by at least half of those methods.

We now compare the results between the partitions $GrayPotsdam_1$ and $GrayPotsdam2$ of Gray-Potsdam dataset. Tab. XV presents the classification results obtained with the partitions $GrayPotsdam_1$ (already given in Tab. VIII) and $GrayPotsdam_2$. Similarly to Pavia, we can observe a significant drop in classification accuracy for all AP-based methods when the new partition $GrayPotsdam_2$ is used. While APs improve the baseline scores by up to 31.38% on the $GrayPotsdam_1$ partition, the classification improvements do not exceed 10.55% on the new partition $GrayPotsdam_2$. Moreover, the classification results of two of the six classes (low vegetation and impervious surfaces) are worsened by most of the AP-based methods, as shown in Tab. XVI.

Fig. 10(d)-(n) illustrate the classification results on the crop of Gray-Potsdam given in Fig. 10(b) (same as Fig. 6(b)) obtained with the partition $GrayPotsdam_2$. The regions of the cropped image which contributed to the training set of $GrayPotsdam_2$ are shown in Fig. 10(c). Our main observation is that the classification results of the regions that do not contribute to the training sets are much poorer if compared to other regions of the same class. For instance, among the pixels belonging to the largest connected component of the the impervious class (which does not contribute to the training sets), virtually no pixels are correctly classified by the tested methods. In contrast, the pixels of this region are fairly well classified when using the partition $GrayPotsdam_1$, as shown in Fig. 6.

Overall, using the new partitions $Pavia_2$ and $GrayPotsdam_2$, we observed a very significant drop of performance across all approaches. Moreover, the drop in performance occurred at different degrees for each thematic class of the datasets. This may be linked to the fact that some classes are composed of regions with similar geometric properties (*e.g.* the 'asphalt' class of Pavia and the 'car' class of Gray-Potsdam) while this is not true for the other classes (*e.g.* the 'meadow' class of Pavia and the 'low vegetation' class of Gray-Potsdam). However, a deeper investigation is necessary to confirm this assumption.

B. Generalization of APs to multiple-image datasets

Though the partition $Pavia_2$ better separates training and test samples when compared to $Pavia_1$, training and test pixels of $Pavia_2$ may still share features obtained from nodes at higher levels of the tree. The same holds for the partition $GrayPotsdam_2$. In this section, we go one step further to completely separate the computation of training and test features.

As mentioned previously, the Potsdam data set is composed of several patches covering a large urban scene. The original ISPRS labeling contest consisted in providing the classification labels for the pixels on the testing patches based on the information given by the training patches. Hence, extracting training and testing features from the same tree is only possible if all patches are connected in the real scene. Otherwise, we would need to compute tree representations separately on the training and testing patches. To approach the generalization of APs to this kind of problem, we split the Gray-Potsdam image in a way that two independent component trees can be computed for the training and testing samples. This way, we ensure that the training and testing pixels will not share any nodes along the trees. Alternatively, we could have chosen two neighbour image patches from the Potsdam dataset, but it wouldn't contribute more to the generalization of the method than splitting Gray-Potsdam in two.

The Gray-Potsdam dataset was divided in two halves (see Fig. 11) such that ten sets of training and test samples were extracted from the upper and lower half, respectively. From the upper half, we randomly selected $60,000$ pixels of each class, as done for the other partitions of this dataset. The test set is composed of all $18,000,000$ pixels in the lower half of the image. This partition of Gray-Potsdam will be denoted as $GrayPotsdam_3$ in the remainder of this section. To evaluate the generalization of APs in this scenario, we performed experiments following two approaches: the standard one, where a single hierarchical representation is computed on the whole image and, then, used to extract training and test features; and a new approach, where two independent trees are computed on the training and test images and, hence, the training and test features are obtained from distinct trees.

Tables XVII and XVIII present the overall and per-class classification results, respectively, with the partition $GrayPotsdam_3$ following those two approaches.

Our first observation is that the baseline results on the partition $GrayPotsdam_3$ present slightly higher scores when compared to the baseline results on $GrayPotsdam_1$ and $GrayPotsdam_2$. On the other hand, we observe an even larger drop in performance for all AP-based approaches with respect to the previous experiments. Following the usual approach, with a single tree computed from the whole data, six over nine methods outperform the baseline. The best method (ω-AP), improves the baseline by 5.85%, 10.58% and 8.63% in terms of OA, AA and κ scores, respectively. On the other hand, with two trees computed independently on the training and test images, only three over nine methods outperform the baseline to a smaller degree. The best approach ($FP_{\mu+a}$), improves the baseline by 3.88%, 5.46% and 3.96% in terms of OA, AA

TABLE XIII: Classification results of the Pavia dataset performed on the training/testing sets from standard partition $Pavia_1$ (Fig. 3) and new partition $Pavia_2$ (Fig. 9).

Method	**Dimension**	**Classification result** ($Pavia_1$)			**Classification result** ($Pavia_2$)		
		OA (%)	AA (%)	$\kappa \times 100$	OA (%)	AA (%)	$\kappa \times 100$
4 PC	4	65.27 ± 0.25	74.88 ± 0.20	56.93 ± 0.27	59.58 ± 0.74	73.44 ± 0.45	50.76 ± 0.76
AP-maxT	80	89.21 ± 0.63	87.37 ± 0.19	85.42 ± 0.80	61.23 ± 0.68	82.27 ± 0.36	54.42 ± 0.69
AP-minT	80	87.11 ± 2.00	92.62 ± 0.61	83.23 ± 2.41	68.72 ± 2.72	79.89 ± 1.06	62.03 ± 2.88
AP	152	90.57 ± 2.60	93.13 ± 0.67	87.53 ± 3.32	64.49 ± 1.35	84.00 ± 0.74	57.90 ± 1.38
SDAP	80	94.10 ± 0.21	93.85 ± 0.37	92.16 ± 0.27	76.63 ± 3.70	82.05 ± 1.08	**70.85 ± 4.09**
α-AP	80	95.46 ± 0.59	95.25 ± 1.23	93.91 ± 0.79	64.72 ± 0.89	76.02 ± 1.19	57.32 ± 1.11
ω-AP	80	96.33 ± 0.40	**97.17 ± 0.89**	95.08 ± 0.54	63.62 ± 1.50	74.59 ± 0.78	55.96 ± 1.55
FP$_\mu$	152	**96.76 ± 0.15**	97.05 ± 0.19	**95.65 ± 0.21**	**75.57 ± 1.43**	**85.36 ± 2.14**	69.61 ± 1.63
FP$_a$	152	85.48 ± 0.98	93.94 ± 0.55	81.36 ± 1.16	66.40 ± 1.17	81.48 ± 0.39	59.80 ± 1.17
FP$_{\mu+a}$	304	96.15 ± 0.14	96.74 ± 0.32	94.82 ± 0.19	72.06 ± 2.91	82.83 ± 0.83	65.81 ± 3.15
TF-AP	72	93.98 ± 0.53	96.26 ± 0.29	92.04 ± 0.68	69.25 ± 5.60	84.46 ± 1.80	63.22 ± 5.94

TABLE XIV: Classification results per class of the Pavia dataset performed on the partition $Pavia_2$ (Fig. 9(b)-(c)).

Method	**Dimension**	**Accuracy per class** (%)								
		Asphalt	Meadow	Gravel	Tree	Metal	Soil	Bitumen	Brick	Shadow
4 PC	4	82.34 ± 0.97	41.69 ± 1.49	44.88 ± 2.67	98.09 ± 0.29	98.43 ± 0.47	59.82 ± 1.80	75.59 ± 3.21	**60.84 ± 3.01**	99.28 ± 0.16
AP-maxT	80	95.06 ± 1.34	30.95 ± 1.30	87.22 ± 2.68	98.25 ± 0.19	99.70 ± 0.11	94.68 ± 0.75	97.43 ± 0.42	37.23 ± 0.16	99.90 ± 0.15
AP-minT	80	92.49 ± 0.80	49.23 ± 6.13	86.24 ± 8.02	98.64 ± 0.21	99.33 ± 0.39	97.61 ± 0.46	99.90 ± 0.09	38.20 ± 0.21	57.41 ± 1.14
AP	152	97.50 ± 1.34	35.74 ± 2.82	92.41 ± 4.09	98.85 ± 0.17	99.69 ± 0.11	98.34 ± 0.87	**100.0 ± 0.00**	37.45 ± 0.45	96.00 ± 6.05
SDAP	80	**98.34 ± 0.44**	**64.60 ± 8.24**	89.62 ± 6.95	96.04 ± 0.39	99.44 ± 0.20	**99.34 ± 0.19**	99.71 ± 0.34	37.16 ± 0.03	54.16 ± 0.13
α-AP	80	95.71 ± 0.55	43.02 ± 0.47	32.89 ± 0.04	**99.10 ± 0.18**	99.10 ± 0.25	98.56 ± 0.16	99.50 ± 0.03	41.13 ± 10.79	75.18 ± 1.71
ω-AP	80	95.65 ± 0.41	41.58 ± 3.25	32.89 ± 0.07	99.06 ± 0.19	99.15 ± 0.28	98.39 ± 0.30	99.51 ± 0.06	38.09 ± 1.46	67.01 ± 7.05
FP$_\mu$	152	96.86 ± 0.93	60.86 ± 2.52	98.62 ± 1.14	98.94 ± 0.15	99.70 ± 0.13	96.41 ± 0.88	**100.0 ± 0.00**	38.17 ± 2.37	78.65 ± 17.0
FP$_a$	152	95.33 ± 1.21	40.76 ± 2.82	99.53 ± 0.49	98.85 ± 0.20	99.68 ± 0.17	98.88 ± 0.35	**100.0 ± 0.00**	41.30 ± 4.53	59.01 ± 1.31
FP$_{\mu+a}$	304	96.32 ± 1.34	53.13 ± 6.58	**99.84 ± 0.09**	98.88 ± 0.16	**99.73 ± 0.11**	98.28 ± 0.40	**100.0 ± 0.00**	40.70 ± 4.47	58.63 ± 1.76
TF-AP	72	95.72 ± 0.59	45.59 ± 11.73	62.79 ± 6.58	98.99 ± 0.15	99.69 ± 0.12	98.72 ± 0.27	**100.0 ± 0.00**	58.70 ± 9.10	**99.95 ± 0.08**

TABLE XV: Classification results obtained with the partitions $GrayPotsdam_1$ and $GrayPotsdam_2$ of the Gray-potsdam dataset.

