



Dissertation

Contextual Semantic Classification of ALS Point Clouds in Urban Environment

Ausgeführt zum Zwecke der Erlangung des akademischen Grades eines
Doktor der technischen Wissenschaften (Dr.techn.)

Unter der Leitung von
Univ.-Prof. Dipl.-Ing. Dr.techn. Norbert Pfeifer

E 120.7

Department für Geodäsie und Geoinformation
Forschungsgruppe Photogrammetrie

Eingereicht an der Technischen Universität Wien TU
Fakultät für Mathematik und Geoinformation

von

Nan Li

Matrikelnummer: 01529924

Wien, am 21. July 2021

.....



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.



Dissertation

Contextual Semantic Classification of ALS Point Clouds in Urban Environment

A thesis submitted in fulfilment of the academic degree of
Doktor der technischen Wissenschaften (Dr.techn.)

Under the supervision of
Univ.-Prof. Dipl.-Ing. Dr.techn. Norbert Pfeifer

E 120.7

Department of Geodesy and Geoinformation
Research Group Photogrammetry

Research conducted at TU Wien
Faculty of Mathematics and Geoinformation

by

Nan Li

Matriculation number: 01529924

Vienna, July 21, 2021

.....

Supervisor: Prof. Dr. Norbert Pfeifer
Technische Universität Wien
Department of Geodesy and Geoinformation
Research Group Photogrammetry
Wiedner Hauptstraße 8/E120, 1040 Vienna, Austria
E-Mail: Norbert.Pfeifer@geo.tuwien.ac.at

Referee: Prof. Dr. Ir. George Vosselman
University of Twente
Faculty of Geo-Information Science and Earth Observation
Department of Earth Observation Science
Hengelosestraat 99, 7514 AE Enschede, Netherlands
E-Mail: george.vosselman@utwente.nl

Referee: Apl. Prof. Dr.-Ing. Norbert Haala
University of Stuttgart
Institute for Photogrammetry
Research Group Photogrammetric Computer Vision
Geschwister-Scholl-Straße 24D, 70174 Stuttgart, Germany
E-Mail: norbert.haala@ifp.uni-stuttgart.de

Erklärung zur Verfassung der Arbeit

Author' Statement

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit- einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

I hereby declare, that I independently drafted this manuscript, that all sources and references used are correctly cited and that the respective parts of this manuscript including tables, maps and figures – which were included from other manuscripts or the internet, either semantically or syntactically, are made clearly evident in the text and all respective sources are correctly cited.

Vienna, July 21, 2021

Nan Li



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Acknowledgments

Foremost, I owe a debt of gratitude to my supervisor, Norbert Pfeifer, who has supported me in all aspects. He has always had the patience to explain, discuss and listen, and most of all, he can always answer the most basic question without making me feel foolish for asking. Moreover, he is a generous giver of energy, especially when I need a pep talk at times during this daunting journey. He can always calmly find a solution whenever I encounter a problem and be very tolerant when I put us in a last-minute deadline rush (at several times, late at night!). All in all, thanks for providing continuous encouragement and guiding my way through this doctoral study.

I have been very fortunate with the colleagues in the research group of Photogrammetry as well as Remote Sensing. Thank all of you providing such a friendly and enjoyable atmosphere to work in. I thoroughly enjoyed our small talk time in the kitchen or on the corridor, as well as all the academic discussion occurred in our photo circle and weekly meeting. Thanks for so many great talks and help both academic and in other areas of life. A special thanks must go to Karel Wilfried for improving the language quality of the German abstract and making it more reading friendly to native German speaker. Thank you, all the IT staff, for facilitating the setup of the hardware, which allows me to start the deep learning experiments in a short time. Thank you, all the secretary staff, for the kind assist in various paperwork. I would like to thank the Active Vision group of Siemens, for funding the project of deep learning classification and helping me gain new perspectives.

I thought I was a hardworking person until I met my boyfriend, Yi Hung, who is always heavily occupied with studying for endless exams. Thanks for sharing working weekends together with me, and making it less difficult as it sounds.

Finally, my deepest thanks to my parents for your years of love, support and encouragement. They selflessly encouraged me to explore new possibilities in life and seek my own destiny. Thanks for always standing by my side.

It has been a long journey, but I'm lucky to have all of you around. Thanks being part of this journey.

Nan Li



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Abstract

ALS (Airborne Laser Scanning)/Airborne LiDAR (Light Detection And Ranging) is characterized by its capability of providing highly accurate and dense 3D point clouds at a city-wide scale. Therefore, ALS has been widely used for a variety of applications in urban environments, which makes the automatic classification of 3D point clouds a crucial task. Urban environments are a complex combination of both built-up and natural objects. Consequently, automatically producing highly accurate classification results from ALS point clouds is challenging. The commonly used machine learning methods such as Random Forest and Support Vector Machine often lead to noisy classification results, since they rely heavily on the input of handcrafted features and lack the consideration of spatial context. In order to improve classification performance, the aim of this dissertation is to incorporate context into the classification of ALS point clouds. The work in this dissertation comprises methodological developments presented in publications I-III, an investigation of the cutting-edge deep learning methods, in which contextual features can be directly learned in the training phase, presented in publication IV, and an effort on extending training data that is presented in publication V.

Publications I-II propose a high-dimensional tensor-based sparse representation for the classification of ALS point clouds. This novel data structure is introduced to keep the handcrafted features in their original geometric 3D space, such that the spatial distribution and handcrafted features can be considered at the same time. Using only a few training samples, promising classification results can be obtained. Publication III develops a label smoothing strategy to refine classification results. Without the aid of additional training data, the proposed label smoothing strategy can directly learn context information from initial classification results by estimating an adaptive neighborhood and a probabilistic label relaxation. Experiments exhibit its strength in improving classification accuracy. Publication IV conducts a comprehensive comparison between three state-of-art deep learning models, namely PointNet++, KPConv, and SparseCNN, w.r.t. classification accuracy, computation time, generalization ability, and the sensitivity to the choices of hyper-parameters. Publication V proposes a method to extend training data by selecting the most informative samples from the neighborhood of the initial training samples.

After an overview of the publications, a comparison between all investigated methods is carried out on two separate ALS datasets in urban areas. Based on the results, the superior performance of the selected deep learning models, especially SparseCNN, is further confirmed. The proposed label smoothing also turns out to be advantageous, regardless of the diversity of scenes and point densities involved. More importantly, it is independent of different training efforts, which have a profound impact on deep learning methods. The works conducted in this dissertation provide practical examples and valuable insights into the contextual classification of ALS point clouds. The presented studies also show that supervised machine learning, especially deep learning, heavily depends on training data. Therefore, the automated generation of training data by active learning could be a way to further facilitate the classification of point clouds.

Kurzfassung

Luftgestütztes Laserscanning (Airborne Laserscanning, ALS) / LiDAR (Light Detection And Ranging) zeichnet sich durch seine Fähigkeit aus, hochpräzise und dichte 3D-Punktwolken großflächig zu erfassen. Daher wird ALS schon lange auch im Bereich städtischer Anwendungen eingesetzt. Diese bedingen aber die automatische Klassifizierung von 3D-Punktwolken, und die städtische Umgebung ist eine komplexe Kombination aus bebauten und natürlichen Objekten. Deshalb ist die automatische Erstellung hochgenauer Klassifizierungsergebnisse aus ALS-Punktwolken eine wichtige und anspruchsvolle Aufgabe. Die üblicherweise verwendeten Methoden des Maschinellen Lernens wie Random Forest und Support Vector Machine führen oft zu verrauschten Klassifizierungsergebnissen, da sie stark von „handgestrickten“ Merkmalen abhängen und keine Rücksicht auf den räumlichen Kontext nehmen. Um die Klassifizierungsleistung zu verbessern, ist es das Ziel dieser Dissertation, den Kontext in die Klassifizierung von ALS-Punktwolken einzubeziehen. Die Arbeit in dieser Dissertation umfasst Entwicklungen der Methodik, die in den Veröffentlichungen I-III vorgestellt werden. Weiters die in der Veröffentlichung IV vorgestellte Untersuchung der hochmodernen Deep-Learning-Methoden, bei denen die kontextbezogenen Merkmale direkt während des Trainings gelernt werden können, und einen Versuch zur Erweiterung der Trainingsdaten, der in der Veröffentlichung V vorgestellt wird.

In den Veröffentlichungen I-II wird eine hochdimensionale, tensorbasierte Sparse-Darstellung zur Klassifizierung von ALS-Punktwolken vorgeschlagen. Diese neuartige Datenstruktur wird eingeführt, um die manuell festgelegten Merkmale in ihrem ursprünglichen geometrischen 3D-Raum zu halten, sodass die räumliche Verteilung und die handgestrickten Merkmale gleichzeitig berücksichtigt werden können. Schon mit kleinen Trainingsstichproben können so vielversprechende Klassifizierungsergebnisse erzielt werden. In der Veröffentlichung III wird eine Strategie zur Glättung von Klassenzuordnungen entwickelt, um die ursprünglichen Klassifizierungsergebnisse zu verfeinern. Ohne Hilfe zusätzlicher Trainingsdaten kann die vorgeschlagene Strategie direkt die Kontextinformationen aus den ursprünglichen Ergebnissen durch die Schätzung einer adaptiven Nachbarschaft und probabilistischer Klassenzuordnungs-Relaxation erlernen. Experimente belegen die Stärke dieser Strategie bei der Verbesserung der Klassifizierungsgenauigkeit. In der Veröffentlichung IV wird ein umfassender Vergleich zwischen den drei aktuellen Deep-Learning-Modellen durchgeführt, nämlich PointNet++, KPConv und SparseCNN, in Bezug auf Klassifizierungsgenauigkeit, Rechenzeit, Generalisierungsfähigkeit sowie die Sensitivität in Bezug auf die Wahl der Hyper-Parameter. In der Veröffentlichung V wird schließlich eine Methode zur Erweiterung der Trainingsdaten vorgeschlagen, indem die informativsten Stichproben aus der Nachbarschaft der ursprünglichen Trainingsstichproben ausgewählt werden.

Nach einem Überblick über die Veröffentlichungen werden alle untersuchten Methoden an Hand zweier ALS-Datensätze in städtischen Gebieten verglichen. Die Ergebnisse bestätigen erneut die überlegene Leistung der ausgewählten Deep-Learning-Modelle, insbesondere von SparseCNN. Die Vorteile der vorgeschlagenen Strategie zur Glättungsmethod von Klassenzuordnungen werden ebenfalls nachgewiesen, unabhängig von der vorhandenen Vielfalt an Szenen und Punktdichten. Noch wichtiger ist, dass sie unabhängig vom Trainingsaufwand ist, der ja einen tiefgreifenden Einfluss auf Deep-Learning-Methoden hat. Die in dieser Dissertation durchgeführten Arbeiten liefern praktische Beispiele und wertvolle Einsichten in die kontextuelle

Klassifizierung von ALS-Punktwolken. Die vorgestellten Studien zeigen auch, dass das überwachte maschinelle Lernen, insbesondere Deep-Learning-Methoden, stark von Trainingsdaten abhängig ist. Daher könnte die automatische Erstellung von Trainingsdaten durch aktives Lernen ein zukünftiger Weg sein, um die Klassifizierung von Punktwolken weiter zu verbessern.



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Table of Contents

1, Introduction.....	1
1.1 Motivation	1
1.2 State-of-the-art	1
1.2.1 Methods for classification of ALS point clouds.....	1
1.2.2 Contextual classification.....	2
1.2.3 Application of ALS in the urban environment.....	6
1.3 Objectives.....	7
1.4 List of Publications	7
1.4.1 Summaries of the Publications	7
1.4.2 Author contributions.....	11
2, Publications	12
Publication I	12
Publication II.....	22
Publication III.....	43
Publication IV.....	63
Publication V	84
3, Classification performance comparison of the investigated methods.....	90
3.1 Test data.....	90
3.2 Setup.....	90
3.3 Results	91
3.3.1 Vienna dataset.....	91
3.3.2 Vaihingen dataset	93
3.4 Discussion.....	96
3.4.1 TSRC	96
3.4.2 Label Smoothing	96
3.4.3 PointNet++, KPConv and SparseCNN	96
4, Conclusions	98
Bibliography.....	99



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

1, Introduction

1.1 Motivation

Urban environment consists of various artificial and natural objects including roads, buildings, trees, vehicles, street lamps, power lines, grass, etc. The geo-information on these objects such as spatial distribution and geometric characteristics plays an essential role in guiding public policy and resource management for authorities.

In recent years, many studies have demonstrated that Airborne LiDAR is an effective and efficient tool to conduct topographic surveys. Airborne LiDAR can collect a large amount of 3D point clouds in a short time by an active distance measurement. In contrast to 2D remote sensing imagery, LiDAR point clouds can accurately describe topographic surface in both horizontal and vertical directions without effects of relief displacement, with penetration of tree canopy and insensitivity to lighting condition. This makes airborne LiDAR an important data source for various applications in urban environment, and normally classification of point clouds is a fundamental step to extract meaningful information for these applications.

In the community of remote sensing and photogrammetry, tremendous efforts have been made for automated classification of LiDAR point clouds. Researchers have computed effective geometric features from LiDAR point clouds to distinguish different objects by their geometric characteristic within a local neighborhood. Regarding classifiers, sophisticated machine learning algorithms such as Random Forest and Support Vector Machine have been successfully applied to the classification of LiDAR points. The current progress has already allowed a relative high degree of automation in point cloud classification. However, due to the complexity of urban scenes, it remains difficult in achieving highly accurate classification results for LiDAR points. The commonly used machine learning methods lack the consideration of context information, and thus often lead to inhomogeneous results. Although some studies employed graphical models such as MRF (Markov Random Field) and CRF (Conditional Random Field) to introduce context in classification rules, the performance still relies heavily on the representation ability of the handcrafted features, as well as the way in which adjacency graphs construct. Thus, it is still worth developing new algorithms to make the best use of context information, in order to achieve highly accurate classification results in urban areas.