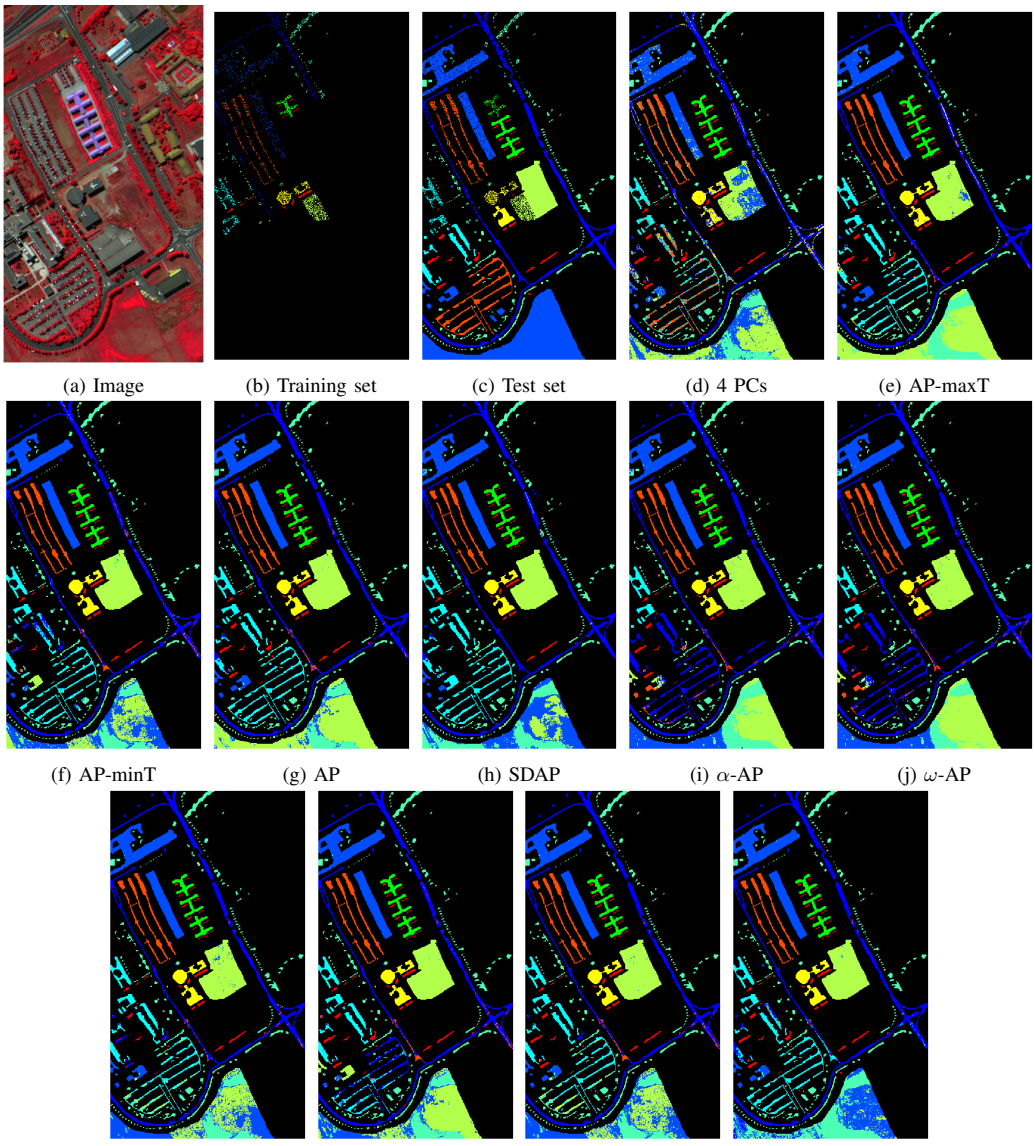
Method	**Dimension**	**Classification result** ($GrayPotsdam_1$)			**Classification result** ($GrayPotsdam_2$)		
		OA (%)	AA (%)	$\kappa \times 100$	OA (%)	AA (%)	$\kappa \times 100$
Gray-values	1	46.56 ± 0.42	34.30 ± 0.02	30.84 ± 0.32	47.76 ± 0.29	34.34 ± 0.04	32.15 ± 0.21
AP-maxT	20	63.65 ± 0.14	59.68 ± 0.02	52.65 ± 0.15	52.80 ± 0.19	45.39 ± 0.09	39.13 ± 0.16
AP-minT	20	58.00 ± 0.23	56.55 ± 0.02	47.13 ± 0.21	46.58 ± 0.16	44.32 ± 0.09	32.97 ± 0.14
AP	38	75.10 ± 0.05	77.56 ± 0.02	67.83 ± 0.05	57.11 ± 0.19	55.54 ± 0.12	44.50 ± 0.23
SDAP	20	76.19 ± 0.08	78.25 ± 0.02	69.20 ± 0.09	55.77 ± 0.14	54.14 ± 0.09	42.92 ± 0.15
α-AP	20	68.32 ± 0.07	67.94 ± 0.04	59.47 ± 0.08	**58.31 ± 0.10**	53.23 ± 0.21	**46.38 ± 0.12**
ω-AP	20	68.10 ± 0.06	67.78 ± 0.03	59.23 ± 0.06	58.17 ± 0.13	53.36 ± 0.20	46.26 ± 0.18
FP$_\mu$	38	77.14 ± 0.05	79.41 ± 0.02	70.42 ± 0.06	57.35 ± 0.29	**56.16 ± 0.14**	44.87 ± 0.34
FP$_a$	38	77.19 ± 0.05	79.46 ± 0.02	70.47 ± 0.05	55.53 ± 0.29	54.81 ± 0.25	42.63 ± 0.36
FP$_{\mu+a}$	76	**77.94 ± 0.04**	**80.17 ± 0.03**	**71.43 ± 0.05**	56.62 ± 0.40	55.68 ± 0.19	43.90 ± 0.46
TF-AP	18	72.34 ± 0.05	74.93 ± 0.04	64.33 ± 0.06	55.52 ± 0.09	54.09 ± 0.07	42.75 ± 0.09

TABLE XVI: Classification results per class of the Gray-Potsdam data performed on the partition $GrayPostam_2$.

Method	**Dimension**	**Accuracy per class** (%)					
		Background	Trees	Cars	Buildings	Low vegetation	Impervious surfaces
Gray-values	4	11.42 ± 0.70	15.64 ± 1.46	8.37 ± 0.26	48.01 ± 0.89	73.23 ± 1.48	49.38 ± 0.94
AP-maxT	80	33.17 ± 0.58	23.10 ± 1.17	47.90 ± 0.42	62.83 ± 0.60	**73.86 ± 1.11**	31.50 ± 0.50
AP-minT	80	**47.83 ± 1.01**	28.56 ± 0.68	46.29 ± 0.29	61.01 ± 0.35	33.92 ± 0.47	48.29 ± 0.88
AP	152	45.98 ± 0.32	45.31 ± 0.25	77.78 ± 0.25	80.02 ± 0.34	37.70 ± 0.38	46.44 ± 0.25
SDAP	80	46.42 ± 0.41	41.47 ± 0.28	74.75 ± 0.38	77.93 ± 0.21	37.66 ± 0.35	46.59 ± 0.17
α-AP	80	35.68 ± 1.42	43.50 ± 0.29	63.38 ± 0.35	79.05 ± 0.17	37.86 ± 0.23	**59.91 ± 0.34**
ω-AP	80	35.34 ± 1.29	43.44 ± 0.26	65.02 ± 0.42	78.67 ± 0.15	37.82 ± 0.43	59.88 ± 0.36
FP$_\mu$	152	46.48 ± 0.21	**47.49 ± 0.41**	**79.30 ± 0.43**	**80.08 ± 0.73**	36.93 ± 0.33	46.71 ± 0.31
FP$_a$	152	48.67 ± 1.36	44.95 ± 0.30	76.30 ± 0.38	77.22 ± 0.67	35.63 ± 0.30	46.08 ± 0.33
FP$_{\mu+a}$	304	47.53 ± 0.62	46.52 ± 0.36	78.34 ± 0.44	79.18 ± 0.99	35.85 ± 0.41	46.67 ± 0.47
TF-AP	72	40.13 ± 0.26	43.30 ± 0.20	78.68 ± 0.18	76.15 ± 0.27	50.54 ± 0.28	35.73 ± 0.21

and κ scores, respectively. Comparing the results obtained with those two approaches, we conclude that the results obtained with a single tree are better in general. We attribute this to the fact that training and test pixels share nodes at higher levels of the trees computed on the whole data, leading to more similar attribute values.

In terms of accuracy per class (see Tab. XVIII), we see that, for the classes 'background', 'trees', 'cars' and 'buildings',



(a) Image (b) Training set (c) Test set (d) 4 PCs (e) AP-maxT

(f) AP-minT (g) AP (h) SDAP (i) α-AP (j) ω-AP

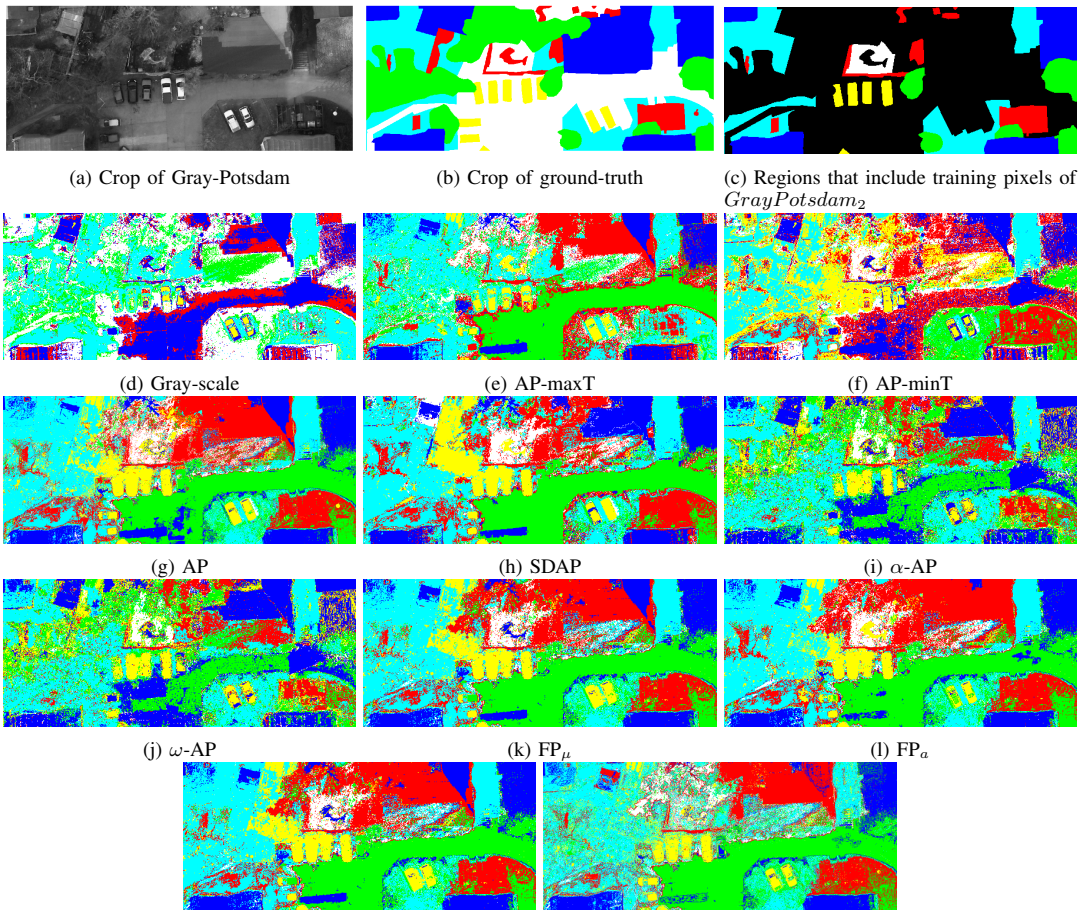
(k) FP_{μ} (l) FP_{a} (m) $\text{FP}_{\mu+a}$ (n) TF-AP

Fig. 9: Classification results of Pavia dataset (using the partition $Pavia_2$) based on the proposed training and testing sets. : trees, : gravel, : meadows, : asphalt, : metal, : bare soil, : bitumen, : shadows, : bricks.

most of the AP-based methods, computed with either one or two independent trees, improve the baseline results. In particular, the largest improvements were observed for the class 'cars', which happens to be composed of the most homogeneous regions in terms of shape and size. Whereas, this was not the case for the 'low vegetation' and 'impervious surface' classes: the performance on those two classes is degraded when APs are used for classification.

From those results, we conclude that the features extracted from APs can be useful in the classification datasets composed of several images extracted from a larger mosaic of images from the same scene. If a single tree representation can be computed from the whole scene, in a semi-supervised scenario, the results can be more promising than if independent trees are computed on each image separately. Moreover, the accuracy scores per class raise the question of whether other geometric and topological attributes could boost the performance for the classes with less homogeneous shapes (like low vegetation and impervious surfaces).

The experiments described in this section highlight some



(a) Crop of Gray-Potsdam (b) Crop of ground-truth (c) Regions that include training pixels of $GrayPotsdam_2$

(d) Gray-scale (e) AP-maxT (f) AP-minT

(g) AP (h) SDAP (i) α-AP

(j) ω-AP (k) FP_{μ} (l) FP_{a}

(m) $\text{FP}_{\mu+a}$ (n) TF-AP

Fig. 10: Classification results of a crop of Gray-Potsdam corresponding to the results of Table XV. : impervious surfaces, : buildings, : low vegetation, : trees, : cars, : background.

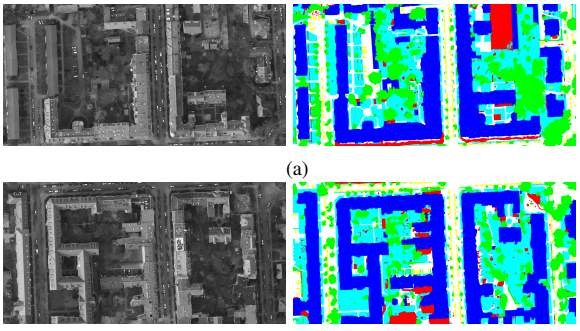
TABLE XVII: Classification results of the Gray-Potsdam dataset with the partition $GrayPotsdam_3$ following two approaches: training and test features extracted from a single tree computed on the whole data; and from independent trees computed on the training and test images.