As deep learning has shown superior performance to other machine learning methods in many fields, classification of LiDAR points also tends to shift in focus to deep learning. Numerous deep learning architectures have been developed and achieved satisfactory results in the classification of ALS point clouds. However, only little attention has been paid to the comparison of different deep learning networks. To provide better insights of deep learning techniques to the community of photogrammetry, a comprehensive comparison of classification performance by different deep learning networks requires to be investigated.

1.2 State-of-the-art

1.2.1 Methods for classification of ALS point clouds

The typical approach for classification of point clouds involves three main steps: local neighborhood recovery, geometric features estimation and classification. The commonly used neighborhood definitions of 3D points are represented by spherical neighborhoods (Lee and Schenk, 2002), cylindrical neighborhoods (Filin and Pfeifer, 2005) and a fixed number of nearest points (Linsen and Prautzsch, 2001). The constant scale parameters (e.g. searching radius or number of nearest neighbors) are often given with respect to prior knowledge of the point clouds

in the scene and may vary with the density and geometric structures. To overcome these restrictions, many studies proposed data-driven adaptive neighborhood. For instance, Weinmann et al. (2015) selected the optimal number of nearest points by finding the minimal entropy of eigenvalues. Demantke et al. (2011) determined the sphere neighborhood size by selecting dominated features related to dimensionality (e.g. linear, planar and volumetric) at various sizes.

Eigen-features derived from a covariance matrix of points within a local neighborhood are the most commonly used features for classification of ALS point clouds. The derived eigenvalues can be used to compute a range of local 3D shape features that intuitively describe local geometric characteristics, such as linearity, planarity and scattering. Such 3D geometric features are frequently combined with height-based features, echo-based features and density-based features in a classification task. Additionally, features resulted from a 2D neighborhood are considered able to provide complementary information for aforementioned features estimated from 3D neighborhood. For instance, height difference and density derived from 2D neighborhood are particular useful to detect objects with vertical structures.

A variety of machine learning methods have been applied for automated classification of 3D point clouds, such as RF (Random Forests) (Guo, 2011), SVM (Support Vector Machines) (Mallet, 2011) and Bayesian Discriminant Classifiers (Khoshelham, 2013). These classifiers take feature vectors as input, which are concatenated by extracted features and are used to represent points. Accordingly, sufficient representative points with reference labels are required to train the involved classifier so that it is afterwards capable to assign class labels to unseen point clouds. Since many machine learning models are publicly available in various software tools, it's easy to implement them on 3D point clouds. However, many respective pointwise classification replies only on the feature vectors and lack the consideration of spatial correlations of neighboring 3D points, thus they tend to deliver noisy classification results.

1.2.2 Contextual classification

Context can be defined as the information or characteristics derived from the surrounding of an overserved point, which has been regarded as an important additional information source for classification of Earth observation data such as remote sensing images and LiDAR point clouds, since objects in the real world are inherently correlated with each other. With the aid of tremendous algorithms developed from the field of computer vision, as well as the regular grids that facilitate implementations of those algorithms, many advanced methods related to context were firstly introduced in the studies of remote sensing images classification. Then some of these methods were adapted to LiDAR point clouds. Thus, at the beginning of this section, a brief review of contextual classification of remote sensing images is given, followed by a comprehensive review of contextual classification of LiDAR point clouds.

Contextual classification of remote sensing images

Remote sensing sensors are able to record each image pixel at various wavelengths of the electromagnetic spectrum. The spectral characteristics of remote sensing images can provide rich input features for most pixel-wise supervised classification algorithms. However with the increasing availability of high spatial resolution images, pixel-wise classification that only considers spectral features often lead to noisy results and fail to model the complicated geospatial structures. Thus, many researchers have proposed to exploit spatial features as a complementary source of information to the spectral features. Note that the term context information normally refers to spatial information for remote sensing images.

A straightforward method is to use image filters, which is implemented by using the weighted sum of the adjacent pixels in successive windows. By applying well-designed filters, certain texture

features or explicit spatial characteristics such as size and shape can be computed from remote sensing images. The commonly used filter-based methods include wavelet textures (Myint, 2004), gray-level co-occurrence matrix (Pacifici, 2009) and morphological profiles (Pesaresi and Benediktsson, 2001). Meanwhile, the object-based image analysis (OBIA), as well as superpixel-based classification that aims to avoid highly accurate results of object segmentation, are also widely used to incorporate spatial information (Zhou, 2009, Vargas, 2015). The basic idea of these methods is to group the spatially neighboring pixels into spectrally similar regions and then regard these objects as classification entities, so that the intra-class variability of spectral features between neighbors can be reduced.

Context information can also be integrated into classification rules with the assumption that neighboring pixels favor correlated class labels. A general framework to model context information among pixels is to apply graphical models such as MRF (Li, 2011) and CRF (Zhao, 2017). Additionally, the regular grid-like structures of images offer the possibility to consider the specific spatial arrangement of class labels as contextual constraints to improve classification performance. For instance, the probabilities of class labels can be refined through an iterative relaxation processing with a matrix containing compatibility coefficients of class labels (Rodríguez-Cuenca, 2012), this procedure is often referred to probabilistic label relaxation.

More recently, Convolutional Neural Network (CNN) has become the most commonly used model for remote sensing images (Ma, 2019, Parikh, 2020), as the convolutional kernels are well-suited to the intrinsic 2D structure of images. The existing sophisticated architectures proposed in the field of computer vision such as VGG (Simonyan and Zisserman, 2014), ResNet (He, 2016), DenseNet (Huang, 2017), have greatly facilitated constructing a task-specific network for remote sensing images. Additional efforts have been made to improve recognition accuracy of fine-structured objects by designing no-downsampling encoder (Sherrah, 2016), multi-scale aggregation (Liu, 2018, Zhang, 2020) and combining features that are learned at multiple resolutions (Maggiori, 2017). Compared with benchmark datasets used in computer vision (e.g., ImageNet), the volume of available remote sensing datasets with reference labels is quite limited. Thus, many studies have focused on transfer learning to enhance deep learning models with limited training samples. As ImageNet consists of huge amount of daily natural images that have the similarity with aerial images, several studies performed fine-tuning on target remote sensing datasets based on the network pre-trained on ImageNet (Marmanis, 2016, Wang, 2017). Moreover, to collect semantic labels of remote sensing images with a low cost, some studies embed the semi-supervised techniques into deep learning networks to obtain an enlarged annotation dataset from unlabeled samples (Han, 2018, Hong, 2020).

Contextual classification of LiDAR point clouds

In contrast to remote sensing images, the geometric features that are commonly used in LiDAR point clouds already include context at a basic and local scale, since these features are computed by a group of neighboring points. As we indicated in previous section, conventional machine learning methods such as SVM and RF treat the feature vector of each point independently, which often leads to an inhomogeneous classification result due to the lack of spatial correlations. Therefore, the context information of LiDAR point clouds needs to be further strengthened. The current contextual classification of LiDAR point clouds can be divided into two categories: contextual classification based on handcrafted features and deep learning methods.

Contextual classification based on handcrafted features

It's implicit to describe the context between neighboring points in 3D point clouds. Thus, handcrafted contextual features are usually extracted at the object level rather than at the point level based on task-specific prior knowledge. For instance, Yang et al. (Yang, 2017) considered

contextual features as the relative positions, the direction relations and the spatial distribution pattern of road facilities. Apparently, the rule-based contextual features are hard to generalize to other classification scenes. More commonly, the combination of context and geometric handcrafted features is accomplished by segmentation. By grouping spatially neighboring and geometrically similar points into homogeneous segments, context of individual points are naturally considered. The geometric features of segments can be computed directly base on the points within the segments or by taking the average of the features of points. Consequently, the geometric features extracted from segments are regarded more representative than point-wise geometric features (Vosselman, 2017), and additional features such as area and size can be extracted from segments to aid classification (Zhang, 2013). Nevertheless, the accuracy classification relies on a reliable segmentation of point clouds.

Like remote sensing images, applying graphical models, such as MRF and CRF is a more general framework for incorporating contextual information in classification. In the community of photogrammetry, most efforts have been made on two aspects: modelling pairwise potentials and constructing adjacency graphs. A simple way to build the pairwise potential is to employ the Potts model, which favors identical labels in a neighborhood. A penalty term related to label changes can be further added into the Potts model, and the penalty is usually measured by the Euclidean distance between the features of adjacent nodes (Weinmann, 2015, Wei, 2019). To reduce over-smoothing, pairwise potentials can also be given by discriminative classifiers, such as linear model and Random Forest, as they can deliver the joint probability of class labels of neighboring nodes based on the observed interaction features (Niemeyer, 2014).

Since the adjacency graph determines the extent to which interactions in a local neighborhood are considered, the topology of the graph is critical to the classification results. Instead of widely used k nearest neighbors within a sphere, Niemeyer et al. (Niemeyer, 2014) employed k nearest neighbors in a 2D cylindrical neighborhood, because points with different heights, such as canopy and ground points, are expected to give valuable hints for local configuration of classes. Some studies (Weinmann, 2015, Landrieu, 2017) used the adaptive neighborhood to select an appropriate neighborhood size in case of varying density. However, the contextual information in such a local neighborhood is still too restrictive. As a result, wrong predictions will occur at isolated groups of points. To incorporate long-range interactions, two types of strategies are developed: constructing adjacency graph by using multi-scale neighborhood (Luo and Sohn, 2014) and implementing graphical models on segments (Vosselman, 2017).

Label smoothing is usually regarded as an alternative form of incorporating context in the post-processing of classification. The commonly used methods for LiDAR points are majority voting or other advanced filters, such as Gaussian, bilateral and edge-aware filters that take account of features and probabilistic outputs of classifiers and impose inversed weights on neighboring points. Besides, label smoothing can be considered from a structured regularization perspective by using MRF or CRF models (Landrieu, 2017).

One major limit of aforementioned methods is that well-designed handcrafted features need to be fed into models, and the overall accuracy still heavily relies on the effectiveness of handcrafted features. Additional challenges include that context considered in the graphical model is determined by the neighborhood definition, and segment-based methods tend to misclassify small and thin objects. As a result, more and more studies shift their interest to the end-to-end classification implemented by deep learning algorithms, in which handcrafted features are no longer required.

Deep learning classification

In deep learning methods, task-specific contextual features can be directly learned from raw point clouds by their multilayer non-linear structures. Moreover, successively increasing the receptive field size in the later layers enables that context information can be included at a very large scale.

Compared to the CNN-based (Convolutional Neural Network) classification of remote sensing images, studies of deep learning on 3D point clouds, especially on ALS point clouds, have started a few years later as CNN is difficult to be implemented directly on irregular 3D point clouds. Numerous efforts have been made in the community of computer vision regarding indoor point clouds. PointNet is a pioneering network that applies deep learning to an unordered point cloud by point-wise encoding and decoding (Qi, 2017). The key idea is to learn features through a sequence of MLPs (Multilayer Perceptron) on a group of neighboring points of each individual point, then aggregate features by pooling using the maximum or average to achieve order invariant. The authors further proposed PointNet++ that applies MLP recursively on a nested partitioning of point clouds, such that both local and global context can be considered (Qi, 2017). Then, based on the architecture of PointNet++, some extension have been proposed to better explore the interactions between neighboring points, such as PointSIFT (Jiang, 2018), PointWeb (Zhao, 2019) and DGCNN (Wang, 2019).

However, input points are flattened in MLPs, which leads in losing inherent spatial distribution of neighboring points. Therefore, many studies have considered applying convolutional kernels on 3D points so that the spatial arrangement of points can be better exploited. The most straightforward way is to first convert 3D point clouds into voxels and implement 3D convolution operations on 3D grids (Maturana and Scherer, 2015). But this is extremely demanding in terms of computation and memory usage for large-scale point clouds with fine voxel resolution. SparseCNN solved the inefficiency in a way that it only took non-empty voxels as input, and also restricted the output of the convolution to the set of non-empty voxels (Graham, 2018). Apart from voxels, various sparse data structures are also employed to apply CNNs to 3D point clouds, such as Octrees used in OctNet (Riegler, 2017) and Kd-trees used in KD-Net (Klokov and Lempitsky, 2017).

In order to keep the resolution of the original point clouds, several strategies have been developed to perform convolution operations directly on unordered 3D point clouds. PointCNN proposed to use a transformation matrix that is able to simultaneously weight and permute the input feature of local points (Li, 2018). In the works of PointConv (Wu, 2019), SpiderCNN (Xu, 2018) and Flex-convolution (Groh, 2018), the convolution kernel are regarded as a parametric function of local 3D coordinates. In the convolution of images, a kernel, which is represented as a matrix, is slid across the image and multiplied with the inputs to extract features. To perform convolution for 3D point clouds in a similar manner, some studies constructed the kernel by a set of 3D spatial points that define the area where each kernel weight is applied, such as PCNN (Atzmon, 2018), ConvPoint (Boulch, 2020) and KPConv (Thomas, 2019).

In the community of remote sensing and photogrammetry where outdoor point clouds are typically considered, not only have the existing deep learning architectures (e.g. PointNet++ and SparseCNN) been successfully adapted to ALS point clouds (Schmohl and Sörgel, 2019, Winiwarter, 2019), but also extensions and new architectures have been developed to improve classification performance on ALS point clouds. For instance, Li et al. (Li, 2020) proposed a geometry-aware convolution to learning high-level features from the low-level handcrafted features, so that the geometric characteristics can be emphasized. Wen (Wen, 2020) constructed the local receptive field by only selecting the directionally constrained nearest neighbors. Li et al. (Li, 2020) introduced an inverse density function to deal with varying points density. A more

comprehensive literature review of deep learning methods for 3D point clouds can be found in the Publication IV.