Method	**Dimension**	**Classification result ($Potsdam_3$, single tree)**			**Classification result ($Potsdam_3$, two trees)**		
		OA (%)	AA (%)	$\kappa \times 100$	OA (%)	AA (%)	$\kappa \times 100$
Gray-level	1	48.04 ± 0.67	33.22 ± 0.14	31.68 ± 0.58	48.04 ± 0.67	33.22 ± 0.14	31.68 ± 0.58
AP-maxT	20	50.89 ± 0.42	39.41 ± 0.19	35.11 ± 0.45	41.07 ± 1.09	33.45 ± 0.60	25.22 ± 0.97
AP-minT	20	36.34 ± 0.31	32.82 ± 0.23	21.87 ± 0.25	37.60 ± 0.64	31.24 ± 0.37	22.22 ± 0.56
AP	38	47.99 ± 0.26	40.22 ± 0.17	32.48 ± 0.30	48.02 ± 0.30	37.49 ± 0.28	31.71 ± 0.37
SDAP	20	48.47 ± 0.31	37.98 ± 0.28	32.35 ± 0.39	48.28 ± 0.32	36.97 ± 0.38	31.68 ± 0.37
α-AP	20	53.80 ± 0.26	43.32 ± 0.19	40.20 ± 0.29	46.33 ± 0.22	31.43 ± 0.31	26.66 ± 0.43
ω-AP	20	**53.89 ± 0.22**	**43.80 ± 0.18**	**40.31 ± 0.25**	46.21 ± 0.38	31.05 ± 0.29	26.00 ± 0.50
FP_{μ}	38	49.66 ± 0.22	39.76 ± 0.17	34.42 ± 0.27	49.77 ± 0.64	37.31 ± 0.26	33.30 ± 0.67
FP_{a}	38	45.03 ± 0.51	37.99 ± 0.27	29.61 ± 0.47	46.58 ± 0.38	36.33 ± 0.30	29.57 ± 0.45
$\text{FP}_{\mu+a}$	76	49.82 ± 0.32	40.70 ± 0.37	34.89 ± 0.38	**51.92 ± 0.39**	**38.68 ± 0.33**	**35.64 ± 0.53**

of the challenges that we can encounter when using APs in different contexts of image classification as, for example, the selection of suitable attributes. Even though the improvements are less significant when compared to the standard data partitions employed in published works (as explained in Section IV), we are still able to benefit from the spatial features extracted from APs. This opens a path for further investigations on the extension of APs for image classification

TABLE XVIII: Classification results per class obtained with the partition $GrayPotsdam_3$ following two approaches: training and test features extracted from a single tree computed on the whole data; and from independent trees computed on the training and test images.

Method	**Trees**	**Accuracy per class (%)**					
		Background	Trees	Cars	Buildings	Low vegetation	Impervious surfaces
Gray-level	-	1.59 ± 0.94	15.46 ± 1.25	10.99 ± 0.40	49.76 ± 1.90	76.20 ± 1.05	45.28 ± 1.35
AP-maxT	1	4.14 ± 0.35	19.17 ± 0.78	43.18 ± 0.52	60.55 ± 0.86	73.20 ± 0.78	36.20 ± 0.92
	2	7.14 ± 1.16	29.22 ± 3.02	44.43 ± 1.09	62.89 ± 0.93	33.01 ± 4.28	24.01 ± 2.68
AP-minT	1	29.45 ± 1.08	38.56 ± 1.08	25.41 ± 1.13	40.48 ± 0.84	42.99 ± 0.89	20.00 ± 1.28
	2	20.89 ± 0.79	38.66 ± 2.29	17.87 ± 0.64	41.74 ± 1.31	47.67 ± 1.75	20.58 ± 2.49
AP	1	10.84 ± 0.26	40.69 ± 0.74	50.81 ± 0.71	65.48 ± 0.63	49.32 ± 1.40	24.17 ± 0.77
	2	5.48 ± 0.37	38.66 ± 0.97	38.95 ± 0.51	67.15 ± 0.46	48.29 ± 1.29	26.43 ± 1.41
SDAP	1	8.71 ± 0.24	34.81 ± 0.95	40.25 ± 1.04	67.77 ± 0.33	52.98 ± 1.35	23.36 ± 0.58
	2	4.98 ± 0.49	30.96 ± 0.53	40.03 ± 1.59	69.60 ± 0.35	50.58 ± 0.79	25.64 ± 0.76
α-AP	1	9.57 ± 0.89	34.49 ± 0.88	43.74 ± 0.38	68.10 ± 0.13	54.04 ± 1.24	49.99 ± 1.60
	2	3.94 ± 0.56	11.46 ± 2.21	25.43 ± 1.39	71.73 ± 0.34	56.21 ± 1.45	19.84 ± 0.92
ω-AP	1	9.43 ± 0.88	34.39 ± 0.67	46.55 ± 0.37	67.97 ± 0.33	54.37 ± 0.98	50.07 ± 1.33
	2	2.39 ± 0.68	12.30 ± 1.90	24.72 ± 0.74	72.07 ± 0.87	55.21 ± 1.76	19.63 ± 0.78
FP_μ	1	10.23 ± 0.24	40.04 ± 0.84	42.02 ± 0.62	68.84 ± 0.45	49.44 ± 1.06	27.97 ± 0.90
	2	4.13 ± 0.31	36.16 ± 0.87	35.64 ± 0.67	71.73 ± 1.43	52.23 ± 1.42	23.97 ± 1.54
FP_a	1	10.27 ± 0.60	42.05 ± 1.49	48.75 ± 1.19	63.71 ± 1.09	41.17 ± 1.70	21.99 ± 1.04
	2	5.35 ± 1.10	32.49 ± 1.20	40.71 ± 1.33	66.74 ± 0.93	46.99 ± 1.45	25.67 ± 1.66
$FP_{\mu+a}$	1	10.0 ± 0.22	41.32 ± 1.50	48.38 ± 1.46	69.90 ± 0.52	49.26 ± 1.26	25.32 ± 0.88
	2	2.42 ± 0.39	37.94 ± 1.18	36.36 ± 0.81	75.43 ± 0.43	50.85 ± 1.69	29.07 ± 0.83



(a)

(b)

Fig. 11: Training and testing sets from Gray-Potsdam dataset, made respectively on the upper half (a) and lower half (b). Both grayscale images and their ground-truth are provided.

using multiple images.

VI. Future directions

Despite a decade of firm advances, and a wide acclaim by the scientific community, many AP related research questions still remain unanswered while new ones are added often due to constant technological advances in terms of image acquisition. The proliferation of satellites and active & passive sensor types, raises the important question of how to represent and analyze through AP heterogeneous as well as multi-resolution data; e.g. optical and SAR. Likewise, the ever increasing temporal resolutions also present the challenge of handling multi-temporal data in the context of hierarchical image representation and processing. Also, an additional significant research direction is the AP based analysis of multivariate data (either multispectral or hyperspectral), since there is no widely accepted multivariate morphology framework as of yet.

Moreover, given the regular availability of large scale data through missions such as Sentinel-1 and Sentinel-2, more general research directions include the computation of domain invariant features and boosting the already high scalability of AP. Furthermore, given the capacity of AP for effective content description through relatively few training samples, and the ground breaking content description performance of deep networks, the combination of their potentials constitutes a powerful concept worthy of pursuing.

Last but not least, regarding the generalization of APs to real-world scenarios, future directions include experiments with multiple images of a dataset (*e.g.* training and testing on multiple image patches of the Potsdam dataset) and a study on the generalization capacity of APs across datasets (with similar image resolutions).

VII. Conclusion

APs have replaced morphological profiles as an effective spatial-spectral pixel description tool, and reinforced them in terms of both computational efficiency and flexibility, thus rendering them one of the paramount approaches of their field during the last decade, prior to the advent of deep learning. They have been employed extensively by a plethora of researchers, referenced in hundreds of publications while having been extended in a wide variety of ways.

On the contrary of past works, this survey has provided an extensive review of an entire decade of AP related developments, organized according to each of the AP calculation steps, as well as in terms of adaptation strategies to multivariate data, underlying tree representations, attribute selection and very recent post-processing strategies.

Furthermore, a comprehensive series of experiments has been conducted with multiple datasets, in order to quantify the relative performances of major AP variants using the standard, as well as additional parameters, that have been investigated

for the first time in the state of the art. And it has turned out that they have a significant effect on classification performance despite being often overlooked: image quantization level and data connectivity. Our results have confirmed that despite their age, the recent AP variants are powerful enough to compete in the case of some datasets even against deep learning. In addition, for the sake of reproducibility, all experiments have been conducted with a publicly accessible software library.

Moreover, one more significant contribution of this survey, is raising the issue of data division in the context of the underlying evaluation protocol. We have underlined the pitfalls of using a single tree structure for the entire image, as is commonly conducted in the state of the art, and proposed a solution through a spatial subdivision of the image with multiple resulting trees to simulate a real deployment scenario. Our findings have shown very important performance gaps that can otherwise lead to false generalization conclusions.

VIII. Acknowledgment

This work was supported by the ANR Multiscale project under the reference ANR-18-CE23-0022. E. Aptoula was supported by the Tubitak project 118E258. The authors would like to thank Prof. Paolo Gamba and ISPRS for making available the Pavia University and Potsdam datasets, respectively.

References

[1] D. A. Landgrebe, *Signal Theory Methods in Multispectral Remote Sensing*. New York: Wiley, 2003.

[2] M. Pesaresi and J. A. Benediktsson, "A new approach for the morphological segmentation of high-resolution satellite imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 39, no. 2, pp. 309–320, 2001.

[3] A. Plaza, P. Martinez, R. Perez, and J. Plaza, "A new approach to mixed pixel classification of hyperspectral imagery based on extended morphological profiles," *Pattern Recogn.*, vol. 37, no. 6, pp. 1097–1116, 2004.

[4] J. A. Palmason, J. A. Benediktsson, J. R. Sveinsson, and J. Chanussot, "Classification of hyperspectral data from urban areas using morphological preprocessing and independent component analysis," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, vol. 1, 2005, pp. 176–179.

[5] J. A. Benediktsson, J. A. Palmason, and J. R. Sveinsson, "Classification of hyperspectral data from urban areas based on extended morphological profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 3, pp. 480–491, 2005.

[6] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson, "Spectral and spatial classification of hyperspectral data using svms and morphological profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 11, pp. 3804–3814, 2008.

[7] M. Dalla Mura, J. A. Benediktsson, B. Waske, and L. Bruzzone, "Morphological attribute profiles for the analysis of very high resolution images," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 10, pp. 3747–3762, 2010.

[8] E. J. Breen, , and R. Jones, "Attribute openings, thinnings, and granulometries," *Computer Vision and Image Understanding*, vol. 64, no. 3, pp. 377–389, 1996.

[9] L. Najman and M. Couprie, "Building the component tree in quasi-linear time," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3531–3539, November 2006.

[10] F. Merciol, T. Balem, and S. Lefèvre, "Efficient and large-scale land cover classification using multiscale image analysis," in *ESA Conf. on Big Data from Space (BiDS)*, 2017.

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

[12] F. Merciol, L. Faucqueur, B. B. Damodaran, P.-Y. Rémy, B. Desclée, F. Dazin, S. Lefèvre, A. Masse, and C. Sannier, "Geobia at the terapixel scale: Toward efficient mapping of small woody features from heterogeneous vhr scenes," *ISPRS International Journal of Geo-Information*, vol. 8, no. 1, p. 46, 2019.

[13] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, and B. A. Johnson, "Deep learning in remote sensing applications: A meta-analysis and review," *ISPRS journal of photogrammetry and remote sensing*, vol. 152, pp. 166–177, 2019.

[14] P. Ghamisi, M. Dalla Mura, and J. A. Benediktsson, "A survey on spectral–spatial classification techniques based on attribute profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 5, pp. 2335–2353, 2015.

[15] P. Ghamisi, E. Maggiori, S. Li, R. Souza, Y. Tarablaka, G. Moser, A. De Giorgi, L. Fang, Y. Chen, M. Chi, *et al.*, "New frontiers in spectral-spatial hyperspectral image classification: The latest advances based on mathematical morphology, markov random fields, segmentation, sparse representation, and deep learning," *IEEE geoscience and remote sensing magazine*, vol. 6, no. 3, pp. 10–43, 2018.

[16] F. Guiotte, S. Lefèvre, and T. Corpetti, "Rasterization strategies for airborne lidar classification using attribute profiles," in *2019 Joint Urban Remote Sensing Event (JURSE)*. IEEE, 2019, pp. 1–4.