1.2.3 Application of ALS in urban environment

ALS technology provides not only the extraordinary ability in collecting highly accurate surface elevation measurements over urban areas, but also the capability of penetrating through volume-scattering objects. With this uniqueness over other remote sensing data, ALS has emerged as a powerful technology for a wide range of applications in urban environment. In this section, applications are categorized by the dominated objects in urban areas, such as buildings, trees and power lines.

Building-based applications

As the most pronounced elements of urban areas, buildings information have been involved in a variety of activities related to urban planning and resource management. Consequently, buildings recognition, extraction and reconstruction have become the hottest topic in ALS point clouds processing. Additionally, the 3D point clouds captured by ALS permit accurately estimating geometric properties of buildings. More specifically, geometric attributes such as aspect, inclination angle and area of each roof plane that are calculated from ALS point clouds, can be used for solar potential assessment (Jochem, 2009) and mapping demand of building thermal energy (Tooke, 2014). Moreover, the large coverage of ALS allows measuring building density information at land lot and urban district scales (Yu, 2010).

Vegetation-based applications

Vegetation inventory can provide fundamental information for urban ecosystem and environmentally sustainable development. The penetration ability of laser rays and the high spatial resolution, especially the vertical accuracy of ALS, have greatly facilitate the measurement of urban trees. This can be confirmed in a group of studies of urban green volume estimation (Hecht, 2008), leaf area index (LAI) mapping (Alonzo, 2015) and tree condition evaluation (Plowright, 2016).

Power lines-based applications

Mapping the power lines is one of the powerful applications of ALS, because such task is labor-intensive and costly by traditional field-based inspection. An efficient and reliable monitoring of power lines and their surroundings are critical for the purpose of maintenance, potential threaten evaluation and upgrading. Therefore, a great many of studies of transmission wires extraction and pylon localization from ALS point clouds can be found in the literature (McLaughlin, 2006, Sohn, 2012). ALS can also be used as a complementary information source to area images for the management in power lines corridors (Mills, 2010).

Terrain-based applications

LiDAR have become the primary technique for producing high resolution DTM in recent years, since LiDAR provides an efficient way in acquiring accurate vertical information over a large area. The DTM generated from ALS have been used as base maps in many applications, which include solar potential analysis (Prieto, 2019), viewshed analysis (Yamagata, 2016), landslide inventory (Bernat Gazibara, 2019) and flood-risk mapping (Hung, 2018). The high resolution of the DTM allows these analyses to be performed at a finer scale.

There are many other applications that require semantic 3D geo-information, such as urban wind ventilation simulation (Peng, 2017) and urban traffic analysis (Yao, 2011).

1.3 Objectives

In general, the aim of this dissertation is to develop appropriate methods for contextual classification of ALS point clouds in urban environment. Since compared to remote sensing images, deep learning methods are less investigated for classification of LiDAR point clouds, another interest of this dissertation lies in the deep learning methods investigation in classification of ALS point clouds.

1.4 List of Publications

This dissertation comprises three peer-reviewed journal papers and two peer-reviewed conference papers, which include:

- **Publication I:** Li, N., Pfeifer, N., & Liu, C. (2017). Airborne LiDAR Points Classification Based on Tensor Sparse Representation. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences, Spatial Inf. Sci., IV-2/W4*, 107–114.
- **Publication II:** Li, N., Pfeifer, N., & Liu, C. (2017). Tensor-based sparse representation classification for Urban Airborne LiDAR points. *Remote Sensing*, 9(12), 1216.
- **Publication III:** Li, N., Liu, C., & Pfeifer, N. (2019). Improving LiDAR classification accuracy by contextual label smoothing in post-processing. *ISPRS Journal of Photogrammetry and Remote Sensing*, 148, 13-31.
- **Publication IV:** Li, N., & Pfeifer, N. (2021). A Comparison of Deep Learning Methods for Airborne Lidar Point Clouds Classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14, 6467-6468.
- **Publication V:** Li, N., & Pfeifer, N. (2019). Active learning to extend training data for large area airborne lidar classification. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Science. Spatial Inf. Sci., XLII-2/W13*, 1033–1037.

1.4.1 Summaries of the Publications

Publication I

This publication proposed a tensor-based sparse representation classification (TSRC) method for ALS point clouds classification in urban areas. The inspiration came from a number of studies regarded a hyperspectral image as a 3D tensor to jointly take advantages of the spatial and spectral information. Regarding sparse representation, it was firstly proposed in signal processing applications, and its assumption is that most natural signals can be compactly represented by only a few coefficients that carry the most important information in a certain basis or dictionary. In the past a few years, the applications of sparse representation have also been successfully extended to face recognition and image classification. The sparse representation of an unknown sample is expressed as a sparse vector whose nonzero entries correspond to the weights of the selected training samples from a structured dictionary, the class label of the test sample is determined by the characteristics of the sparse vector.

This publication introduced a new data structure that represents each point as a 4th-order tensor to simultaneously take the spatial distribution and handcrafted features of neighboring points into

account. The first three dimensions are generated by a voxel representation of surrounding points within a 3D neighborhood, the fourth dimension is the average handcrafted features of points within each voxel. Furthermore, this publication presented a sparse representation adapted to the high dimensional tensor data. Only a few of training samples are used to learn a structured and discriminative dictionary, a sparse tensor can be obtained by projecting a designed 4th-order tensor onto the learned dictionary. The points are classified based on the observation that, for points with same class labels, the non-zero entries in their sparse tensors usually lie in the same subset.

The developed approach is evaluated on two tiles of the ALS dataset of Vienna city, and promising classification results have been obtained with an overall accuracy of 82%. Given that only 6 training points are selected from each class to build the dictionary, thus we considered that it is an overall good classification performance.

This publication developed a novel contextual classification method which integrates local context from a perspective of feature projection. The classification results have demonstrated the potential of the sparse representation classification based on high dimensional tensors, especially in the case that only a few of training samples are available.

Publication II

This publication is an extension of the publication I. In this publication, a robust method for 4th-order tensor representation of 3D points is developed, meanwhile more comprehensive evaluation and analysis of the proposed tensor-based sparse representation are conducted in this publication. Besides, this publication proposed a method to measure height difference without using a DTM.

This publication followed the same procedure as proposed in the publication I. For the high-dimensional tensor representation, global Cartesian coordinates of the neighboring points are transformed to the local PCA (Principal Component Analysis) coordinate system before the voxelization, such that the planar points can share the same spatial distribution in the tensor regardless the orientation. Height difference is an important feature to distinguish ground and non-ground points, which is normally calculated between a center points and the lowest point found in its neighborhood. In an ideal case, the ground point should be selected as the lowest point and have a small height difference. However, the ground points tend to have relatively large height differences in the areas with inclined ground. In this publication, the lowest point is found in a multiple scale cylindrical neighborhood, and the height difference is determined by a given threshold that indicates the minimal value for height difference.

The evaluation is carried out on 8 tiles of ALS dataset of Vienna city, and is compared with four classic machine learning methods, namely K Nearest Neighbors, Decision Tree, Random Forest and Support Vector Machine. The results have demonstrated the effectiveness of TSRC, the OAs of TSRC are beyond 80% with only 27 training tensors used per class. Compared with other classifiers, TSRC has respectable performance in identifying objects with less distinguishable features, such as façade. Moreover, a series of experiments of TSRC suggest that the TSRC is barely dependent on the neighborhood size of tensor generation and the sparsity level.

Through the comparison and analysis of the classification results, the effectiveness of TSRC is further confirmed. However, we also observed that the improvement in classification performance by TSRC is very limited, and the computation time is demanding due to the high dimensionality of the tensor representation.

Publication III

This publication developed a contextual label smoothing method to improve classification accuracy in a post-processing step. A framework of global graph-structured regularization for label smoothing was proposed in the study (Landrieu, 2017), and was successfully applied on LiDAR

point clouds to refine initial classification results. However, it is difficult to decide a constant neighborhood size across the entire scene. Wrong labels remain in some isolated clusters by using a small neighborhood size, while a large neighborhood size tends to cause over-smoothing. Moreover, in some regions with a poor initial classification, the neighborhood consistency is not reliable enough to deduce the correct label interactions. Thus, an improvement of initial classification is required to provide sufficient reliable contextual information.

Under the framework of graphical regularization, this publication developed a strategy to enhance effectiveness of label smoothing from two aspects. First, we designed an optimal neighborhood that can adaptively select geometry-relevant points from a large initial neighborhood, so that each point can collect sufficient long-range interactions without over-smoothing small objects. Second, probabilistic label relaxation (PLR) is utilized to improve the initial label probability set according to the spatial correlation of class labels, e.g., the points above a vegetation point are likely vegetation rather than belonging to other classes. In order to build explicit spatial relationships of semantic labels between 3D points, a cylindrical neighborhood that consists of upper neighbors, middle neighbors and lower neighbors is used. Then, combined with corresponding neighborhood compatibility, the label probabilities of neighboring points iteratively reinforce the weights to each class probability of the concerned point. After PLR, the resulting probability values are expected to be more consistent with the prior knowledge of neighborhood dependencies. In the end, with this optimal graph and the updated label probability set, the final labels can be computed by graph-structured regularization.

The contextual label-smoothing approach is evaluated on two independent urban airborne LiDAR datasets with complex urban scenes: ALS dataset of Vienna city and the Vaihingen dataset in ISPRS 3D benchmark. Significant improvement in the classification accuracy is achieved without losing the accuracy of small objects (such as façades and cars). The overall accuracy is increased by 7.01% on the Vienna dataset and 6.88% on the Vaihingen dataset. Moreover, the strategy proposed in this publication effectively solved the aforementioned issues. Most clusters that contain wrong labels are correctly classified with the aid of optimal neighborhood, and the refinement of initial label probability successfully improve classification performance in regions where correct label interactions were missing from the neighborhood.

The main contribution of this publication is to develop a robust contextual label-smoothing approach, which only takes the label probability set and 3D coordinates as input and require a few of parameters to define neighborhood size. Furthermore, the neighborhood compatibility coefficients are object driven rather than data driven, thus it can also be derived from other sources of data in different regions as long as the semantics are consistent. Therefore, this publication is particularly valuable for label refinement in urban environment which is characterized by the composition of fixed types of objects with specific spatial distributions.

Publication IV

This publication conducted a comprehensive comparison between three state-of-the-art deep learning architectures: namely PointNet++, KPConv and SparseCNN. PointNet++ is a point-wise network, in which features are learned by successive levels of MLPs followed by order-invariant pooling operations. In contrast, SparseCNN uses voxels as internal representation and applies convolutional operations directly on this regular grid, as CNNs normally specialize in processing data that has a grid-like topology. KPConv employs a 3D convolution in which 3D spatial points are used as kernel points to carry weights, thus the convolution in KPConv is directly applied on 3D point clouds without any voxelization.

Compared to the MLP based networks, CNNs are theoretically considered advantageous, because they can exploit the spatial arrangement of points inherently, whereas MLP takes flattened vectors as input, which leads to a loss of spatial information. In a nutshell, while the CNNs employed in

KPConv and SparseCNN are considered superior to MLPs used in PointNet++, the 3D kernerlization in KPConv may be very memory-demanding and the resolution loss by SparseCNN may decrease classification accuracy. Thus, a deep investigation is required to examine the capabilities of the three architectures for classification of ALS point clouds.

The performances of the three deep learning networks are compared w.r.t. classification accuracy, computation time, generalization ability as well as the sensitivity to the choices of hyper-parameters. In order to conduct a comprehensive comparison, the classification is performed on two large-scale ALS datasets with very different application purposes. Overall, PointNet++, KPConv and SparseCNN are all superior to the Random Forest that uses the handcrafted features computed at multiple scales. Compared to PointNet++ and KPConv, SparseCNN has a better classification performance on both two datasets, without high demands for time and memory. Moreover, SparseCNN shows a better generalization ability and is less affected by the different choices of hyper-parameters. KPConv slightly outperforms PointNet++, especially in the detection of small-size objects.

The results in this publication have proven the effectiveness of the selected deep learning methods, but also explored the defects of PointNet++, KPConv and SparseCNN in classification. The findings are in accordance with the previous statement that CNNs are more powerful than MLPs for semantic classification, as well as the expensive memory cost of KPConv. Additionally, we also observed that small objects such as cars and insulators on the power line facilities, can be well classified by SparseCNN, this indicates that the resolution loss caused by voxelization has quite small impact on the classification results, which is negligible compared to the gains.

Publication V

This publication made an effort to automatically extent a small set of training data by label propagation. Supervised classification methods rely heavily on the quantity and quality of the training data. The training data should provide a full representative of all concerned classes to allow the classifier to find the best solution for the given data. However, the generation of training data is a difficult and expensive task for large-scale point clouds. Thus, it's a common problem for large amounts of data that only a small amount of reference points can be manually labeled due to the limited economical and temporal resources.

To solve the aforementioned issue, this publication presented an active learning method to automatically select new samples whose inclusion in the training set would be beneficial to classification performance. By the assumption that the knowledge about class labels from the training set can be correctly extended to their well-defined neighborhood, the informative points are iteratively selected from the neighborhood of training points to update the training set. Experiments have shown that a more homogenous and accurate classification result can be obtained by extending training set. A further analysis indicated that, the most informative samples are selected from the 1st iteration, whereas samples selected from the later iterations only produce slight impact on the classification performance.

However, new training samples are selected at an individual point level in a local neighborhood, as a result, the classification improvement is very limited and is fundamentally impacted by the initial amount of training samples. And the diversity of spatial distribution can not be enhanced by selecting new training samples only in a local neighborhood.