[17] N. Audebert, B. Le Saux, and S. Lefèvre, "Deep learning for classification of hyperspectral data: a comparative review," *IEEE Geosci. Remote Sens. Magaz.*, vol. 7, no. 2, pp. 159–173, 2019.

[18] M.-T. Pham, S. Lefèvre, E. Aptoula, and L. Bruzzone, "Recent developments from attribute profiles for remote sensing image classification," in *ICPRAI*, 2018.

[19] M. Dalla Mura, J. Atli Benediktsson, B. Waske, and L. Bruzzone, "Extended profiles with morphological attribute filters for the analysis of hyperspectral data," *Int. J. Remote Sens.*, vol. 31, no. 22, pp. 5975–5991, 2010.

[20] M. Dalla Mura, A. Villa, J. A. Benediktsson, J. Chanussot, and L. Bruzzone, "Classification of hyperspectral images by using extended morphological attribute profiles and independent component analysis," *IEEE Geosci. Remote Sens. Lett.*, vol. 8, no. 3, pp. 542–546, 2011.

[21] Y. Wu, L. Zheng, D. Xie, R. Zhong, and Q. Chen, "Classification of high-resolution multispectral satellite remote sensing images using extended morphological attribute profiles and independent component analysis," in *Ninth International Conference on Digital Image Processing (ICDIP 2017)*, vol. 10420. International Society for Optics and Photonics, 2017, p. 104203I.

[22] S. Bernabe, P. R. Marpu, A. Plaza, M. Dalla Mura, and J. A. Benediktsson, "Spectral–spatial classification of multispectral images using kernel feature space representation," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 1, pp. 288–292, 2014.

[23] M. Imani and H. Ghassemian, "Attribute profile based feature space discriminant analysis for spectral-spatial classification of hyperspectral images," *Computers & Electrical Engineering*, vol. 62, pp. 555–569, 2017.

[24] P. R. Marpu, M. Pedergnana, M. Dalla Mura, S. Peeters, J. A. Benediktsson, and L. Bruzzone, "Classification of hyperspectral data using extended attribute profiles based on supervised and unsupervised feature extraction techniques," *Int. J. Image Data Fusion*, vol. 3, no. 3, pp. 269–298, 2012.

[25] G. Cavallaro, M. Dalla Mura, J. A. Benediktsson, and L. Bruzzone, "Extended self-dual attribute profiles for the classification of hyperspectral images," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 8, pp. 1690–1694, 2015.

[26] B. B. Damodaran, N. Courty, and S. Lefèvre, "Sparse hilbert schmidt independence criterion and surrogate-kernel-based feature selection for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 4, pp. 2385–2398, 2017.

[27] E. Aptoula, M. Dalla Mura, and S. Lefèvre, "Vector attribute profiles for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 6, pp. 3208–3220, 2016.

[28] B. B. Damodaran, J. Höhle, and S. Lefèvre, "Attribute profiles on derived features for urban land cover classification," *Photogrammetric Engineering & Remote Sensing*, vol. 83, no. 3, pp. 183–193, 2017.

[29] M.-T. Pham, S. Lefèvre, and F. Merciol, "Attribute profiles on derived textural features for highly textured optical image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 7, pp. 1125–1129, 2018.

[30] A. Taghipour and H. Ghassemian, "Hyperspectral anomaly detection using attribute profiles," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 7, pp. 1136–1140, 2017.

[31] G. A. Licciardi, A. Villa, M. Dalla Mura, L. Bruzzone, J. Chanussot, and J. A. Benediktsson, "Retrieval of the height of buildings from worldview-2 multi-angular imagery using attribute filters and geometric invariant moments," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 1, pp. 71–79, 2012.

[32] N. Falco, M. Dalla Mura, F. Bovolo, J. A. Benediktsson, and L. Bruzzone, "Change detection in vhr images based on morphological attribute profiles," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 3, pp. 636–640, 2012.

[33] M. Boldt, A. Thiele, K. Schulz, and S. Hinz, "Sar image segmentation using morphological attribute profiles," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 40, no. 3, p. 39, 2014.

[34] L. Xue, X. Yang, and Z. Cao, "Building extraction of sar images using morphological attribute profiles," in *Communications, Signal Processing, and Systems*. Springer, 2012, pp. 13–21.

[35] P. R. Marpu, K.-S. Chen, C.-Y. Chu, and J. A. Benediktsson, "Spectral-spatial classification of polarimetric SAR data using morphological profiles," in *Synthetic Aperture Radar (APSAR), 2011 3rd Int. Asia-Pacific Conf.*, 2011, pp. 1–3.

[36] A. Tombak, E. Aptoula, and K. Kayabol, "Pixel-based classification of sar images using features," in *Proceedings of 26th Signal Processing and Communications Applications Conference*, Cesme, Turkey, 2018.

[37] M. Boldt, K. Schulz, A. Thiele, and S. Hinz, "Using morphological differential attribute profiles for change categorization in high resolution sar images," *Inter. Arch. Photogramm. Remote Sens. Spatial Inf. Sci*, vol. 1, no. 1, pp. 29–34, 2013.

[38] M. Boldt, A. Thiele, K. Schulz, and S. Hinz, "Feature extraction for change analysis in sar time series," in *Earth Resources and Environmental Remote Sensing/GIS Applications VI*, vol. 9644. International Society for Optics and Photonics, 2015, p. 964410.

[39] M. Boldt, A. Thiele, K. Schulz, F. J. Meyer, and S. Hinz, "Practical approach for synthetic aperture radar change analysis in urban environments," *Journal of Applied Remote Sensing*, vol. 13, no. 3, p. 034528, 2019.

[40] S. Lefèvre and E. Aptoula, "Morphological tools for spatial and multiscale analysis of passive microwave remote sensing data," in *2016 14th Specialist Meeting on Microwave Radiometry and Remote Sensing of the Environment (MicroRad)*. IEEE, 2016, pp. 145–150.

[41] D. Mongus, N. Lukač, D. Obrul, and B. Žalik, "Detection of planar points for building extraction from lidar data based on differential morphological and attribute profiles," in *VCM 2013–The ISPRS Workshop on 3D Virtual City Modeling*, vol. 2. Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2013, pp. 21–26.

[42] M. Pedergnana, P. R. Marpu, M. Dalla Mura, J. A. Benediktsson, and L. Bruzzone, "Classification of remote sensing optical and lidar data using extended attribute profiles," *IEEE J. Sel. Topics Sig. Proc.*, vol. 6, no. 7, pp. 856–865, 2012.

[43] W. Liao, J. Chanussot, M. Dalla Mura, X. Huang, R. Bellens, S. Gautama, and W. Philips, "Taking optimal advantage of fine spatial resolution: Promoting partial image reconstruction for the morphological analysis of very-high-resolution images," *IEEE Geosci. Remote Sens. Magaz.*, vol. 5, no. 2, pp. 8–28, 2017.

[44] F. Guiotte, S. Lefèvre, and T. Corpetti, "Voxel-based attribute profiles on lidar data for land cover mapping," in *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2019, pp. 2491–2494.

[45] C. Tuna, F. Merciol, and S. Lefèvre, "Attribute profiles for satellite image time series," in *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2019, pp. 126–129.

[46] C. Tuna, B. Mirmahboub, F. Merciol, and S. Lefèvre, "Component trees for image sequences and streams," *Pattern Recognition Letters*, vol. 129, pp. 255–262, 2020.

[47] M. Zhang, P. Ghamisi, and W. Li, "Classification of hyperspectral and lidar data using extinction profiles with feature fusion," *Remote Sensing Letters*, vol. 8, no. 10, pp. 957–966, 2017.

[48] C. Kwan, D. Gribben, B. Ayhan, S. Bernabe, A. Plaza, and M. Selva, "Improving land cover classification using extended multi-attribute profiles (emap) enhanced color, near infrared, and lidar data," *Remote Sensing*, vol. 12, no. 9, p. 1392, 2020.

[49] P. Ghamisi, R. Souza, J. A. Benediktsson, L. Rittner, R. Lotufo, and X. X. Zhu, "Hyperspectral data classification using extended extinction profiles," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 11, pp. 1641–1645, 2016.

[50] D. Hong, L. Gao, R. Hang, B. Zhang, and J. Chanussot, "Deep encoder-decoder networks for classification of hyperspectral and lidar data," *IEEE Geoscience and Remote Sensing Letters*, 2020.

[51] P. Bosilj, E. Kijak, and S. Lefèvre, "Partition and inclusion hierarchies of images: A comprehensive survey," *Journal of Imaging*, vol. 4, no. 2, p. 33, 2018.

[52] C. Ronse, "Ordering partial partitions for image segmentation and filtering: merging, creating and inflating blocks," *Journal of Mathematical Imaging and Vision*, vol. 49, no. 2, pp. 202–233, 2014.

[53] M. Dalla Mura, J. Benediktsson, and L. Bruzzone, "Self-dual attribute profiles for the analysis of remote sensing images," in *Int. Symp. Math. Morpho. Appl. Sig. Image Proc.*, 2011, pp. 320–330.

[54] P. Monasse and F. Guichard, "Scale-space from a level lines tree," *Journal of Visual Communication and Image Representation*, vol. 11, no. 2, pp. 224–236, 2000.

[55] G. Cavallaro, M. Dalla Mura, J. A. Benediktsson, and A. Plaza, "Remote sensing image classification using attribute filters defined over the tree of shapes," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 7, pp. 3899–3911, 2016.

[56] P. Bosilj, B. B. Damodaran, E. Aptoula, M. Dalla Mura, and S. Lefèvre, "Attribute profiles from partitioning trees," in *Int. Symp. Math. Morpho. Its Appl. Sig. Image Proc.*, 2017, pp. 381–392.

[57] P. Soille, "Constrained connectivity for hierarchical image partitioning and simplification," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 30, no. 7, pp. 1132–1145, 2008.

[58] P. Soille, "On genuine connectivity relations based on logical predicates," in *International Conference on Image Analysis and Processing*, 2007, pp. 487–492.

[59] S. Lefèvre, L. Chapel, and F. Merciol, "Hyperspectral image classification from multiscale description with constrained connectivity and metric learning," in *IEEE Wksh. Hyper. Image and Sig. Proc.: Evol. Remote Sens. (WHISPERS)*, 2014, pp. 1–4.

[60] S. G. Koç, E. Aptoula, P. Bosilj, B. B. Damodaran, M. Dalla Mura, and S. Lefevre, "A comparative noise robustness study of tree representations for attribute profile construction," in *2017 25th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2017, pp. 1–4.

[61] P. Ghamisi, J. A. Benediktsson, and J. R. Sveinsson, "Automatic spectral–spatial classification framework based on attribute profiles and supervised feature extraction," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 9, pp. 5771–5782, 2014.

[62] P. Ghamisi, J. A. Benediktsson, G. Cavallaro, and A. Plaza, "Automatic framework for spectral–spatial classification based on supervised feature extraction and morphological attribute profiles," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2147–2160, 2014.

[63] P. R. Marpu, M. Pedergnana, M. Dalla Mura, J. A. Benediktsson, and L. Bruzzone, "Automatic generation of standard deviation attribute profiles for spectral–spatial classification of remote sensing data," *IEEE Geosci. Remote Sens. Lett.*, vol. 10, no. 2, pp. 293–297, 2013.

[64] A. Das, K. Bhardwaj, S. Patra, and L. Bruzzone, "A novel threshold detection technique for the automatic construction of attribute profiles in hyperspectral images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 1374–1384, 2020.

[65] A. Das, K. Bhardwaj, and S. Patra, "Morphological complexity profile for the analysis of hyperspectral images," in *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*. IEEE, 2018, pp. 1–6.

[66] K. Bhardwaj, S. Patra, and L. Bruzzone, "Threshold-free attribute profile for classification of hyperspectral images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 10, pp. 7731–7742, 2019.

[67] U. Bhangale, S. S. Durbha, R. L. King, N. H. Younan, and R. Vatsavai, "High performance gpu computing based approaches for oil spill detection from multi-temporal remote sensing data," *Remote Sensing of Environment*, vol. 202, pp. 28–44, 2017.

[68] E. Aptoula, "Hyperspectral image classification with multidimensional attribute profiles," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 10, pp. 2031–2035, 2015.

[69] Z. Mahmood, G. Thoonen, and P. Scheunders, "Automatic threshold selection for morphological attribute profiles," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, 2012, pp. 4946–4949.

[70] M. Pedergnana, P. R. Marpu, M. Dalla Mura, J. A. Benediktsson, and L. Bruzzone, "A novel technique for optimal feature selection in attribute profiles based on genetic algorithms," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 6, pp. 3514–3528, 2013.

[71] G. Cavallaro, N. Falco, M. Dalla Mura, L. Bruzzone, and J. A. Benediktsson, "Automatic threshold selection for profiles of attribute filters based on granulometric characteristic functions," in *Int. Symp. Math. Morpho. Its Appl. Sig. Image Proc.*, 2015, pp. 169–181.

[72] G. Cavallaro, N. Falco, M. Dalla Mura, and J. A. Benediktsson, "Automatic attribute profiles," *IEEE Trans. Image Processing*, vol. 26, no. 4, pp. 1859–1872, 2017.

[73] K. Bhardwaj and S. Patra, "An unsupervised technique for optimal feature selection in attribute profiles for spectral-spatial classification of hyperspectral images," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 138, pp. 139–150, 2018.

[74] E. Aptoula and S. Guner Koc, "Attribute profiles without thresholds," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, 2018.

[75] U. Derbashi and E. Aptoula, "Attribute profiles without thresholds through histogram based tree path description," in *Proc. IEEE Mediterranean and Middle-East Geosci. and Remote Sens. Symp. (M2GRSS)*, 2020.

[76] P. Salembier and M. H. F. Wilkinson, "Connected operators," *IEEE Signal Processing Magazine*, pp. 136–157, November 2009.

[77] E. R. Urbach, J. B. T. M. Roerdink, and M. H. F. Wilkinson, "Connected shape-size pattern spectra for rotation and scale-invariant classification of gray-scale images," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 29, no. 2, pp. 272–285, February 2007.

[78] G. Cavallaro, M. Dalla Mura, J. A. Benediktsson, and L. Bruzzone, "A comparison of self-dual attribute profiles based on different filter rules for classification," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, 2014, pp. 1265–1268.

[79] B. Demir and L. Bruzzone, "Histogram-based attribute profiles for classification of very high resolution remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 4, pp. 2096–2107, 2016.

[80] M.-T. Pham, S. Lefèvre, and E. Aptoula, "Local feature-based attribute profiles for optical remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 1199–1212, 2018.

[81] R. Battiti, B. Demir, and L. Bruzzone, "Compressed histogram attribute profiles for the classification of VHR remote sensing images," in *SPIE Remote Sens.*, 2015, pp. 96 430R–96 430R.

[82] M.-T. Pham, S. Lefèvre, E. Aptoula, and B. B. Damodaran, "Classification of VHR remote sensing images using local feature-based attribute profiles," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, 2017, pp. 1083–1086.

[83] B. Song, J. Li, M. Dalla Mura, P. Li, A. Plaza, J. M. Bioucas-Dias, J. A. Benediktsson, and J. Chanussot, "Remotely sensed image classification using sparse representations of morphological attribute profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 8, pp. 5122–5136, 2014.

[84] J. Xia, M. Dalla Mura, J. Chanussot, P. Du, and X. He, "Random subspace ensembles for hyperspectral image classification with extended morphological attribute profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 9, pp. 4768–4786, 2015.

[85] R. Bao, J. Xia, M. Dalla Mura, P. Du, J. Chanussot, and J. Ren, "Combining morphological attribute profiles via an ensemble method for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 3, pp. 359–363, 2016.

[86] E. Aptoula, M. C. Ozdemir, and B. Yanikoglu, "Deep learning with attribute profiles for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 12, pp. 1970–1974, 2016.

[87] T. Tian, L. Gao, W. Song, K.-K. R. Choo, and J. He, "Feature extraction and classification of VHR images with attribute profiles and convolutional neural networks," *Multimedia Tools and Applications*, vol. 77, no. 14, pp. 18 637–18 656, 2018.

[88] H. Teffahi, H. Yao, S. Chaib, and N. Belabid, "A novel spectral-spatial classification technique for multispectral images using extended multi-attribute profiles and sparse autoencoder," *Remote Sensing Letters*, vol. 10, no. 1, pp. 30–38, 2019. [Online]. Available: https://doi.org/10.1080/2150704X.2018.1523581

[89] Ş. G. Koç and E. Aptoula, "Hyperspectral image classification with convolutional networks trained with self-dual attribute profiles," in *2017 25th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2017, pp. 1–4.

[90] K. Zhu, Y. Chen, P. Ghamisi, X. Jia, and J. A. Benediktsson, "Deep convolutional capsule network for hyperspectral image spectral and spectral-spatial classification," *Remote Sensing*, vol. 11, no. 3, p. 223, 2019.

[91] P. Ghamisi, R. Souza, J. A. Benediktsson, X. X. Zhu, L. Rittner, and R. A. Lotufo, "Extinction profiles for the classification of remote sensing data," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 5631–5645, 2016.

[92] J. Xia, P. Ghamisi, N. Yokoya, and A. Iwasaki, "Random forest ensembles and extended multiextinction profiles for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 1, pp. 202–216, 2018.

[93] L. Fang, N. He, S. Li, P. Ghamisi, and J. A. Benediktsson, "Extinction profiles fusion for hyperspectral images classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 3, pp. 1803–1815, 2017.

[94] W. Liao, M. Dalla Mura, J. Chanussot, R. Bellens, and W. Philips, "Morphological attribute profiles with partial reconstruction," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1738–1756, 2016.

[95] D. Hong, X. Wu, P. Ghamisi, J. Chanussot, N. Yokoya, and X. X. Zhu, "Invariant attribute profiles: A spatial-frequency joint feature extractor for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, 2020.

[96] M.-T. Pham, E. Aptoula, and S. Lefèvre, "Feature profiles from attribute filtering for remote sensing image classification," *IEEE J. Sel. Topics Appl. Earth Observations Remote Sens.*, vol. 11, no. 1, pp. 249–256, 2018.

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

[98] B. Perret, G. Chierchia, J. Cousty, S. Guimarães, Y. Kenmochi, and L. Najman, "Higra: Hierarchical graph analysis," *SoftwareX*, vol. 10, p. 100335, 2019.

[99] D. Pascale, "A review of rgb color spaces," Tech. Rep., 2003. [Online]. Available: http://www.babelcolor.com/index_htm_files/A%20review%20of%20RGB%20color%20spaces.pdf

[100] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomForest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[101] P. Ghamisi, B. Höfle, and X. X. Zhu, "Hyperspectral and lidar data fusion using extinction profiles and deep convolutional neural network," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 6, pp. 3011–3024, 2016.

[102] P. G. Bascoy, P. Quesada-Barriuso, D. B. Heras, F. Argüello, B. Demir, and L. Bruzzone, "Extended attribute profiles on gpu applied to hyperspectral image classification," *The Journal of Supercomputing*, vol. 75, no. 3, pp. 1565–1579, 2019.

[103] Y. Li, H. Zhang, and Q. Shen, "Spectral–spatial classification of hyperspectral imagery with 3d convolutional neural network," *Remote Sensing*, vol. 9, no. 1, p. 67, 2017.

[104] M. H. Wilkinson, "Hyperconnectivity, attribute-space connectivity and path openings: Theoretical relationships," in *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, 2009, pp. 47–58.

[105] G. K. Ouzounis and M. H. Wilkinson, "Mask-based second-generation connectivity and attribute filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 990–1004, 2007.

SIMPLE ATTRIBUTE PROFILES USER DOCUMENTATION

This appendix contains the documentation of the Python package SAP (v0.2.4) developed during this PhD thesis. SAP is an open source library aiming to compute Attribute Profiles (and their extensions) and Pattern Spectra. The source code is available at `https://gitlab.inria.fr/fguiotte/sap`.

The online version of this documentation is available at `https://python-sap.rtfd.io`.

CHAPTER

ONE

GETTING STARTED

1.1 Installation

To install SAP simply run

```
pip install sap
```

To help develop sap or to use the latest unstable version of sap you can install the `develop` version:

```
git clone --branch develop https://gitlab.inria.fr/fguiotte/sap.git
cd sap
pip install -e .
```

CHAPTER

TWO

SAP PACKAGE

2.1 Simple Attribute Profiles

This package provides:

1. Easy to use tree structures.
2. Straight forward attribute profiles computation.

This package heavily relies on the outstanding library higra which provide efficient tree structure in C++. For more specific needs you definitely have to check this out.

2.1.1 Documentation

Documentation is available on docstrings and on the web page python-sap documentation.

Once sap imported with:

```
>>> import sap
```

Use *help()* on the module, submodules, classes and functions to print directly the docstrings.

```
>>> help(sap.trees)
```

2.1.2 Submodules access

For simplicity the submodules classes and function are directly available at the root of the module. In doing so:

```
>>> from sap import trees
>>> trees.MaxTree
```

Is equivalent to:

```
>>> import sap
>>> sap.MaxTree
```

2.2 Submodules

2.2.1 Profiles

This submodule contains the attribute profiles related classes.

Example

```
>>> import sap
>>> import numpy as np
```

```
>>> image = np.arange(5*5).reshape(5, 5)
```

Create the attribute profiles (AP) of *image* based on area attribute and three thresholds.

```
>>> aps = sap.attribute_profiles(image, {'area': [10, 100, 1000]})
>>> aps.vectorize()
[[...]]
```

Create the extended AP of *image* based on area compactness and volume attributes.

```
>>> attributes = {'compactness': [.1, .5, .7], 'volume': [10, 100]}
>>> eaps = sap.attribute_profiles(image, attributes)
>>> eaps.vectorize()
[[...]]
```

Concatenation of profiles to create complex extended profiles.

```
>>> profiles = sap.attribute_profiles(image, {'area': [10, 100]}) \
...            + sap.feature_profiles(image, {'compactness': [.3, .7]}) \
...            + sap.self_dual_attribute_profiles(image, {'height': [5, 15]})
Profiles[{'attribute': 'area',
  'filtering rule': 'direct',
  'image': -7518820387991786804,
  'name': 'attribute profiles',
  'out feature': 'altitude',
  'profiles': [{'operation': 'thinning', 'threshold': 100},
               {'operation': 'thinning', 'threshold': 10},
               {'operation': 'copy feature altitude'},
               {'operation': 'thickening', 'threshold': 10},
               {'operation': 'thickening', 'threshold': 100}]},
 {'attribute': 'compactness',
  'filtering rule': 'direct',
  'image': -7518820387991786804,
  'name': 'feature profiles',
  'out feature': 'compactness',
  'profiles': [{'operation': 'thinning', 'threshold': 0.7},
               {'operation': 'thinning', 'threshold': 0.3},
               {'operation': 'copy feature compactness'},
               {'operation': 'thickening', 'threshold': 0.3},
               {'operation': 'thickening', 'threshold': 0.7}]},
 {'attribute': 'height',
  'filtering rule': 'direct',
  'image': -7518820387991786804,
  'name': 'self dual attribute profiles',
```

(continues on next page)

(continued from previous page)

```
  'out feature': 'altitude',
  'profiles': [{'operation': 'copy feature altitude'},
               {'operation': 'sd filtering', 'threshold': 5},
               {'operation': 'sd filtering', 'threshold': 15}]}]
```

class sap.profiles.**Profiles**(*data*, *description*)

Bases: object

Base class for profiles.

Parameters

- **data** (*list of ndarray*) – List of ndarray representing profiles grouped by image or attribute filtering.
- **description** (*list of dict*) – List of dictionary containing the metadata of the profiles.

diff()

Compute the differential of profiles.

Refer to *differential()* for full documentation.

Returns **differential** – The processed differential profiles.

Return type *Profiles*

lf(*self*, *local_feature=np.mean, np.std*, *patch_size=7*)

Compute the local features of profiles

Refer to *local_features()* for full documentation.

local_feature [function or tuple of functions] The function(s) to describe the local patches.

patch_size [int] The size of the patches.

Returns **local_features** – The local features of profiles.

Return type *Profiles*

strip(*lambda x: x['operation'] != 'thinning'*)

Remove profiles according to condition. Iteration is done on profiles description.

Refer to *strip_profiles()* for full documentation.

Parameters **condition** (*function*) – The function (or lambda function) to use on profiles description to filter the profiles.

Returns **new_profiles** – Filtered profiles.

Return type *Profiles*

See also:

strip_profiles() equivalent function

strip_copy()

Remove all the copied images in profiles.

Refer to *strip_profiles_copy()* for full documentation.

Parameters **profiles** (Profiles) – The profiles to strip on the copied images.