1.4.2 Author contributions

- **Publication I:** Design of the study; implementation of algorithms; writing of the final article.
- **Publication II:** Design of the study; crafting height features; implementation of algorithms; writing of the final article.
- **Publication III:** Design of the study; implementation of algorithms; writing of the final article.
- **Publication IV:** Design of the study; implementing existing deep learning models on ALS point clouds; writing of the final article.
- **Publication V:** Design of the study; implementation of algorithms; writing of the final article.



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

2 Publications

Publication I

- Title: Airborne LiDAR Points Classification Based on Tensor Sparse Representation
- Authors: Li, N., Pfeifer, N. and Liu, C.
- Published in: ISPRS Geospatial Week 2017, ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences
DOI: 10.5194/isprs-annals-IV-2-W4-107-2017

This is a reproduced version of the publication I, and the missing part of the first equation in section 2.2.1 is added, while the equation is not complete in the published version.



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Airborne LiDAR Points Classification Based on Tensor Sparse Representation

N. Li^{a,b}, C. Liu^{a,*}, N. Pfeifer^b

^a College of Survey and Geoinformation, Tongji University, 200092, Shanghai, China

^b Department of Geodesy and Geoinformation, Technische Universität Wien, 1040 Vienna, Austria

ISPRS WG II/3

KEY WORDS: Point cloud, Sparse coding, Dictionary Learning

ABSTRACT:

The common statistical methods for supervised classification usually require a large amount of training data to achieve reasonable results, which is time consuming and inefficient. This paper proposes a tensor sparse representation classification (SRC) method for airborne LiDAR points. The LiDAR points are represented as tensors to keep attributes in its spatial space. Then only a few of training data is used for dictionary learning, and the sparse tensor is calculated based on tensor OMP algorithm. The point label is determined by the minimal reconstruction residuals. Experiments are carried out on real LiDAR points whose result shows that objects can be distinguished by this algorithm successfully.

1. INTRODUCTION

LiDAR point cloud classification in urban areas has always been an essential and challenging task. Due to the complexity in urban scenes, it is difficult to label objects correctly using only single or multi thresholds. Thus, research mainly focus on the use of statistical method for supervised classification of LiDAR points in recent years. Common machine learning methods include support vector machine (SVM) algorithm, adaboost, decision trees, random forest and other classifiers. SVM seeks out the optimal hyperplane that efficiently separates the classes, and the Gaussian kernel function can be used to map non-linear decision boundaries to higher dimensions where they are linear (Secord and Zakhor, 2007). Adaboost is a binary algorithm, but several extensions are explored for

multiclass categorization, hypothesis generation routines are used to classify terrain and non-terrain area (Lodha et al., 2007). Decision trees can be used to carry out the classification by training data and make a hierarchical binary tree model, new objects can be classified based on previous knowledge (Garcia-Gutierrez et al, 2009) (Niemeyer et al., 2013). Random Forest is an ensemble learning method that uses a group of decision trees, provides measures of feature importance for each class (Guo et al., 2011) (Niemeyer et al., 2013), and runs efficiently on large datasets.

Those approaches barely consider the spatial distribution of points, which is an important cue for the classification in complex urban scenes. Some studies have applied graphical models to incorporate spatial context information in the classification. Graphical

models take neighboring points into account, which allow us to encode the spatial and semantic relationships between objects via a set of edges between nodes in a graph (Najafi et al. 2014). Markov network and conditional random field (CRF) are two mainstream methods to define the graphical model. However, a large amount of training data is necessary to obtain the classifier in those statistical studies. Angelov et al (Angelov et al. 2005) use Associated Markov Network (AMN) to classify objects on the ground. The study takes 1/6 points as training data and achieve an overall classification accuracy as 93%. Niemeyer et al (Niemeyer et al. 2014) use 3000 points per class to train the CRF model. Seven classes (grass land, road, tree, low vegetation, buildings gable, building flat, facade) are distinguished based on the CRF and the overall accuracy is 83%, which is a fine result in complex urban scene. As for statistical methods, Im (Im et al. 2008) uses 316 training samples (1%) to generate decision trees with an overall accuracy of 92.5%. Niemeyer (Niemeyer et al., 2013) uses 3000 samples per class to build random forest model, the overall accuracy achieves 83.7%. Moreover, Lodha uses half of dataset as training data through adaboost algorithm and the average accuracy is 92%. As a consequence, classifier training would be very time-consuming, especially when Markov network or CRF are used as classifier.

This paper aims to use as few training data as possible to achieve effective classification. Therefore, sparse representation-based classification is used in this paper. Sparse representation-based classification (SRC) is a well-known technique to represent data sparsely on the basis of a fixed dictionary or learned dictionary. It classifies unknown data based on the reconstruction criteria. SRC has been successfully applied to the processing of signals (Huang and Aviyente 2006) and images (Wright et al. 2009). Normally, the dimensional data has to be embedded into vectors in traditional methods. However, the vectorization breaks the original multidimensional structure of the signal and reduces the reliability of post processing. Therefore, some research formulates high dimensional data SRC

problem in terms of tensors. Tensor extensions of the dictionary learning and sparse coding algorithms have been developed, such as Tensor MOD and KSVD for dictionary learning (Roemer et al. 2014), tensor OMP (Caiafa and Cichocki 2012). Moreover, tensor based representation has yielded good performance in high-dimensional data classification (Renard and Bourennane 2009), face recognition (Lee et al. 2015) and image reduction (Peng et al. 2014).

We represent LiDAR point as a 4-order tensor to keep feature description in their original geometrical space. With few training data, the dictionary is learned based on the Tucker decomposition (Kolda and Bader 2009). Then, the sparse representation of each point can be obtained by projecting the tensor onto dictionaries, which is expressed as a sparse tensor whose nonzero entries correspond to the selected training samples. Thus, the sparse tensors of points that belong to the same class should have similar structure. At last, the label of unknown points can be predicted by the minimum reconstruction residual from sparse tensor and dictionaries.

2. LIDAR CLASSIFICATION BASED ON SPARSE REPRESENTATION

2.1 Sparse Representation Classification

The sparsity algorithm is to find the best representative of a test sample by sparse linear combination of training samples from a dictionary (Wright et al. 2009). Given a certain number of training samples from each class, the sub-dictionary D_i from i^{th} class is learned. Assume that there are c classes of subjects, and let $D = ([D_1], [D_2], [D_3] \dots [D_c])$ which is the overall structured dictionary over the entire dataset. Denote by y a test sample, the sparse coefficient x is calculated by projecting y on dictionary D , which is called sparse coding procedure.

SRC use the reconstruction error e_i associated with each class to do data classification. x_i is the sparse coefficient associated with class i , e_i is the reconstruction error from sub-dictionary in i class.

The class label of y is then determined as the one with minimal residual.

$$e_i = \|y - D_i x_i\|_2 \quad \text{class } i = [1, 2, \dots, c] \quad (1)$$

$$\text{identify}(y) = \arg \min_i \{e_i\}$$

2.2 Tensor Representation of Lidar Points

2.2.1 Preliminaries on tensors

The tensor is to denote a multidimensional object, whose the elements are to be addressed by more than two indices. The order of a tensor, also known as modes (Kolda and Bader 2009), is the number of dimensions. Tensors are denoted as boldface italic capital letters, e.g., \mathbf{A} ; matrices are denoted as italic capital letters, e.g., A ; vectors are denoted as italic lowercase letters, e.g., a .

The tensor can be transformed into a vector or matrix, and this processing is known as unfolding or flattening. Given an N -order tensor $\mathbf{T} \in \mathbf{R}^{I_1 \times I_2 \times \dots \times I_N}$, the n -mode unfolding vector of tensor \mathbf{T} is obtained by fixing every index except the one in the mode n (Yang et al., 2015). The n -mode unfolding matrix is defined by arranging all the n -mode vectors as columns of a matrix, i.e., the n -mode unfolding matrix $T_{(n)} \in \mathbf{R}^{I_1 \times I_2 \times \dots \times I_{n-1} \times I_{n+1} \times \dots \times I_N, I_n}$. The n -mode product of a tensor $\mathbf{T} \in \mathbf{R}^{I_1 \times I_2 \times \dots \times I_N}$ with a matrix $U \in \mathbf{R}^{J \times I_n}$ is denoted by $\mathbf{T} \times_n \mathbf{U}$, and the processing can convert to product that each mode- n vector is multiplied to the matrix \mathbf{U} . so it can also be expressed in terms of unfold tensors:

$$\mathbf{Y} = \mathbf{T} \times_n \mathbf{U} \Leftrightarrow \mathbf{Y}_n = \mathbf{U} \times \mathbf{T}_{(n)}$$

The tucker decomposition is a form of higher-order principal component analysis. It decomposes a tensor $\mathbf{T} \in \mathbf{R}^{I_1 \times I_2 \times \dots \times I_N}$ into a core tensor $\mathbf{C} \in \mathbf{R}^{J_1 \times J_2 \times \dots \times J_N}$ multiplied by the matrix $\mathbf{U}_1, \mathbf{U}_2, \dots, \mathbf{U}_N$ along each mode. The matrix can be considered as the principal components in each mode. Since the tensor is generated according to the point spatial distribution, the principal components in attribute mode are spatial connection considered.

$$\mathbf{T} = \mathbf{C} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times \dots \times_n \mathbf{U}_N$$

2.2.2 Tensor representation of Lidar points voxel

In order to preserve spatial structure and attribution information, LiDAR points are represented as tensor data. In previous work, the LiDAR data is rasterized into feature images. The LiDAR tensor is generated by stacking images into 3-order tensor (Li et al., 2016). This paper consider each point voxel as a tensor. First of all, multiple attributes from raw LiDAR data are extracted to form a vector on the point \mathbf{p} , then the 3D neighborhood of the point \mathbf{p} is selected as the voxel of point \mathbf{p} . After that, the voxel is represented as a 4-order tensor $\mathbf{T} \in \mathbf{R}^{I_x \times I_y \times I_z \times I_a}$ of point \mathbf{p} , where I_x, I_y, I_z, I_a indicate the X, Y, Z coordinates and attributes mode, respectively. \mathbf{R} is the real manifold. Points in this voxel are regarded as entries in the tensor, which are arranged as $\Gamma_{I_x I_y I_z I_a}$, where $i_x = 1, \dots, I_x; i_y = 1, \dots, I_y; i_z = 1, \dots, I_z; i_a = 1, \dots, I_a$. The voxel size is defined as 1m and tensor size is defined as $10 \times 10 \times 10 \times 10$. It means that the voxel is equally partitioned as 10 intervals along X, Y, Z coordinate, and 10 attributes contained in each point. Therefore, the entries are the attributes of each point, which are accessed via I_x, I_y, I_z, I_a indices. That means, attributes are spatially constrained along local direction and implicitly exploited by tensor representation.

The tensor can be represented in terms of its factors using the Tucker model, which is shown as equation (2).

$$\mathbf{T} \approx \mathbf{X} \times_1 U^{(x)} \times_2 U^{(y)} \times_3 U^{(z)} \times_4 U^{(a)} \quad (2)$$

Here, $U^{(x)} \in \mathbf{R}^{I_x \times J_x}$, $U^{(y)} \in \mathbf{R}^{I_y \times J_y}$, $U^{(z)} \in \mathbf{R}^{I_z \times J_z}$, $U^{(a)} \in \mathbf{R}^{I_a \times J_a}$ are the factor matrices and contain the basis vectors on X coordinate, Y coordinate, Z coordinate and attribute mode. $\mathbf{X} \in \mathbf{R}^{I_x \times J_y \times J_z \times J_a}$ is the core tensor, where $J_x, J_y, J_z, J_a \leq I_x, I_y, I_z, I_a$, and its entries show the level of interaction between the different components (Tamara et al., 2007). As such $J_x, J_y, J_z, J_a \leq I_x, I_y, I_z, I_a$, the original tensor can be well recovered with the core tensor and a few basis vectors on each mode.

Tucker mode can be written as Kronecker representation: the two representations are equivalent. Let \otimes denotes the Kronecker product, t is the vectorized version of tensor \mathbf{T} , x is the vectorized

version of tensor \mathbf{X} . The equal Kronecker representation is shown as following:

$$t = (U^{(x)} \otimes U^{(y)} \otimes U^{(z)} \otimes U^{(a)})x \quad (3)$$

2.3 Dictionary Learning

For a set of LiDAR tensors $\{\mathbf{T}_k\}_{k=1}^K$, where $\mathbf{T}_k \in \mathbf{R}^{I_x \times I_y \times I_z \times I_a}$ is 4-order point tensor and K is number of tensors. Dictionaries $D = [D^x, D^y, D^z, D^a]$, and D^x, D^y, D^z, D^a are dictionaries on X coordinate, Y coordinate, Z coordinate and attribute mode, respectively. Tensor dictionary learning aims to calculate the dictionaries $D = [D^x, D^y, D^z, D^a]$ and sparse tensor $\{\mathbf{X}_k\}_{k=1}^K$ by following model. ($\|\cdot\|_2$ denotes the l_2 -norm).

$$\min_{D^x, D^y, D^z, D^a} \sum_{k=1}^K \|\mathbf{T}_k - \mathbf{X}_k \times_1 D^x \times_2 D^y \times_3 D^z \times_4 D^a\|_2 \quad (4)$$

The dictionary learning can be performed independently for each class to build a sub-dictionary. Denoted by $D_1^x, D_1^y, D_1^z, D_1^a$ are sub-dictionaries associated with class i on X coordinate, Y coordinate, Z coordinate and attribute mode, class $i = [1, 2, \dots, c]$. Let $\{\mathbf{T}_{ik}\}_{k=1}^K$ be the training tensor set

from class i , and K is number of tensors belong to class i . The sub-dictionaries from each class $D_1^x, D_1^y, D_1^z, D_1^a$ can be learned from model:

$$\min_{D_1^x, D_1^y, D_1^z, D_1^a} \sum_{k=1}^K \|\mathbf{T}_{ik} - \mathbf{X}_{ik} \times_1 D_1^x \times_2 D_1^y \times_3 D_1^z \times_4 D_1^a\|_2 \quad (5)$$

Equation (5) can be solved by the Tucker decomposition based on equation (2).