Returns **new_profiles** – Copy of profiles without copied image.

Return type *Profiles*

See also:

strip_profiles_copy() equivalent function

vectorize()
Return the vectors of the profiles.

Refer to *vectorize()* for full documentation.

Returns **vectors** – The vectors of the profiles.

Return type numpy.ndarray

See also:

vectorize() equivalent function.

sap.profiles.**alpha_profiles**(*image*, *attribute*, *adjacency=4*, *image_name=None*, *filtering_rule='direct'*)
Compute the alpha profiles of an image.

Parameters

- **image** (*ndarray*) – The image
- **attribute** (*dict*) – Dictionary of attribute (as key, str) with according thresholds (as values, number).
- **adjacency** (*int*) – Adjacency used for the tree construction. Default is 4.
- **image_name** (*str*) – The name of the image (optional). Useful to track filtering process and display. If not set, the name is replaced by the hash of the image.
- **filtering_rule** (*str, optional*) – The filtering rule to use. It can be direct, min, max or subtractive. Default is direct.

Examples

```
>>> image = np.arange(5 * 5).reshape(5, 5)
>>> sap.alpha_profiles(image, {'area': [10, 100]})
Profiles{'attribute': 'area',
 'filtering rule': 'direct',
 'name': 'alpha profiles',
 'out feature': 'altitude',
 'profiles': [{'operation': 'copy feature altitude'},
              {'operation': 'alpha filtering', 'threshold': 10},
              {'operation': 'alpha filtering', 'threshold': 100}],
 'tree': {'adjacency': 4, 'image_hash': '44f17c0f', 'image_name': None}}
```

See also:

sap.trees.available_attributes() List available attributes.

sap.profiles.**attribute_profiles**(*image*, *attribute*, *adjacency=4*, *image_name=None*, *filtering_rule='direct'*)
Compute the attribute profiles of an image.

Parameters

- **image** (*ndarray*) – The image
- **attribute** (*dict*) – Dictionary of attribute (as key, str) with according thresholds (as values, number).
- **adjacency** (*int*) – Adjacency used for the tree construction. Default is 4.
- **image_name** (*str*) – The name of the image (optional). Useful to track filtering process and display. If not set, the name is replaced by the hash of the image.
- **filtering_rule** (*str, optional*) – The filtering rule to use. It can be direct, min, max or subtractive. Default is direct.

Examples

```
>>> image = np.arange(5*5).reshape(5,5)
```

```
>>> sap.attribute_profiles(image, {'area': [10, 100]})
Profiles{'attribute': 'area',
 'filtering rule': 'direct',
 'image': -7518820387991786804,
 'name': 'attribute profiles',
 'out feature': 'altitude',
 'profiles': [{'operation': 'thinning', 'threshold': 100},
              {'operation': 'thinning', 'threshold': 10},
              {'operation': 'copy feature altitude'},
              {'operation': 'thickening', 'threshold': 10},
              {'operation': 'thickening', 'threshold': 100}]}
```

See also:

sap.trees.available_attributes() List available attributes.

sap.profiles.**concatenate**(*(profiles_1, profiles_2, ...)*)

Concatenate a sequence of profiles.

Parameters **sequence** (*sequence of Profiles*) – The sequence of profiles to concatenate.

Returns **profiles** – The concatenated profiles.

Return type *Profiles*

Examples

```
>>> aps_a = sap.attribute_profiles(image, {'area': [10, 100]})
>>> aps_b = sap.attribute_profiles(image, {'compactness': [.1, .5]})
```

```
>>> aps = sap.concatenate((aps_a, aps_b))
```

```
>>> len(aps) == len(aps_a) + len(aps_b)
True
```

sap.profiles.**create_profiles**(*tree*, *attribute*, *out_feature='altitude'*, *filtering_rule='direct'*, *profiles_name='unknow'*)

Compute the profiles of an images. Generic function.

Parameters

- **image** (*ndarray*) – The image to be profiled.
- **attribute** (*dict*) – Dictionary of attribute (as key, str) with according thresholds (as values, iterable of thresholds).
- **tree_type** (sap.trees.Tree, *serie of sap.trees.Tree*) – Tree or pair of tree for non dual filtering (e.g. min-tree and max-tree for attribute profiles).
- **adjacency** (*int, optional*) – Adjacency used for the tree construction. Default is 4.
- **image_name** (*str, optional*) – The name of the image Useful to track filtering process and display. If not set, the name is replaced by the hash of the image.
- **out_feature** (*str or list, optional*) – Out feature of the profiles. Can be altitude (default), same or a list of feature. If same then out feature of the profiles match the filtering attribute. Refer to *feature_profiles()* and *self_dual_feature_profiles()* for more details.
- **filtering_rule** (*str, optional*) – The filtering rule to use. It can be direct, min, max or subtractive. Default is direct.
- **profiles_name** (*str, optional*) – Name of the profiles (e.g. *attribute profiles*).

Todo: out_feature takes a list of features.

Example

```
>>> image = np.arange(5*5).reshape(5, 5)
```

```
>>> sap.create_profiles(image, {'area': [5, 10]},
...                     (sap.MinTree, sap.MaxTree))
Profiles{'attribute': 'area',
 'filtering rule': 'direct',
 'image': -7518820387991786804,
 'name': 'unknow',
 'out feature': 'altitude',
 'profiles': [{'operation': 'thinning', 'threshold': 10},
              {'operation': 'thinning', 'threshold': 5},
              {'operation': 'copy feature altitude'},
              {'operation': 'thickening', 'threshold': 5},
              {'operation': 'thickening', 'threshold': 10}]}
```

sap.profiles.**differential**(*profiles*)

Compute the differential of profiles.

Parameters **profiles** (Profiles) – Attribute profiles or other profiles to process the differential on.

Returns **differential** – The processed differential profiles.

Return type *Profiles*

sap.profiles.**feature_profiles**(*image*, *attribute*, *adjacency=4*, *image_name=None*, *out_feature='same'*, *filtering_rule='direct'*)

Compute the feature profiles of an image.

Parameters

- **image** (*ndarray*) – The image
- **attribute** (*dict*) – Dictionary of attribute (as key, str) with according thresholds (as values, number).
- **adjacency** (*int*) – Adjacency used for the tree construction. Default is 4.
- **image_name** (*str*) – The name of the image (optional). Useful to track filtering process and display. If not set, the name is replaced by the hash of the image.
- **out_feature** (*str or list, optional*) – Out feature of the profiles. Can be altitude (default), same or a list of feature. If same then out feature of the profiles match the filtering attribute.
- **filtering_rule** (*str, optional*) – The filtering rule to use. It can be direct, min, max or subtractive. Default is direct.

Examples

```
>>> image = np.arange(5*5).reshape(5,5)
```

```
>>> sap.feature_profiles(image, {'area': [5, 10]})
Profiles{'attribute': 'area',
 'filtering rule': 'direct',
 'image': -7518820387991786804,
 'name': 'feature profiles',
 'out feature': 'area',
 'profiles': [{'operation': 'thinning', 'threshold': 10},
              {'operation': 'thinning', 'threshold': 5},
              {'operation': 'copy feature area'},
              {'operation': 'thickening', 'threshold': 5},
              {'operation': 'thickening', 'threshold': 10}]}
```

See also:

sap.trees.available_attributes() List available attributes.

attribute_profiles() other profiles.

sap.profiles.**local_features**(*profiles*, *local_feature=np.mean, np.std*, *patch_size=7*)

Compute the local features of profiles

Parameters

- **profiles** (Profiles) – Input Profiles.
- **local_feature** (*function or tuple of functions*) – The function(s) to describe the local patches.
- **patch_size** (*int*) – The size of the patches.

Returns **local_feature** – The local features of profiles.

Return type *Profiles*

sap.profiles.**omega_profiles**(*image*, *attribute*, *adjacency=4*, *image_name=None*, *filtering_rule='direct'*)

Compute the omega profiles of an image.

Parameters

- **image** (*ndarray*) – The image
- **attribute** (*dict*) – Dictionary of attribute (as key, str) with according thresholds (as values, number).
- **adjacency** (*int*) – Adjacency used for the tree construction. Default is 4.
- **image_name** (*str*) – The name of the image (optional). Useful to track filtering process and display. If not set, the name is replaced by the hash of the image.
- **filtering_rule** (*str, optional*) – The filtering rule to use. It can be direct, min, max or subtractive. Default is direct.

Examples

```
>>> image = np.arange(5 * 5).reshape(5, 5)
>>> sap.omega_profiles(image, {'area': [10, 100]})
Profiles{'attribute': 'area',
 'filtering rule': 'direct',
 'name': 'omega profiles',
 'out feature': 'altitude',
 'profiles': [{'operation': 'copy feature altitude'},
              {'operation': '() filtering', 'threshold': 10},
              {'operation': '() filtering', 'threshold': 100}],
 'tree': {'adjacency': 4, 'image_hash': '44f17c0f', 'image_name': None}}
```

See also:

sap.trees.available_attributes() List available attributes.

sap.profiles.**self_dual_attribute_profiles**(*image*, *attribute*, *adjacency=4*, *image_name=None*, *filtering_rule='direct'*)

Compute the self dual attribute profiles of an image.

Parameters

- **image** (*ndarray*) – The image
- **attribute** (*dict*) – Dictionary of attribute (as key, str) with according thresholds (as values, number).
- **adjacency** (*int*) – Adjacency used for the tree construction. Default is 4.
- **image_name** (*str*) – The name of the image (optional). Useful to track filtering process and display. If not set, the name is replaced by the hash of the image.
- **filtering_rule** (*str, optional*) – The filtering rule to use. It can be direct, min, max or subtractive. Default is direct.

Examples

```
>>> image = np.arange(5*5).reshape(5,5)
```

```
>>> sap.self_dual_attribute_profiles(image, {'area': [10, 100]})
Profiles{'attribute': 'area',
 'filtering rule': 'direct',
 'image': -7518820387991786804,
 'name': 'self dual attribute profiles',
 'out feature': 'altitude',
 'profiles': [{'operation': 'copy feature altitude'},
              {'operation': 'sd filtering', 'threshold': 10},
              {'operation': 'sd filtering', 'threshold': 100}]}
```

See also:

sap.trees.available_attributes() List available attributes.

attribute_profiles() other profiles.

sap.profiles.**self_dual_feature_profiles**(*image*, *attribute*, *adjacency=4*, *image_name=None*, *out_feature='same'*, *filtering_rule='direct'*)

Compute the self dual features profiles of an image.

Parameters

- **image** (*ndarray*) – The image
- **attribute** (*dict*) – Dictionary of attribute (as key, str) with according thresholds (as values, number).
- **adjacency** (*int*) – Adjacency used for the tree construction. Default is 4.
- **image_name** (*str*) – The name of the image (optional). Useful to track filtering process and display. If not set, the name is replaced by the hash of the image.
- **out_feature** (*str or list, optional*) – Out feature of the profiles. Can be altitude (default), same or a list of feature. If same then out feature of the profiles match the filtering attribute.
- **filtering_rule** (*str, optional*) – The filtering rule to use. It can be direct, min, max or subtractive. Default is direct.

Examples

```
>>> image = np.arange(5*5).reshape(5,5)
```

```
>>> sap.self_dual_feature_profiles(image, {'area': [10, 100]})
Profiles{'attribute': 'area',
 'filtering rule': 'direct',
 'image': -7518820387991786804,
 'name': 'self dual feature profiles',
 'out feature': 'area',
 'profiles': [{'operation': 'copy feature area'},
              {'operation': 'sd filtering', 'threshold': 10},
              {'operation': 'sd filtering', 'threshold': 100}]}
```

See also:

sap.trees.available_attributes() List available attributes.

attribute_profiles() other profiles.

sap.profiles.**show_all_profiles**(*profiles*, *attribute=None*, *image=None*, *height=None*, *fname=None*, ***kwargs*)

Display profiles with matplotlib.

Parameters

- **profiles** (*sap.profiles.Profiles*) – The profiles to display.
- **attribute** (*sring, optional*) – Name of attribute to display. By default display all the attributes contained in profiles.
- **image** (*string, optional*) – Name of the image to display. By default display the profiles of all images.
- **height** (*scalar, optional, default: None*) – Height of the figure in inches. Automatically adjust the size of the figure to display correctly the profiles and the title with matplot.
- **fname** (*str or PathLike, optional*) – If set, the file path to save the figure. The attribute name is automatically inserted in the file name.

See also:

show_profiles() Display a profiles stack.