Every training point tensor \mathbf{T}_{ik} from class i is tucker decomposed to get the $U^{(x)}, U^{(y)}, U^{(z)}, U^{(a)}$, then a certain number of basis vectors of $U^{(x)}, U^{(y)}, U^{(z)}, U^{(a)}$ are added into dictionaries $D_1^x, D_1^y, D_1^z, D_1^a$. The final dictionaries D^x, D^y, D^z, D^a are described as following, c is the number of classes.

$$\begin{aligned} D^x &= [D_1^x, D_2^x, \dots, D_c^x]; \\ D^y &= [D_1^y, D_2^y, \dots, D_c^y]; \\ D^z &= [D_1^z, D_2^z, \dots, D_c^z]; \\ D^a &= [D_1^a, D_2^a, \dots, D_c^a]; \end{aligned}$$

Algorithm: Tensor OMP

Require: input point tensor $\mathbf{T} \in \mathbf{R}^{I_1 \times I_2 \times I_3 \times I_4}$, Dictionaries $D^x \in \mathbf{R}^{I_1 \times J_1}$, $D^y \in \mathbf{R}^{I_2 \times J_2}$, $D^z \in \mathbf{R}^{I_3 \times J_3}$, $D^a \in \mathbf{R}^{I_4 \times J_4}$, maximum number of non-zeros coefficients k in each mode.

Output: sparse tensor \mathbf{X} , non-zeros coefficients index in sparse tensor (M_1, M_2, M_3, M_4)

Step:

- 1, initial: $M_n = [\emptyset] (n = 1, 2, 3, 4)$, Residual $\mathbf{R} = \mathbf{T}$, $\mathbf{X} = \mathbf{0}$, $k=0$, $t = \text{vec}(\mathbf{T})$
 - 2, while $\|\mathbf{M}_n\|_0 \leq k$ do
 - 3, $[m_1, m_2, m_3, m_4] = \arg \max_{[m_1, m_2, m_3, m_4]} |\mathbf{R} \times_1 D^{xT}(:, m_1) \times_2 D^{yT}(:, m_2) \times_3 D^{zT}(:, m_3) \times_4 D^{aT}(:, m_4)|$
 - 4, $M_n = M_n \cup [m_1, m_2, m_3, m_4] (n = 1, 2, 3, 4)$. $TD^x = D^x(:, M_1)$, $TD^y = D^y(:, M_2)$, $TD^z = D^z(:, M_3)$, $TD^a = D^a(:, M_4)$;
 - 5, $x = \arg \min_u \|(TD^a \otimes TD^z \otimes TD^y \otimes TD^x)u - t\|_2^2$;
 - 6, $\mathbf{X} = \text{tensorize}(x)$;
 - 6, $\mathbf{R} = \mathbf{T} - \mathbf{X} \times_1 TD^1 \times_2 TD^2 \times_3 TD^3 \times_4 TD^4$;
 - 7, $t=t+1$;
 - 8, end while
 - 9, return \mathbf{X} , (M_1, M_2, M_3, M_4)
-

2.4 Tensor Sparse Representation for Classification

The objective of tensor sparse coding is to find a sparse representation of a tensor T with respect to the factors D on each mode. This means that the sparse coding is obtained by solving following optimization model:

$$\min_X \sum_{k=1}^K \|T - X \times_1 D^x \times_2 D^y \times_3 D^x \times_4 D^x\|_2$$

subject to $\|X\|_0 \leq n$ (6)

$\|X\|_2$ denotes the l_0 -norm of tensor X , which is also considered as the sparsity of tensor X . The problem is presented as minimizing the approximation error within a certain sparsity level, which can be approximately solved by greedy pursuit algorithms such as Orthogonal Matching Pursuit (OMP). Classical OMP locates the support of the sparse vector that have best approximation of sample data from dictionary. It selects the support set by one index at each iteration until K atoms are selected or the approximation error is within a preset threshold (Chen et al. 2011), where K is the sparsity. We use the steps of the classical OMP algorithm for tensors as it is shown in Algorithm TensorOMP. The algorithm is proposed by Caiafa and Cichocki (Caiafa and Cichocki, 2012).

In the step 5 of the algorithm, TD^x , TD^y , TD^z , TD^a correspond to the sub-dictionaries obtained by restricting the n -mode dictionaries to the columns indicated by indices M_n .

3. EXPERIMENT

3.1 Data description

We perform the classification on two sections of airborne LiDAR dataset of Vienna city. The density of datasets mostly range from 8 to 75 points $/m^2$. The area of both dataset is 100×100 m. The dataset1 is with flat terrain and contains 710870 points. The dataset2 has more complex environment and 817939 points in total. Both datasets contains complex objects like buildings with various height and shape, single trees, grouped and low vegetation, hedges, fences, cars and telegraph poles. In the classification procedure, the

objects are categorized into 5 classes: *open ground* which is uncovered or not blocked by any objects; *building roof*; *vegetation*; *covered ground* which is usually under the high trees or building roof; *building wall*. However, in the evaluation session the open ground and covered ground are merged into ground to achieve the overall ground detection accuracy.

To build the tensors, the neighborhood threshold is defined as 1 meter for selecting points into the voxel, then the voxel is represented as a 4-order tensor $T \in \mathbf{R}^{10 \times 10 \times 10 \times 10}$. It means that the spatial coordinates of the voxel are regularized into a $10 \times 10 \times 10$ cube, and 10 attributes are attached on each point. The entries are the normalized attribute values and accessed via four indices. Fig1 shows the points in the voxel and tensor representation by X, Y, Z coordinate indices. And the 10 attributes are described as following:

(1) Relative height. It is a binary value, which is defined as 1 if the point height above the threshold, otherwise is defined as 0. This is useful for indicating ground and non-ground points.

(2) NormalZ. NormalZ are the normal vectors of local planes in Z direction, which are estimated by points in a small neighborhood.

(3)-(5) Eigenvalue1; Eigenvalue2; Eigenvalue3. The covariance matrix for the normal vectors is computed to find the eigenvalues, which include Eigenvalue1 λ_1 ; Eigenvalue2 λ_2 ; Eigenvalue3 λ_3 ($\lambda_1 > \lambda_2 > \lambda_3$). λ_2 λ_3 have low values for planar object and higher values for voluminous point clouds. Three structure features derived from eigenvalues are anisotropy, sphericity and planarity, which describe the spatial local points' distribution and defined as following equation (Chehata et al., 2009).

(6) Anisotropy. Anisotropy = $(\lambda_1 - \lambda_3)/\lambda_1$.

(7) Sphericity. Sphericity = $\lambda_3/\lambda_1(\lambda_1 - \lambda_3)/\lambda_1$.

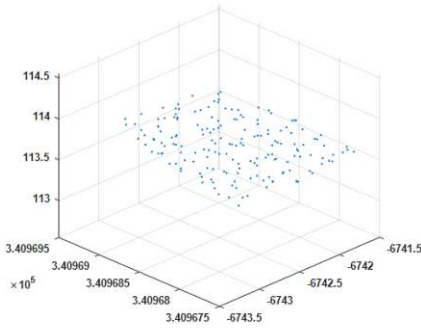
(8) Planarity. Planarity = $(\lambda_2 - \lambda_3)/\lambda_1$.

(9) NormalSigma0. The standard deviation of normal estimation. The value would be high in rough area and low in smooth area.

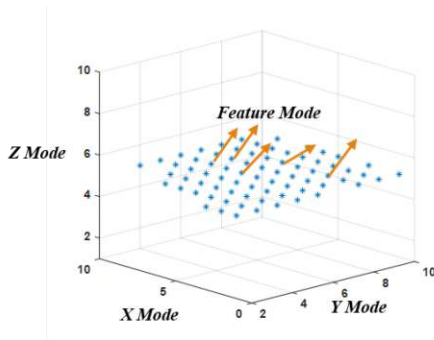
(10) Echo Ratio. the echo ratio is a measure for local transparency and roughness. It is defined as follows(Höfle et al. 2009)

$$ER = n_{3D}/n_{2D} \times 100$$

With $n_{3D} \leq n_{2D}$, n_{3D} is the number of neighbors found in a certain search distance measured in 3D and n_{2D} is the number of neighbors found in same distance measured in 2D. The ER is nearly 100% for flat surface, whereas the ER decreases for penetrable surface parts since there are more points in a vertical search cylinder than there are points in a sphere with the same radius.



(a) LDAR ground points visualization in the voxel



(b) LiDAR ground points visualized by 4-order tensor form

Figure 1. Point distribution in voxel and tensor

3.2 Classification

The experiments show that the LiDAR tensor can be fully recovered with the compressed core tensor $6 \times 6 \times 6 \times 6$, which means it needs at least 6 basis vectors in each mode and corresponding core tensor for reconstruction. Hence, only 6 basis vectors are added into each sub-dictionary, and the final dictionary would be a 10×30 matrix in each mode.

First of all, the sub-dictionary associated with a specific class i is created. Denote by $D_i^x, D_i^y, D_i^z, D_i^a$ the sub-dictionary associated with class i on mode X,Y,Z and attribute, $class\ i = [1,2,3,4,5] \cdot 6$ training tensors are randomly selected from each class, T_i^k donates the k -th training tensor in class i , $k = 1,2 \dots 6$. Training tensor T_i^k is decomposed by Tucker model to get basis vectors in each mode:

$$T_i^k \approx X_i^k \times_1 U_i^{(x)} \times_2 U_i^{(y)} \times_3 U_i^{(z)} \times_4 U_i^{(a)} \quad (7)$$

The first column in matrix $U_i^{(x)}, U_i^{(y)}, U_i^{(z)}, U_i^{(a)}$ is added into the corresponding sub-dictionary in each mode. Thus, $D_i^x, D_i^y, D_i^z, D_i^a$ can be described as :

$$D_i^x = [U_1^{(x)}(:,1), U_2^{(x)}(:,1) \dots U_n^{(x)}(:,1)]$$

$$D_i^y = [U_1^{(y)}(:,1), U_2^{(y)}(:,1) \dots U_n^{(y)}(:,1)]$$

$$D_i^z = [U_1^{(z)}(:,1), U_2^{(z)}(:,1) \dots U_n^{(z)}(:,1)]$$

$$D_i^a = [U_1^{(a)}(:,1), U_2^{(a)}(:,1) \dots U_n^{(a)}(:,1)]$$

And the final dictionary on each mode can be represented as following:

$$D^x = [D_1^x, D_2^x, D_3^x, D_4^x, D_5^x]$$

$$D^y = [D_1^y, D_2^y, D_3^y, D_4^y, D_5^y]$$

$$D^z = [D_1^z, D_2^z, D_3^z, D_4^z, D_5^z]$$

$$D^a = [D_1^a, D_2^a, D_3^a, D_4^a, D_5^a]$$

3.3 Classification Result And Discussion

Visual inspection indicates that most objects are detected correctly in Fig2. The overall classification accuracy is 82% for dataset1 and 80% for dataset2. Tab1 and Tab2 are the confusion matrices which demonstrates prediction ability of the algorithm on various objects.

Some buildings and trees are extracted from dataset1 and dataset2 for error points analysis. Fig3(a) and (e) indicate that some parts of boundary points in roof in dataset2 are misclassified into ground(12.3%), but the algorithm performs very well in identifying roofs with dataset1, which achieve a high accuracy of 98.3%. 8% of vegetation are wrongly predicted as walls in dataset1, which is mainly caused by trees with a vertical structure or high pruned and trimmed trees (Fig

3(c) and (g)). And 6.9% of vegetation are misclassified into roofs in dataset1. This usually occurred on low flat vegetation which has similar attributes with roofs, examples can be found in Fig3 (d) and (f). Wall points are usually located in a more complex scenario. In Fig4 (a) and (e), it is a balcony wall and ends up confused with vegetation. In Fig4 (b) and (f), top and bottom wall points are labeled as roof and ground due to the close location to roof and ground, but middle part of walls can obtain correct labels. Thus, this algorithm works

well in distinguishing objects with clear spatial structures.

Considering that only 30 training points are used, this tensor SRC algorithm can achieve an overall good performance, especially in roof identification. However, high accuracy cannot be obtained in complex scene, such as wall boundary identification from roof and ground. Since the main part objects could be correctly labeled, the error points can be reduced by further filtering method.

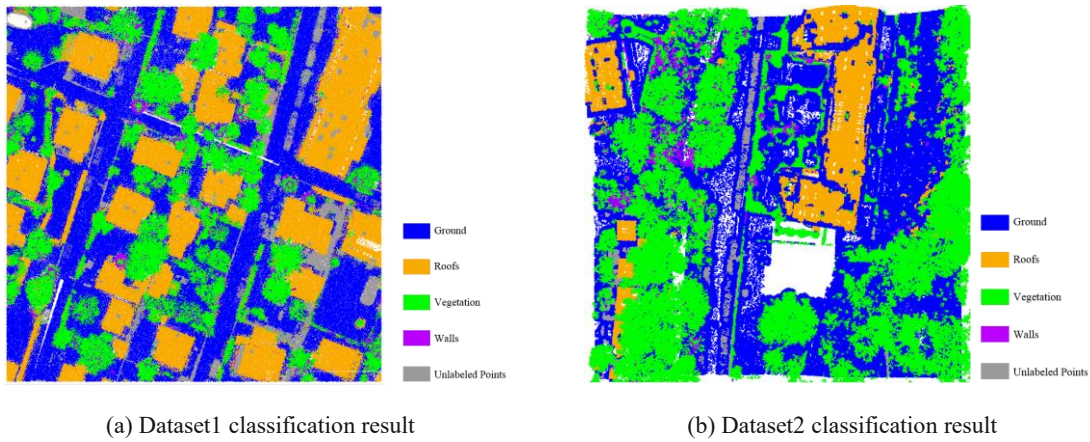


Figure 2. Classification result

Table 1. Dataset1 Classification confusion matrix

Class	Ground	Roofs	Veg	Walls
Ground	89.0%	7.0%	2.2%	1.8%
Roofs	0.3%	98.3%	1.1%	0.2
Veg	5.1%	6.9%	79.7%	8.3%
Walls	17.9%	15.5%	9.7%	57.0%

Table 2. Dataset2 Classification confusion matrix

Class	Ground	Roofs	Veg	Walls
Ground	79.5%	3.0%	15.3%	2.15%
Roofs	12.3%	84.3%	3.4%	0%
Veg	7.5%	2.2%	85.8%	4.6%
Walls	4.9%	3.8%	20.9%	70.4%

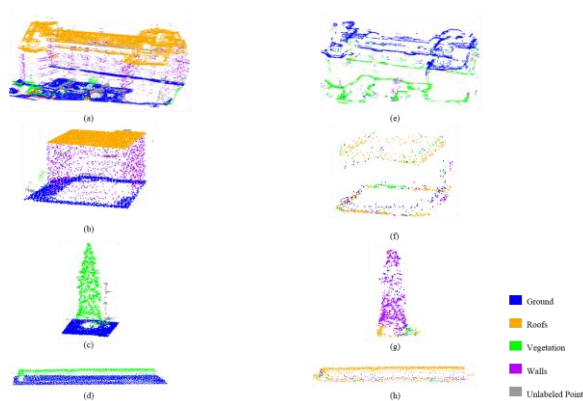


Figure 3. Reference and misclassified point in 4 scenes

4. CONCLUSION

A tensor sparse representation classification method has been proposed and tested with real airborne LiDAR data. The method integrates spatial distribution and attributes by tensor representation. Only 6 training points from each class are utilized to build the dictionary. It achieves an overall classification accuracy of 82%. This algorithm has respectable performance in distinguishing object with clear shape pattern. Further work will focus on the dictionary improvement based on dictionary learning algorithm, which can distinguish more minor and unambiguous objects.

5. ACKNOWLEDGEMENT

The LiDAR data comes from Vienna city government. The project is supported by National Natural Science Foundation of China (Project No.41371333).

6. REFERENCE

Anguelov, D., B. Taskarf, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz and A. Ng., 2005. "Discriminative learning of markov random fields for segmentation of 3d scan data". Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, IEEE.

Caiafa, C. F. and A. Cichocki., 2012. "Block sparse representations of tensors using Kronecker bases". Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, IEEE.

Chehata, N., L. Guo and C. Mallet., 2009. Airborne lidar feature selection for urban classification using random forests. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* **38**(Part 3): W8.

Chen, Y., N. M. Nasrabadi and T. D. Tran., 2011. Hyperspectral image classification using dictionary-based sparse representation. *IEEE Transactions on Geoscience and Remote Sensing* **49**(10): 3973-3985.

Höfle, B., W. Mücke, M. Dutter, M. Rutzinger and P. Dorninger., 2009. "Detection of building regions using airborne lidar—A new combination of raster and point

cloud based GIS methods". Proceedings of GI-Forum 2009—International Conference on Applied Geoinformatics.

Huang, K. and S. Aviyente., 2006. "Sparse representation for signal classification". NIPS.

Im, J., J. R. Jensen and M. E. Hodgson., 2008. Object-based land cover classification using high-posting-density LiDAR data. *GIScience & Remote Sensing* **45**(2): 209-228.

Joachim Niemeyer, F. R., Uwe Soergel., 2013. "Classification of Urban LiDAR data using Conditional Random Field and Random Forests". Proc. 7th IEEE/GRSS/ISPRS Joint Urban Remote Sens.

Kolda, T. G. and B. W. Bader., 2009. Tensor decompositions and applications. *SIAM review* **51**(3): 455-500.

Lee, Y. K., C. Y. Low and A. B. J. Teoh., 2015. "Tensor kernel supervised dictionary learning for face recognition". Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2015 Asia-Pacific, IEEE.

Li, N., C. Liu, N. Pfeifer, J. F. Yin, Z. Y. Liao and Y. Zhou. 2016. TENSOR MODELING BASED FOR AIRBORNE LiDAR DATA CLASSIFICATION. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* XLI-B3: 283-287.

Najafi, M., S. T. Namin, M. Salzmann and L. Petersson., 2014. "Non-associative higher-order markov networks for point cloud classification". European Conference on Computer Vision, Springer.

Niemeyer, J., F. Rottensteiner and U. Soergel., 2014. Contextual classification of lidar data and building object detection in urban areas. *ISPRS journal of photogrammetry and remote sensing* **87**: 152-165.

Peng, Y., D. Meng, Z. Xu, C. Gao, Y. Yang and B. Zhang., 2014. "Decomposable nonlocal tensor dictionary learning for multispectral image denoising". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Renard, N. and S. Bourennane., 2009. Dimensionality reduction based on tensor modeling for classification methods. *Geoscience and Remote Sensing, IEEE Transactions on* **47**(4): 1123-1131.

- Roemer, F., G. Del Galdo and M. Haardt., 2014. "Tensor-based algorithms for learning multidimensional separable dictionaries". Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, IEEE.
- Wright, J., A. Y. Yang, A. Ganesh, S. S. Sastry and Y. Ma., 2009. Robust face recognition via sparse representation. *IEEE transactions on pattern analysis and machine intelligence* 31(2): 210-227.
- Yang, S., M. Wang, P. Li, L. Jin, B. Wu and L. Jiao. 2015. Compressive hyperspectral imaging via sparse tensor and nonlinear compressed sensing. *IEEE Transactions on Geoscience and Remote Sensing* 53(11): 5943-5957.



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Publication II

Title: Tensor-based sparse representation classification for Urban Airborne LiDAR points

Authors: Li, N., Liu, C. and Pfeifer, N

Published in: Remote Sensing, 9(12), 1216
DOI: 10.3390/rs9121216



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Article

Tensor-Based Sparse Representation Classification for Urban Airborne LiDAR Points

Nan Li ^{1,2}, Norbert Pfeifer ¹ and Chun Liu ^{2,*}

¹ Department of Geodesy and Geoinformation, Technische Universität Wien, 1040 Vienna, Austria; nan.li@tuwien.ac.at or 123linan@tongji.edu.cn (N.L.); Norbert.pfeifer@tuwien.geo.ac.at (N.P.)

² College of Survey and Geoinformation, Tongji University, 200092 Shanghai, China

* Correspondence: liuchun@tongji.edu.cn

Academic Editor: name

Received: 22 October 2017; Accepted: 24 November 2017; Published: date

Abstract: The common statistical methods for supervised classification usually require a large amount of training data to achieve reasonable results, which is time consuming and inefficient. In many methods, only the features of each point are used, regardless of their spatial distribution within a certain neighborhood. This paper proposes a tensor-based sparse representation classification (TSRC) method for airborne LiDAR (Light Detection and Ranging) points. To keep features arranged in their spatial arrangement, each LiDAR point is represented as a 4th-order tensor. Then, TSRC is performed for point classification based on the 4th-order tensors. Firstly, a structured and discriminative dictionary set is learned by using only a few training samples. Subsequently, for classifying a new point, the sparse tensor is calculated based on the tensor OMP (Orthogonal Matching Pursuit) algorithm. The test tensor data is approximated by sub-dictionary set and its corresponding subset of sparse tensor for each class. The point label is determined by the minimal reconstruction residuals. Experiments are carried out on eight real LiDAR point clouds whose result shows that objects can be distinguished by TSRC successfully. The overall accuracy of all the datasets is beyond 80% by TSRC. TSRC also shows a good improvement on LiDAR points classification when compared with other common classifiers.

Keywords: tensor sparse coding; structured and discriminative dictionary learning; feature extraction

1. Introduction

LiDAR (Light Detection and Ranging) point cloud classification in urban areas has always been an essential and challenging task. Before classification, various features are extracted from the raw three-dimensional (3D) point cloud, which should be able to distinguish different objects. Due to the complexity in urban scenes, it is difficult to label objects correctly using only single or multi feature thresholds. Thus, research mainly focused on the use of statistical method for supervised classification of LiDAR points in recent years. Common machine learning methods include the support vector machine (SVM) algorithm, AdaBoost, decision trees, random forest, and other classifiers. Those machine learning methods aim to build a classification rule or probability function to determine the label based on the features. SVM seeks out the optimal hyperplane that efficiently separates the classes, and the Gaussian kernel function can be used to map non-linear decision boundaries to higher dimensions, where they are linear [1]. AdaBoost is a binary algorithm, but several extensions are explored for multiclass categorization. The weak hypothesis generation routines are combined into AdaBoost algorithm to classify terrain and non-terrain areas in [2]. Decision trees can be used to carry out the classification by training data and make a hierarchical binary tree model, new objects can be classified based on previous knowledge [3]. Random Forest is

an ensemble learning method that uses a group of decision trees, provides measures of feature importance for each class [4], and runs efficiently on large datasets.

However, those approaches barely consider the spatial distribution of points, which is an important cue for the classification in complex urban scenes. Some studies have applied graphical models to incorporate spatial context information in the classification. Graphical models take neighboring points into account, which allow for us to encode the spatial and semantic relationships between objects via a set of edges between nodes in a graph [5]. Markov network and conditional random field (CRF) are two mainstream methods to define the graphical model. However, a large amount of training data is necessary to obtain the classifier in those statistical studies. Anguelov et al. [6] use the Associated Markov Network (AMN) to classify objects on the ground. The study takes 1/6 points as training data and achieve an overall classification accuracy as 93%. Niemeyer et al. [7] use 3000 points per class to train the CRF model. Seven classes (grass land, road, tree, low vegetation, buildings with gable roof, buildings with flat roof, and facade) are distinguished based on the CRF and the overall accuracy is 83%, which is a fine result in a complex urban scene. As for statistical methods, Im [8] uses 316 training samples (1%) to generate decision trees with an overall accuracy of 92.5%. Moreover, Lodha uses half of dataset as training data through AdaBoost algorithm and the average accuracy is 92%. As a consequence, classifier training would be very time-consuming, especially when Markov network or CRF are used as classifiers.

In order to combine spatial distribution and feature information, we suggest using the high-dimensional tensor data structure for representing each point (to avoid misunderstandings we stress that tensor refers here to a high dimensional data structure, and is not related to the concepts of tensor voting or the structure tensor). Normally, the dimensional data has to be embedded into vectors in traditional methods. However, the vectorization breaks the original multidimensional structure of the signal and reduces the reliability of post processing. Therefore, high-dimensional tensors are utilized in several approaches. The high-dimensional tensor means that the elements in the data are to be addressed by more than two indices. Tensors have been widely applied to hyperspectral images, face images, and video data representation. Renard and Boourennane introduce a hyperspectral image representation based on tensors to jointly take advantage of the spatial and spectral information [9]. It shows that the spatial projection into a lower orthogonal subspace joint with spectral dimension reduction can efficiently improve the classification. In face recognition, the Tensorfaces are proposed by Vasilescu et al. to overcome the influence of different factors that are related to facial geometries, expressions, head poses, and lighting conditions [10]. The Tensorfaces improve the facial recognition rates when compared with the standard eigenfaces. Kuang et al. use a unified tensor model to represent the large-scale and heterogeneous data [11]. It shows a great ability of dimensionality reduction by using incremental high order singular value decomposition.

This paper aims to use as few training data as possible to achieve effective classification. Therefore, sparse representation-based classification is used in this paper. Sparse representation-based classification (SRC) is a well-known technique to represent data by sparse linear combination of bases, which are extracted from a fixed dictionary or learned dictionary. It classifies unknown data based on the reconstruction criteria. SRC has been successfully applied to the processing of signals [12] and images [13]. SRC includes two important parts: sparse coding and dictionary learning. Sparse coding is to find a certain small number of base atoms from the dictionary for reconstruction raw data. Sparse coding can be solved by Orthogonal Matching Pursuit (OMP) [14], LASSO (least absolute shrinkage and selection operator) [13], or the gradient descent algorithm [15]. The dictionary can generally come from two sources: mathematical model-based methods and the dictionary learning from training data. The mathematical model-based methods for building a dictionary include: Fourier series, wavelets and discrete cosine transform bases [16,17]. But, this predefined dictionary is fixed and cannot be adapted according to the dataset. Therefore, dictionary learning from the dataset is an optimal choice due to its flexibility for a specific dataset. The dictionary learning problem can be solved by the method of optimal directions (MOD) [18], K-SVD [19], and the gradient descent algorithm [20]. Furthermore, previous research formulates the high dimensional data SRC problem in terms of tensors. Tensor extensions of the dictionary learning

and sparse coding algorithms have been developed, such as Tensor MOD and KSVD for dictionary learning [21], and tensor OMP [22]. Moreover, tensor based sparse representation has yielded good performance in high-dimensional data classification [9], face recognition [23], and image de-noising [24].

In this paper, we propose a tensor-based sparse representation classification method for urban airborne LiDAR points identification. The main innovations of this paper are summarized below:

1. A new data structure is introduced to represent each point. To keep the feature description in their original geometrical 3D space, the LiDAR points are represented as 4th-order tensors. A point and its neighboring points are rearranged by their spatial distribution in the tensor space, meanwhile the features of each point in the neighborhood are also attached as the fourth mode of the tensor. In this tensor data structure, both spatial and feature information can be used for classification.
2. A structured and discriminative dictionary set is learned for tensors based on a few samples of training data. Firstly, we present a structured and discriminative dictionary learning adapted to the high dimensional tensor data. Additionally, the dictionary learning only uses a few samples of training data. The dictionary classifier shows better classification ability than other popular classifiers (KNN, decision tree, random forest, SVM) when using the same amount of training data.