Notes

This is a utility function to call recursively *show_profiles*. Attribute and image filters are available to filter the profiles to display.

sap.profiles.**show_profiles**(*profiles*, *height=None*, *fname=None*, ***kwargs*)

Display a profiles stack with matplotlib.

Parameters

- **profiles** (*Profiles*) – The profiles to display. Can be only of length 1.
- **height** (*scalar, optional, default: None*) – Height of the figure in inches. Automatically adjust the size of the figure to display correctly the profiles and the title with matplot.
- **fname** (*str or PathLike, optional*) – If set, the file path to save the figure. The attribute name is automatically inserted in the file name.

See also:

show_profiles_all() Display several profiles at once.

sap.profiles.**strip_profiles**(*lambda x: x['operation'] != 'thinning'*, *profiles*)

Remove profiles according to condition. Iteration is done on profiles description (see Notes).

Parameters

- **condition** (*function*) – The function (or lambda function) to use on profiles description to filter the profiles.

- **profiles** (Profiles) – The profiles to filter.

Returns **new_profiles** – Filtered profiles.

Return type *Profiles*

Notes

The condition is tested on the description of each profiles. Considering this stack:

```
>>> aps
Profiles{'attribute': 'area',
 'image': -8884649894275650052,
 'profiles': [{'operation': 'thinning', 'threshold': 1000},
              {'operation': 'thinning', 'threshold': 100},
              {'operation': 'thinning', 'threshold': 10},
              {'operation': 'copy'},
              {'operation': 'thickening', 'threshold': 10},
              {'operation': 'thickening', 'threshold': 100},
              {'operation': 'thickening', 'threshold': 1000}]}
```

The condition function is tested on each item of the list `'profiles'`.

See also:

Profiles.strip() Remove profiles based on condition.

Examples

Strip profiles depending on thresholds level:

```
>>> image = np.random.random((100, 100))
>>> aps = sap.attribute_profiles(image, {'area': [10, 100, 1000]})
```

```
>>> sap.strip_profiles(lambda x: 'threshold' in x and x['threshold'] > 20, aps)
Profiles{'attribute': 'area',
 'image': 2376333419322655105,
 'profiles': [{'operation': 'thinning', 'threshold': 10},
              {'operation': 'copy'},
              {'operation': 'thickening', 'threshold': 10}]}
```

Strip profiles depending on operation:

```
>>> sap.strip_profiles(lambda x: x['operation'] == 'thinning', aps)
Profiles{'attribute': 'area',
 'image': 2376333419322655105,
 'profiles': [{'operation': 'copy'},
              {'operation': 'thickening', 'threshold': 10},
              {'operation': 'thickening', 'threshold': 100},
              {'operation': 'thickening', 'threshold': 1000}]}
```

sap.profiles.**strip_profiles_copy**(*profiles*)

Remove all the copied images in profiles.

Copy are the original images where profiles are computed on.

Parameters **profiles** (Profiles) – The profiles to strip on the copied images.

Returns **new_profiles** – Copy of profiles without copied image.

Return type *Profiles*

See also:

`sap.strip_profiles()` Filter profiles according to condition.

sap.profiles.**vectorize**(*profiles*)

Return the classification vectors of the profiles.

Parameters **`profiles`** (`Profiles`) – Profiles on which process the vectors.

Returns **vectors** – The vectors of the profiles.

Return type numpy.ndarray

See also:

`Profiles.vectorize()` get the vectors of profiles.

Example

```
>>> image = np.random.random((100, 100))
>>> aps = sap.attribute_profiles(image, {'area': [10, 100]})
```

```
>>> vectors = sap.vectorize(aps)
>>> vectors.shape
(5, 100, 100)
```

2.2.2 Spectra

This submodule contains pattern spectra related functions.

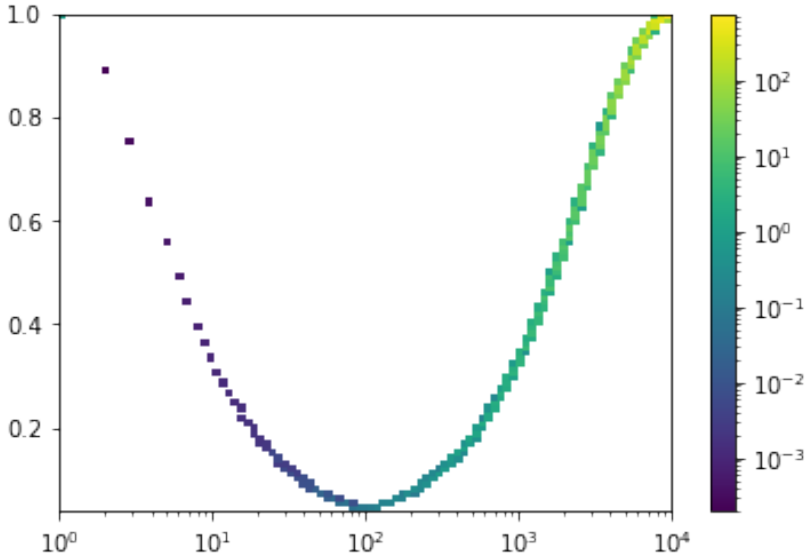
Example

```
>>> import sap
>>> import numpy as np
>>> from matplotlib import pyplot as plt
```

```
>>> image = np.arange(100 * 100).reshape(100, 100)
```

Create the pattern spectrum (PS) of *image* with attributes area and compactness with the max-tree.

```
>>> tree = sap.MaxTree(image)
>>> ps = sap.spectrum2d(tree, 'area', 'compactness', x_log=True)
>>> sap.show_spectrum(*ps)
```



sap.spectra.**get_bins**(*x*, *count=10*, *space='lin'*, *outliers=0.0*)
Return the bin edges over the values of x.

Parameters

- **x** (*ndarray*) – The values to be binned.
- **count** (*int, optional*) – Bin count to be returned. Default is 10.
- **space** (*str, optional*) – Spacing rule used to compute the bin. Can be lin for linear spacing (default) or geo for logarithmic spacing.
- **outliers** (*float, optional*) – Extremum quantiles to be considered as outliers to remove from the values before computing the bins.

Returns **bin_edges** – The edges defining the bins.

Return type ndarray, shape(count + 1,)

See also:

get_space() Return spaced numbers with min and max values.

sap.spectra.**get_space**(*vmin*, *vmax*, *thresholds=10*, *space='lin'*)
Return spaced numbers over the range defined by vmin and vmax.

Parameters

- **vmin** (*scalar*) – The min value of the range.
- **vmax** (*scalar*) – The max value of the range.
- **thresholds** (*int, optional*) – The count of samples to be returned. Default is 10.
- **space** (*str, optional*) – Spacing rule used to compute the samples. Can be lin for linear spacing (default) or geo for logarithmic spacing.

Returns **samples** – Spaced numbers over the range defined by vmin and vmax.

Return type ndarray, shape(thresholds,)

See also:

get_bins() Return the bin edges of a distribution.

Notes

When using geo spacing, the range cannot include 0. The function will offset `vmin` to 0.1 if `vmin` is 0, as a workaround.

`sap.spectra.`**`show_spectrum`**(*s*, *xedges*, *yedges*, *x_log*, *y_log*, *log_scale=True*)
Display a pattern spectrum with matplotlib.

Parameters

- **s** (*ndarray*) – The pattern spectrum.
- **xedges** (*ndarray*) – The bin edges along the x-axis.
- **yedges** (*ndarray*) – The bin edges along the y-axis.
- **x_log** (*bool*) – Parameter indicating if the x-axis is a log scale.
- **y_log** (*bool*) – Parameter indicating if the y-axis is a log scale.
- **log_scale** (*bool*) – If `True`, the colormap use a log scale. Default is `True`.

See also:

spectrum2d() Compute a 2D pattern spectrum.

Examples

```
>>> import numpy as np
>>> from matplotlib import pyplot as plt
>>> import sap
```

Create a toy image, compute the area and compactness pattern spectrum with the tree of shape of the image.

```
>>> image = np.arange(100 * 100).reshape(100, 100)
>>> tree = sap.TosTree(image)
>>> ps = sap.spectrum2d(tree, 'area', 'compactness', x_log=True)
```

Setup a matplotlib figure with 2 subplots.

```
>>> plt.figure(figsize=(10,4))
>>> plt.subplot(1, 2, 1)
```

Draw the spectrum with a linear color map.

```
>>> sap.show_spectrum(*ps, log_scale=False)
```

Decorate the subplot.

```
>>> plt.colorbar()
>>> plt.xlabel('area')
>>> plt.ylabel('compactness')
>>> plt.title('Linear colormap')
```

```
>>> plt.subplot(1, 2, 2)
```

Draw the spectrum with a log color map (default).

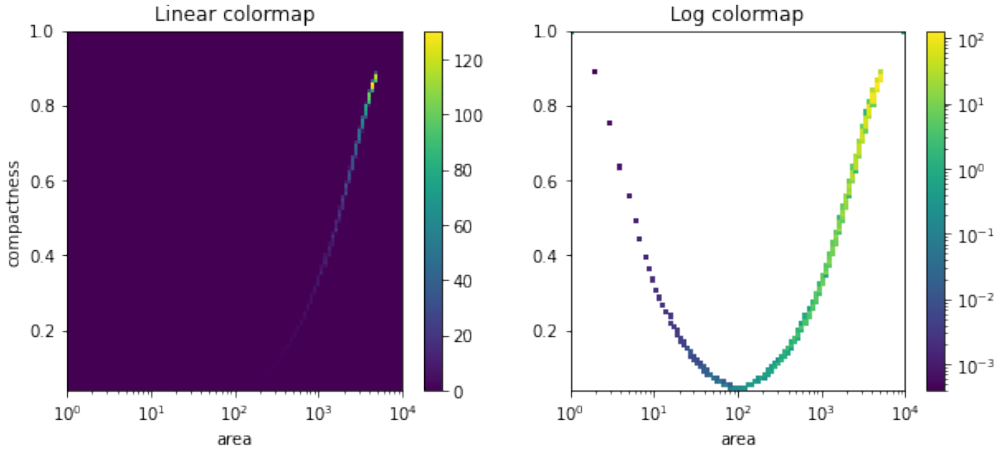
```
>>> sap.show_spectrum(*ps)
```

Decorate the subplot.

```
>>> plt.colorbar()
>>> plt.xlabel('area')
>>> plt.title('Log colormap')
```

Display the figure.

```
>>> plt.show()
```



sap.spectra.**spectrum2d**(*tree*, *x_attribute*, *y_attribute*, *x_count=100*, *y_count=100*, *x_log=False*, *y_log=False*, *weighted=True*, *normalized=True*)

Compute 2D pattern spectrum.

Parameters

- **tree** (sap.trees.Tree) – The tree used for creating the pattern spectrum.
- **x_attribute** (*str*) – The name of the attribute to be used on the x-axis.
- **y_attribute** (*str*) – The name of the attribute to be used on the y-axis.
- **x_count** (*int, optional*) – The number of bins along the x-axis. Default is 100.
- **y_count** (*int, optional*) – The number of bins along the y-axis. Default is 100.
- **x_log** (*bool, optional*) – If `True`, the x-axis will be set to be a log scale. Default is `False`.
- **y_log** (*bool, optional*) – If `True`, the y-axis will be set to be a log scale. Default is `False`.
- **weighted** (*bool, optional*) – If `True`, the pattern spectrum is weighted. Each node of the tree will be weighted according to its size. This is the normal behaviour of pattern spectrum. If `False` the spectrum is not weighted, the output is a 2D histogram counting the number of nodes. Default is `True`.

- **normalized** (*bool, optional*) – If `True`, the weights of the spectrum are normalized with the size of the image. If `False` or `weighted` is `False`, the spectrum is not normalized. Default is `True`.

Returns

- **s** (*ndarray, shape(x_count, y_count)*) – The pattern spectrum.
- **xedges** (*ndarray, shape(x_count + 1,)*) – The bin edges along the x-axis.
- **yedges** (*ndarray, shape(y_count + 1,)*) – The bin edges along the y-axis.
- **x_log** (*bool*) – The parameter x_log indicating if the x-axis is a log scale.
- **y_log** (*bool*) – The parameter y_log indicating if the y-axis is a log scale.

See also:

sap.trees.available_attributes() List the name of available attributes.

2.2.3 Trees

This submodule contains the component tree classes.

Example

Simple creation of the max-tree of an image, compute the area attributes of the nodes and reconstruct a filtered image removing nodes with area less than 100 pixels:

```
>>> t = sap.MaxTree(image)
>>> area = t.get_attribute('area')
>>> filtered_image = t.reconstruct(area < 100)
```

class sap.trees.**AlphaTree**(*image*, *adjacency=4*, *image_name=None*, *weight_function='L1'*)
Bases: *sap.trees.Tree*

Alpha tree, partition the image depending of the weight between pixels.

Parameters

- **image** (*ndarray*) – The image to be represented by the tree structure.
- **adjacency** (*int*) – The pixel connectivity to use during the tree creation. It determines the number of pixels to be taken into account in the neighborhood of each pixel. The allowed adjacency are 4 or 8. Default is 4.
- **image_name** (*str, optional*) – The name of the image Useful to track filtering process and display.
- **weight_function** (*str or higra.WeightFunction*) – The weight function to use during the construction of the tree. Can be L0, L1, L2, L2_squared, L_infinity, max, min, mean or a *higra.WeightFunction*. The default is L1.

class sap.trees.**MaxTree**(*image*, *adjacency=4*, *image_name=None*)
Bases: *sap.trees.Tree*

Max tree class, the local maxima values of the image are in leafs.

Parameters

- **image** (*ndarray*) – The image to be represented by the tree structure.

- **adjacency** (*int*) – The pixel connectivity to use during the tree creation. It determines the number of pixels to be taken into account in the neighborhood of each pixel. The allowed adjacency are 4 or 8. Default is 4.
- **image_name** (*str, optional*) – The name of the image Useful to track filtering process and display.

Notes

Inherits all methods of *Tree* class.

class sap.trees.**MinTree**(*image*, *adjacency=4*, *image_name=None*)
Bases: *sap.trees.Tree*

Min tree class, the local minima values of the image are in leafs.

Parameters

- **image** (*ndarray*) – The image to be represented by the tree structure.
- **adjacency** (*int*) – The pixel connectivity to use during the tree creation. It determines the number of pixels to be taken into account in the neighborhood of each pixel. The allowed adjacency are 4 or 8. Default is 4.
- **image_name** (*str, optional*) – The name of the image Useful to track filtering process and display.

Notes

Inherits all methods of *Tree* class.

class sap.trees.**OmegaTree**(*image*, *adjacency=4*, *image_name=None*)
Bases: *sap.trees.Tree*

Partition the image depending of the constrained weight between pixels.

Parameters

- **image** (*ndarray*) – The image to be represented by the tree structure.
- **adjacency** (*int*) – The pixel connectivity to use during the tree creation. It determines the number of pixels to be taken into account in the neighborhood of each pixel. The allowed adjacency are 4 or 8. Default is 4.
- **image_name** (*str, optional*) – The name of the image Useful to track filtering process and display.

class sap.trees.**TosTree**(*image*, *adjacency=4*, *image_name=None*)
Bases: *sap.trees.Tree*

Tree of shapes, the local maxima values of the image are in leafs.

Parameters

- **image** (*ndarray*) – The image to be represented by the tree structure.
- **adjacency** (*int*) – The pixel connectivity to use during the tree creation. It determines the number of pixels to be taken into account in the neighborhood of each pixel. The allowed adjacency are 4 or 8. Default is 4.
- **image_name** (*str, optional*) – The name of the image Useful to track filtering process and display.

Notes

Inherits all the methods of *Tree* class.

Todo:

- take into account adjacency

class sap.trees.**Tree**(*image*, *adjacency*, *image_name=None*, *operation_name='non def'*)
Bases: object

Abstract class for tree representations of images.

Notes

You should not instantiate class *Tree* directly, use *MaxTree* or *MinTree* instead.

available_attributes()
Return a dictionary of available attributes and parameters.

Returns **dict_of_attributes** – The names of available attributes and parameters required. The names are keys (str) and the parameters are values (list of str) of the dictionary.

Return type dict

See also:

get_attribute() Return the attribute values of the tree nodes.

Notes

The list of available attributes is generated dynamically. It is dependent of higras installed version. For more details, please refer to higra documentation according to the appropriate higras version.

Example

```
>>> sap.Tree.available_attributes()
{'area': ['vertex_area=None', 'leaf_graph=None'],
 'compactness': ['area=None', 'contour_length=None', ...],
 ...
 'volume': ['altitudes', 'area=None']}
```

get_attribute(*attribute_name*, ***kwargs*)
Get attribute values of the tree nodes.

Parameters **attribute_name** (*str*) – Name of the attribute (e.g. area, compactness,)

Returns **attribute_values** – The values of attribute for each nodes.

Return type ndarray

See also:

available_attributes() Return the list of available attributes.

Notes

Some attributes require additional parameters. Please refer to *available_attributes*. If not stated, some additional parameters are automatically deducted. These deducted parameters are altitudes and vertex_weights.

The available attributes depends of higras installed version. For further details Please refer to higra documentation according to the appropriate higras version.

Examples

```
>>> image = np.arange(20 * 50).reshape(20, 50)
>>> t = sap.MaxTree(image)
>>> t.get_attribute('area')
array([   1.,    1.,    1., ...,  998.,  999., 1000.])
```

get_params()

num_nodes()
Return the node count of the tree.

Returns **nodes_count** – The node count of the tree.

Return type int

reconstruct(*deleted_nodes=None*, *feature='altitude'*, *filtering='direct'*)
Return the reconstructed image according to deleted nodes.

Parameters

- **deleted_nodes** (*ndarray or boolean, optional*) – Boolean array of nodes to delete. The length of the array should be of same of node count.
- **feature** (*str, optional*) – The feature to be reconstructed. Can be any attribute of the tree (see *available_attributes()*). The default is *altitude*, the grey level of the node.
- **filtering** (*str, optional*) – The filtering rule to use. It can be direct, min, max or subtractive. Default is direct.

Returns **filtered_image** – The reconstructed image.

Return type ndarray

Examples

```
>>> image = np.arange(5 * 5).reshape(5, 5)
>>> mt = sap.MaxTree(image)
```

```
>>> mt.reconstruct()
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

```
>>> area = mt.get_attribute('area')
```

```
>>> mt.reconstruct(area > 10)
array([[ 0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

sap.trees.**available_attributes**()
Return a dictionary of available attributes and parameters.

Returns **dict_of_attributes** – The names of available attributes and parameters required. The names are keys (str) and the parameters are values (list of str) of the dictionary.

Return type dict

See also:

get_attribute() Return the attribute values of the tree nodes.

Notes

The list of available attributes is generated dynamically. It is dependent of higras installed version. For more details, please refer to higra documentation according to the appropriate higras version.

Example

```
>>> sap.available_attributes()
{'area': ['vertex_area=None', 'leaf_graph=None'],
 'compactness': ['area=None', 'contour_length=None', ...],
 ...
 'volume': ['altitudes', 'area=None']}
```

sap.trees.**load**(*file*)
Load a tree from a Higra tree file.

Parameters **file** (*str or pathlib.Path*) – File to which the tree is loaded.

Examples

```
>>> mt = sap.MaxTree(np.arange(10000).reshape(100,100))
>>> sap.save('tree.npz', mt)
```

```
>>> sap.load('tree.npz')
MaxTree{num_nodes: 20000, image.shape: (100, 100), image.dtype: int64}
```

sap.trees.**save**(*file*, *tree*)
Save a tree to a NumPy archive file.

Parameters

- **file** (*str or pathlib.Path*) – File to which the tree is saved.
- **tree** (Tree) – Tree to be saved.

Examples

```
>>> mt = sap.MaxTree(np.random.random((100,100)))
>>> sap.save('tree.npz', mt)
```

2.2.4 Utils

Various utilities unrelated to trees or profiles.

sap.utils.**local_patch**(*arr*, *patch_size=7*)
Create local patches around each value of the array

Parameters

- **arr** (*ndarray*) – The input data.
- **patch_size** (*int*) – The size w of the patches. For a 2D nadarray the returned patch size will be $w \times w$.

Returns patches – The local patches. The shape of the returned array is `arr.shape + (patch_size,) * arr.ndim`.

Return type ndarray

Notes

This implementation is memory efficient. The returned patches are a view of original array and are not writeable.

This function works regardless of the dimension of `arr` with hypercubes shaped patches, according to the dimension of `arr`.

See also:

local_patch_f() use a function over the local patches.

sap.utils.**local_patch_f**(*arr*, *patch_size=7*, *f=np.mean*)
Describe local patches around each value of the array

Parameters

- **arr** (*ndarray*) – The input data.
- **patch_size** (*int*) – The size w of the patches.
- **f** (*function*) – The function to run over the local patches. For now it is necessary to use a function with `axis` parameter such as `np.mean`, `np.std`, etc See more functions on Numpy documentation.

Returns patches – The description of the local patches. The shape of the returned array is `arr.shape`.

Return type ndarray

Notes

Refer to *local_patch()* for full documentation.

See also:

local_patch() create the local patches.

sap.utils.**ndarray_hash**(*x*, *l=8*, *c=1000*)
Compute a hash from a numpy array.

Parameters

- **x** (*ndarray*) – The array to hash.
- **l** (*int, optional*) – The length of the hash. Must be an even number.
- **c** (*int, optional*) – A variable to affect the sampling of the hash. It has to be the same along the matching process. Refer to notes.

Returns **hash** – The hash of array x.

Return type str

Notes

Python hash is slow and will offset the random generator in each kernel. The hash of the same data will not match in different kernels.

The idea is to sparsely sample the data to speed up the hash computation. By fixing the number of samples the hash computation will take a fixed amount of time, no matter the size of the data.

This hash function output a hash of x in hexadecimal. The length of the hash is l. The hashes are consistent when tuning the length l: shorter hashes are contained in the longer ones for the same data x. The samples count taken in x is $\frac{l \times c}{2}$.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ RENNES 2

Titre : Discrétisation 2D/3D de nuages de points LiDAR : traitement à l'aide des hiérarchies morphologiques et des réseaux de neurones profonds

Mots-clés : LiDAR, morphologie mathématique, apprentissage profond

Résumé : Cette thèse évalue le potentiel des méthodes hiérarchiques d'analyses morphologiques et des réseaux de neurones profonds pour analyser les données LiDAR au moyen de plusieurs stratégies de discrétisation. La quantité de données LiDAR augmente de manière exponentielle en termes de superficies couvertes et de résolution spatiale. Cependant, ces données ne sont pas encore pleinement exploitées en raison du manque d'outils méthodologiques efficaces. Les approches morphologiques sont connues pour extraire des caractéristiques multi-échelles fiables tout en étant extrêmement efficaces sur le plan calculatoire. Dans le même temps, la formidable percée de l'apprentissage profond en vision par ordinateur a bouleversé la communauté de la télédétection. Afin d'évaluer ces outils, nous définissons et évaluons différentes stratégies de discrétisation des données. Dans une première partie, nous réorganisons les nuages de points LiDAR en grilles régulières 2D. Nous proposons de dériver plusieurs caractéristiques, en extrayant, en plus de l'altitude, des informations spécifiques au LiDAR (valeurs spectrales, nombre de retours, etc). Dans une deuxième partie, nous réorganisons les nuages de points en grilles régulières 3D. Cela permet de fournir le contexte de voisinage nécessaire aux approches morphologiques hiérarchiques et les grilles proposées sont aussi adaptées aux couches d'entrée des réseaux de neurones profonds. Les différents développements proposés ont systématiquement fait l'objet d'une validation en télédétection.

Title: 2D/3D discretization of LiDAR point clouds: Processing with morphological hierarchies and deep neural networks

Keywords: LiDAR, mathematical morphology, machine learning, deep learning

Abstract: This thesis evaluates the relevance of morphological hierarchies and deep neural networks for analysing LiDAR data by means of several discretization strategies. The quantity of data increases exponentially in coverage and resolution. However, actual datasets are not yet fully exploited due to the lack of efficient methodological tools for this specific type of data. Morphological structures are known to extract reliable multi-scale features while being extremely computationally efficient. In the mean time, the tremendous breakthrough of deep learning in computer vision has shaken up the remote sensing community. To this end we define and evaluate different discretization strategies of LiDAR data. In a first part, we re-organise the point clouds into 2D regular grids. We propose to derive several LiDAR features, trying to extract complete elevation description and spectral values along with LiDAR specific information. In a second part we re-organise the point clouds into 3D regular grids. The regular grids are sufficient to provide the neighboring context needed for the morphological hierarchies, and the proposed grids are also adapted to the input layers of state-of-the-art deep neural networks. The different methods are systematically validated in remote sensing scenarios.

Imp. Service Reprographie - Rennes 2