Finally, the decision which class a point belong to is based on the minimum reconstruction residual from the sub-dictionary and its subset of sparse tensor. The sparse tensor approximation of each test tensor can be obtained by projecting the test tensor onto dictionaries, and the sparse tensor only has a few nonzero entries that are corresponding to the selected atoms in the dictionary set. We expect that the sparse tensors of points belong to the same class have similar structure. At last, the label of unknown points can be predicted by the minimum reconstruction residual from each class specific sub-dictionaries and the subset of the sparse tensor.

In the following, we first introduce the tensor generation processing in Section 2. Subsequently, the conventional sparse representation classification (SRC) is briefly introduced in Section 3.1. Then, the procedure of tensor-based sparse representation classification is written in detail in Section 3.2, which includes the sparse coding algorithm for tensor data (in Section 3.2.1), structured and discriminative dictionary learning (in Section 3.2.2), and the classification procedure (in Section 3.2.3). After that, the tensor-based sparse representation classification (TSRC) classification results and the comparison with other classifiers are presented in Section 4, followed by a discussion on the influence of parameters selection in Section 5. Finally, the major findings of this work are summarized in Section 6.

2. Tensor Representation of LiDAR Points

2.1. Tensor Notations and Preliminaries

A tensor is denoting a multidimensional object, whose elements are to be addressed by more than two indices. The order of a tensor, also known as modes [25], is the number of dimensions. Tensors are denoted as boldface italic capital letters, e.g., $\mathbf{T} \in \mathbf{R}^{I_1 \times I_2 \times \dots \times I_N}$; matrices are denoted as upright capital letters, e.g., $\mathbf{T} \in \mathbf{R}^{I_1 \times I_2}$; vectors are denoted as upright lowercase letters, e.g., $\mathbf{t} \in \mathbf{R}^I$. The element (i_1, i_2, \dots, i_N) of a tensor \mathbf{T} is expressed as t_{i_1, i_2, \dots, i_N} , where $1 \leq i_n \leq I_n$. The Frobenius norm of a tensor \mathbf{T} is defined as:

$$\|\mathbf{T}\|_F = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \dots \sum_{i_N=1}^{I_N} t_{i_1, i_2, \dots, i_N}^2}$$

The tensor can be transformed into a vector or matrix, and this processing is known as unfolding or flattening. Given an N^{th} -order tensor $\mathbf{T} \in \mathbf{R}^{I_1 \times I_2 \times \dots \times I_N}$, the n -mode unfolding vector of tensor \mathbf{T} is obtained by fixing every index except the one in the mode n [26]. The n -mode unfolding matrix is defined by arranging all of the n -mode vectors as columns of a matrix, i.e., the n -mode unfolding matrix $\mathbf{T}_{(n)} \in \mathbf{R}^{I_n \times I_1 \cdot I_2 \cdot \dots \cdot I_{n-1} \cdot I_{n+1} \cdot \dots \cdot I_N}$. The product between two matrices can be extended to

the product of a tensor and a matrix. The n -mode product of a tensor $\mathbf{T} \in \mathbf{R}^{I_1 \times I_2 \times \dots \times I_N}$ with a matrix $\mathbf{U} \in \mathbf{R}^{J_n \times I_n}$ is denoted by $\mathbf{Y} = \mathbf{T} \times_n \mathbf{U} \in \mathbf{R}^{I_1 \times I_2 \times \dots \times I_{n-1} \times J_n \times I_{n+1} \times \dots \times I_N}$. For the processing, this can be converted to the matrix product of \mathbf{U} and the unfolded tensor $\mathbf{T}_{(n)}$, which is expressed as Equation (1):

$$\mathbf{Y} = \mathbf{T} \times_n \mathbf{U} \Leftrightarrow \mathbf{Y}_{(n)} = \mathbf{U} \times \mathbf{T}_{(n)} \quad (1)$$

The Tucker decomposition is a form of higher-order principal component analysis, which can be described as Equation (2). It decomposes a tensor $\mathbf{T} \in \mathbf{R}^{I_1 \times I_2 \times \dots \times I_N}$ into a core tensor $\mathbf{X} \in \mathbf{R}^{J_1 \times J_2 \times \dots \times J_N}$ multiplied by the matrix $\mathbf{U}_1 \in \mathbf{R}^{J_1 \times I_1}$, $\mathbf{U}_2 \in \mathbf{R}^{J_2 \times I_2}$, \dots , $\mathbf{U}_N \in \mathbf{R}^{J_N \times I_N}$ along each mode. \mathbf{X} is the core tensor, and its entries show the level of interaction between the different components [25]. The matrix $\mathbf{U}_1, \mathbf{U}_2, \dots, \mathbf{U}_N$ can be considered as the principal components in each mode. If $J_1, J_2, \dots, J_N \leq I_1, I_2, \dots, I_N$ the core tensor \mathbf{X} can be considered as a compressed version of the original tensor. In the following equations, the \cong sign means “approximately equal”.

$$\mathbf{T} \cong \mathbf{X} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times \dots \times_N \mathbf{U}_N \quad (2)$$

Tucker mode can be written as the Kronecker representation, these two representations are equivalent. Let \otimes denote the Kronecker product, and define the vectorization operation on tensors as $\mathbf{t} = \text{vec}(\mathbf{T})$, $\mathbf{t} \in \mathbf{R}^{I_1 I_2 \dots I_N}$. The vectorization operation stacks all of the columns of the mode-1 tensor $\mathbf{T}_{(1)}$ in a single vector. Then, given $\mathbf{t} = \text{vec}(\mathbf{T})$, $\mathbf{x} = \text{vec}(\mathbf{X})$, the equivalent Kronecker representation is shown as following:

$$\mathbf{t} \cong (\mathbf{U}_1 \otimes \mathbf{U}_2 \otimes \dots \otimes \mathbf{U}_N) \cdot \mathbf{x} \quad (3)$$

2.2. Tensor Representation of LiDAR Point

The process of presenting a point p as the tensor \mathbf{T} is described, as seen in Figure 1. Firstly, the k closest neighbors of the point p are found and denoted as point set P . Then global Cartesian coordinates of point set P are transformed to the local PCA (Principal Component Analysis) coordinate system. The local axes ($\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$) are defined by the principal direction of variance of the point set P based on PCA. The PCA transformation ensures that the first axis \mathbf{e}_1 has the most variation, the second axis \mathbf{e}_2 has the second-most, and the third axis \mathbf{e}_3 the least. Therefore, the spatial distribution is reflected by the coordinate values in the third axis. Points of volumetric structure will have various coordinates in the third axis in the local coordinate system, whereas points of local planar surfaces will have consistent coordinates in the third axis. Let $p_i (x_i, y_i, z_i)$ be the point in global Cartesian coordinate system, $p_{ei} (u_i, v_i, w_i)$ be the point in local PCA coordinate system. The transformation is calculated by:

$$\begin{aligned} u_i &= \mathbf{e}_1(x_i - x_0) \\ v_i &= \mathbf{e}_2(y_i - y_0) \\ w_i &= \mathbf{e}_3(z_i - z_0) \end{aligned} \quad (4)$$

where (x_0, y_0, z_0) are the mean coordinates of points within the neighborhood in the global coordinate system.

After that, all points in the local PCA coordinate system are converted into voxel coordinates by the following equations:

$$\begin{aligned} v_i^x &= \text{Int} \left(\frac{u_i - \min(u)}{\Delta v^x} \right) + 1 \\ v_i^y &= \text{Int} \left(\frac{v_i - \min(v)}{\Delta v^y} \right) + 1 \\ v_i^z &= \text{Int} \left(\frac{w_i - \min(w)}{\Delta v^z} \right) + 1 \end{aligned} \quad (5)$$

where (v_i^x, v_i^y, v_i^z) represents the voxel index within the voxel array, $\text{Int}()$ is the function that rounds off the result to the nearest integer, $(\min(u_i), \min(v_i), \min(w_i))$ is the minimum values of (u, v, w)

and $(\Delta v^x, \Delta v^y, \Delta v^z)$ indicates the voxel element size. In this paper, $\Delta v^x = \Delta v^y = \Delta v^z = 0.2$ covering a cubic space of 1 m^3 , which means $v_i^x \in [1,5]; v_i^y \in [1,5]; v_i^z \in [1,5]$, a unique natural number ranging from 1 to 5 can be associated to each voxel in X, Y, Z dimensions. Subsequently, the mean feature vector of all the points that is assigned to each voxel is set as the voxel value, which is shown as in Figure 1. As a result, the center point p with its k nearest neighborhood is represented as a 4th-order tensor, the entries are accessed by the voxel index and the feature number. Based on this data structure, the attribute of each point are regarded as entries in the tensor, which are arranged as $r_{i_1 i_2 i_3 i_4}$, where $i_1 = 1, \dots, I_1; i_2 = 1, \dots, I_2; i_3 = 1, \dots, I_3; i_4 = 1, \dots, I_4$, $I_1 = I_2 = I_3 = 5$ and I_4 equals to the number of attribute on each points in our approach. Finally, the point p is described as the 4th-order tensor $T \in \mathbf{R}^{I_1 \times I_2 \times I_3 \times I_4}$, where I_1, I_2, I_3, I_4 indicate the X, Y, Z and attribute mode, respectively. In this regard, the spatial distribution and attributions can be simultaneously preserved. Each point in the LiDAR dataset is processed as the 4th-order tensor, which is used for tensor-based dictionary learning and sparse coding.

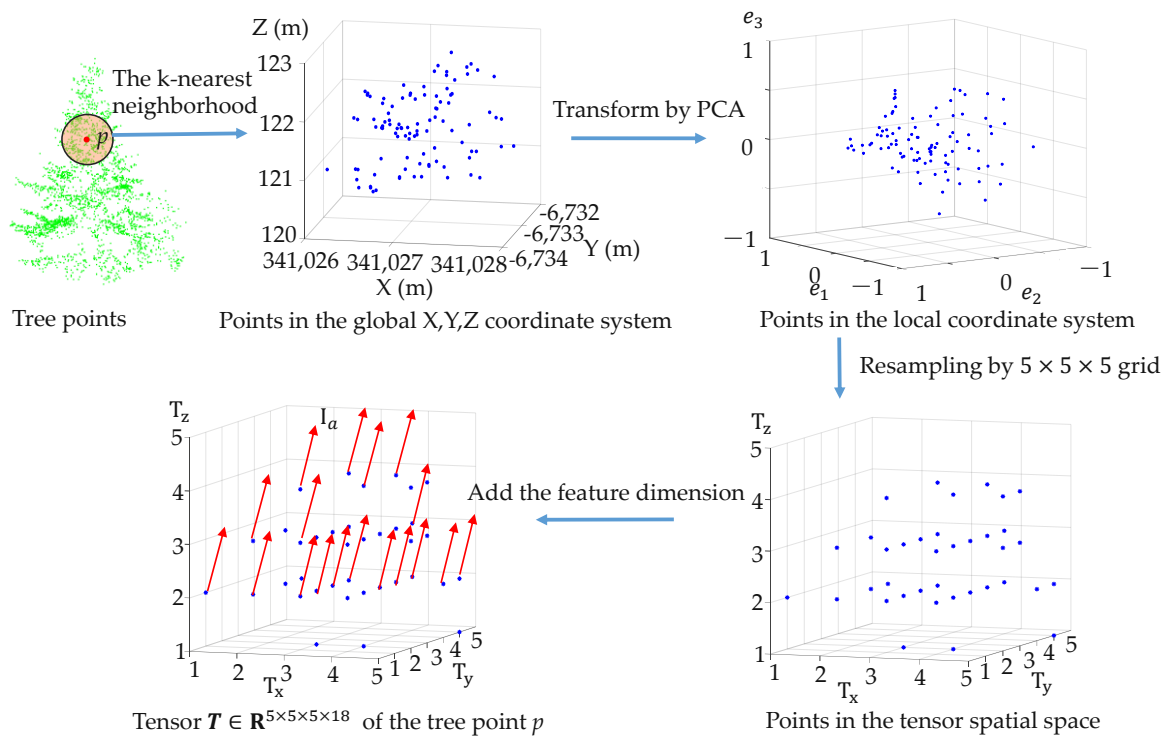


Figure 1. The tensor generation from a point cloud set procedure.

3. Tensor-Based Sparse Representation Classification Methodology

3.1. Sparse Representation Classification Model

The sparsity algorithm is to find the best representative of a test sample by sparse linear combination of training samples from a dictionary [13]. Given a certain number of training samples from each class, the sub-dictionary D^i from i th class is learned. Assume that there are c classes of subjects, and let $D = [D^1, D^2, \dots, D^c]$, which is the overall structured dictionary over the entire dataset. Denote by y a test sample vector and x the sparse coefficient vector of y , the linear representation of y can be written as:

$$y = Dx \tag{6}$$

The sparse coefficient vector x is calculated by projecting y on the dictionary D , which is called sparse coding procedure. The sparse coefficient vector x can be obtained by solving the following optimization problem:

$$x = \arg \min_x \|x\|_0 \quad s. t. \|y - Dx\|_2 \leq \varepsilon \quad (7)$$

where $\|\cdot\|_0$ is the l_0 -norm of vector x which defines the number of nonzero elements in x and ε is a pre-specified residual level parameter. The problem in (7) is a nondeterministic polynomial-time hard (NP-hard problem) due to the non-differentiability and non-convex nature of the l_0 -norm. Typical approaches for solving (7) are either approximation of the original problem with l_1 -norm based convex relaxation [27], or resorting to greedy schemes, such as match pursuit and basis pursuit algorithms [28]. The optimization of (7) can also be reformulated as:

$$x = \arg \min_x \|y - Dx\|_2 \quad s. t. \|x\|_0 \leq s \quad (8)$$

where s is the sparsity level of vector x . By the additional constraint $\|x\|_0 \leq s$, the sparse vector x for the test sample y on the dictionary D can be obtained. Based on the class information of structured dictionary D , the sparse coefficient vector x can be written as $x = [x^1, x^2, \dots, x^c]$, where x^i is the subset of the sparse coefficient vector x associated with class i . Thus, x should be a sparse coefficient vector whose entries is zero except those corresponding to the i th class. According to this assumption, the test sample y^i from class i can be well represented by a linear combination of the sub-dictionary D^i and its corresponding subset of sparse vector x^i .

Sparse representation classification (SRC) uses the reconstruction error e_i that is associated with each class to perform the data classification. First of all, the sparse representation x of test sample y is recovered with respect to the whole dictionary. Then, x^i is extracted from x as the subset vector corresponding to the class i . The test sample is reconstructed by each class specific sub-dictionary D^i and its corresponding sparse vector x^i . The class label of y is then determined as the one with minimal residual.

$$e_i = \|y - D^i x^i\|_2 \quad \text{class } i = [1, 2, \dots, c] \quad (9)$$

$$\text{identify}(y) = \arg \min_i \{e_i\}$$

3.2. Tensor-Based Sparse Representation Classification

When considering the 4th-order tensors that are used in this work, the dictionary set (D_1, D_2, D_3, D_4) is required to be learned on X, Y, Z , and attribute modes. The sparse coefficient vector x is also extended to a 4th-order sparse tensor \mathbf{X} . After the tensor generation for each point, the 4th-order tensors are used as the input data. At the beginning, the training tensor samples are randomly selected for the dictionary set learning. The dictionary is also composed of several sub-dictionaries that are associated with class i . Subsequently, for the sparse coding, the test tensor is projected into dictionaries on each mode to achieve the sparse tensor. This sparse coding is solved by TOMP (Tensor-based Orthogonal Matching Pursuit). Then, the test tensor data is recovered by the class specific sub-dictionaries and their corresponding subset of the sparse tensor. Finally, the label of the test tensor is predicted by the minimal reconstruction errors. The whole procedure is shown in Figure 2.

We use an alternating strategy to solve the dictionary learning problem. It can be divided into two sub-problems: updating the sparse tensor \mathbf{X} by fixing the dictionary set (D_1, D_2, D_3, D_4) , and updating the dictionary set (D_1, D_2, D_3, D_4) by fixing the sparse tensor \mathbf{X} , until convergence. As a result, the desired dictionary set (D_1, D_2, D_3, D_4) and the sparse tensor \mathbf{X} can be obtained.

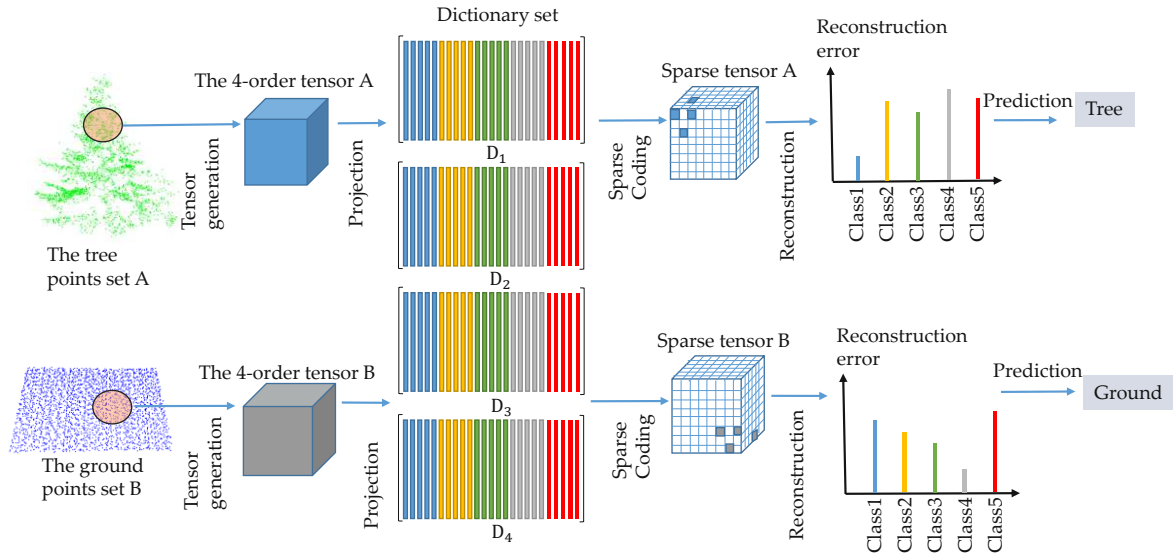


Figure 2. Tensor-based sparse representation classification procedure.

3.2.1. Tensor-Based Sparse Coding

To calculate the sparse core tensor \mathbf{X} , we use a greedy algorithm TOMP (Tensor-based Orthogonal Matching Pursuit) proposed by [22]. Classical OMP locates the support of the sparse vector that have the best approximation of sample data from the dictionary. It selects the support set by one index at each iteration until s atoms are selected or the approximation error is within a preset threshold [14], where s is the sparsity.

Given a fourth-order LiDAR point tensor $\mathbf{T} \in \mathbf{R}^{I_1 \times I_2 \times I_3 \times I_4}$, suppose that the dictionary set (D_1, D_2, D_3, D_4) is fixed, $D_1 \in \mathbf{R}^{I_1 \times J_1}$, $D_2 \in \mathbf{R}^{I_2 \times J_2}$, $D_3 \in \mathbf{R}^{I_3 \times J_3}$, $D_4 \in \mathbf{R}^{I_4 \times J_4}$, $\mathbf{X} \in \mathbf{R}^{J_1 \times J_2 \times J_3 \times J_4}$ is the sparse tensor of \mathbf{T} to be calculated. The objective function is converted to a sparse coding problem with l_0 -norm regularization which can be written as:

$$\begin{aligned} \min_{\mathbf{X}_k} \|\mathbf{T} - \mathbf{X} \times_1 D_1 \times_2 D_2 \times_3 D_3 \times_4 D_4\|_F \\ \text{s.t. } x_{j_1, j_2, j_3, j_4} = 0 \quad \forall (j_1, j_2, j_3, j_4) \notin \Gamma_1 \times \Gamma_2 \times \Gamma_3 \times \Gamma_4 \end{aligned} \quad (10)$$

where $\|\cdot\|_F$ is the Frobenius norm, $\Gamma_n = [j_n^1, j_n^2, \dots, j_n^{s_n}]$ is the subset of s_n indices of non-zero values in the sparse core tensor on mode X, Y, Z and attribute, and thus, denotes all possible combination of s_n non-zero indices on the four modes. Therefore, the cross product $\Gamma_1 \times \Gamma_2 \times \Gamma_3 \times \Gamma_4$ is the set of all the possible non-zero indices that can appear. Moreover, s_1, s_2, s_3, s_4 represents the X, Y, Z and attribute mode sparsity, indicating the number of selected column of each dictionary for the sparse representation. The total sparsity of the fourth-order core sparse tensor is denoted by $s = s_1 \times s_2 \times s_3 \times s_4$. Due to the overcomplete dictionary set, the size of the sparse tensor \mathbf{X} is larger than the LiDAR tensors \mathbf{T} .

Tensor-based Orthogonal Matching Pursuit (TOMP) relies on the equivalence of Tucker model and its Kronecker representation. Given $\mathbf{t} = \text{vec}(\mathbf{T})$, $\mathbf{x} = \text{vec}(\mathbf{X})$, the following two representations are equivalent:

$$\begin{aligned} \mathbf{T} &\cong \mathbf{X} \times_1 D_1 \times_2 D_2 \times_3 D_3 \times_4 D_4 \\ \mathbf{t} &\cong (D_4 \otimes D_3 \otimes D_2 \otimes D_1) \cdot \mathbf{x} \end{aligned} \quad (11)$$

where \otimes is the Kronecker product. Equation (11) is similar to the conventional linear sparse representation formulation. Based on this equivalence, if the vectorized version \mathbf{t} admits a s -sparse representation over the Kronecker dictionary $D_{\text{Kron}} = (D_4 \otimes D_3 \otimes D_2 \otimes D_1)$, then the 4th-order tensor $\mathbf{T} \in \mathbf{R}^{I_1 \times I_2 \times I_3 \times I_4}$ also has a sparse representation with respect to the dictionaries D_1, D_2, D_3, D_4 on each mode. In the standard Tucker model, the core tensor usually has smaller size than the data tensor and the main objective is to find such a decomposition, which is to compute both the core

tensor and the factor matrices. In our approach, the data tensor and dictionaries are known and the objective is to calculate the core tensor \mathbf{X} that can approximately recover the input tensor. Additionally, the core tensor \mathbf{X} is sparse and its size is larger than the data tensor. The TOMP algorithm is given in Table 1.

Table 1. The algorithm for TOMP (Tensor-based Orthogonal Matching Pursuit).

Algorithm: Tensor OMP
Require: input point tensor $\mathbf{T} \in R^{I_1 \times I_2 \times I_3 \times I_4}$, Dictionaries $D_1 \in R^{I_1 \times J_1}$, $D_2 \in R^{I_2 \times J_2}$, $D_3 \in R^{I_3 \times J_3}$, $D_4 \in R^{I_4 \times J_4}$, maximum number of non-zeros coefficients s_n in each mode.
Output: sparse tensor $\mathbf{X} \in R^{J_1 \times J_2 \times J_3 \times J_4}$, non-zeros coefficients index in sparse tensor $(\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4)$
Step:
1, initial: $\Gamma_n = [\emptyset] (n = 1, 2, 3, 4)$, Residual $\mathbf{R} = \mathbf{T}$, $\mathbf{X} = 0$, $k = 0$, $t = \text{vec}(\mathbf{T})$
2, while $\ \Gamma_n\ _0 \leq s$ do
3, $[j_1, j_2, j_3, j_4] = \arg \max_{[j_1, j_2, j_3, j_4]} \mathbf{R} \times_1 D_1^T(:, j_1) \times_2 D_2^T(:, j_2) \times_3 D_3^T(:, j_3) \times_4 D_4^T(:, j_4) $
4, $\Gamma_n = \Gamma_n \cup [j_1, j_2, j_3, j_4] (n = 1, 2, 3, 4)$. $TD_1 = D_1(:, \Gamma_1)$, $TD_2 = D_2(:, \Gamma_2)$, $TD_3 = D_3(:, \Gamma_3)$, $TD_4 = D_4(:, \Gamma_4)$;
5, $x = \arg \min_u \ (TD_1 \otimes TD_2 \otimes TD_3 \otimes TD_4)u - t \ _2^2$;
6, $\mathbf{X} = \text{tensorize}(x)$;
7 $\mathbf{R} = \mathbf{T} - \mathbf{X} \times_1 TD_1 \times_2 TD_2 \times_3 TD_3 \times_4 TD_4$;
8, $t = t + 1$;
9, end while
10, return \mathbf{X} , $(\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4)$.

3.2.2. Structured and Discriminative Dictionary Learning

Dictionary learning aims to build a dictionary that is composed of basis vectors, which can fully represent test samples by the sparse coding procedure. Regarding the 4th-order tensor data, the dictionary set (D_1, D_2, D_3, D_4) on X, Y, Z , attribute mode should be learned. To improve the performance of the dictionary learning method, a structured and discriminative dictionary is estimated in our approach. Instead of learning a shared dictionary over all the classes, we derive a structured dictionary $D_1 = [D_1^1, D_2^1, \dots, D_c^1]$; $D_2 = [D_1^2, D_2^2, \dots, D_c^2]$; $D_3 = [D_1^3, D_2^3, \dots, D_c^3]$; $D_4 = [D_1^4, D_2^4, \dots, D_c^4]$, where $D_1^i, D_2^i, D_3^i, D_4^i$ is the class specified sub-dictionary associated with class i on X, Y, Z , and attribute mode, and c is the total number of classes. With such a dictionary set, we can use the reconstruction error for classification based on SRC.

Denote by $\mathbf{T} = [\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_c]$ the set of training point tensors, where \mathbf{T}_i is the subset of the training tensor samples from class i . Correspondingly, \mathbf{X}_i is the sparse tensor of \mathbf{T}_i over the entire dictionary set (D_1, D_2, D_3, D_4) . Furthermore, \mathbf{X}_i can be represented as $\mathbf{X}_i = [\mathbf{X}_i^1, \mathbf{X}_i^2, \dots, \mathbf{X}_i^j, \dots, \mathbf{X}_i^c]$, where \mathbf{X}_i^j is the subset of sparse tensors corresponding to the class specific dictionary $D_1^i, D_2^i, D_3^i, D_4^i$.

The initial dictionaries are composed of k leading principal vectors of matrices along each mode by Tucker decomposition. Denote by \mathbf{T}_{ij} the j th tensor from class i , \mathbf{T}_{ij} is tucker decomposed to get the U_1, U_2, U_3, U_4 , as shown in Equation (12), then the first k number of basis vectors of U_1, U_2, U_3, U_4 are added into dictionaries $D_1^i, D_2^i, D_3^i, D_4^i$. Then, the initial dictionary set is optimized by the discriminative dictionary learning model.

$$\mathbf{T}_{ij} \cong \mathbf{X}_{ij} \times_1 U_1 \times_2 U_2 \times_3 U_3 \times_4 U_4 \quad (12)$$

Besides requiring (D_1, D_2, D_3, D_4) should have strong reconstruction ability of for each tensor, the dictionary set should also own the powerful capability to distinguish tensor samples between various classes. Consequently, the discriminative fidelity terms are added to the dictionary learning model. Firstly, the dictionary set (D_1, D_2, D_3, D_4) should be able to well recover the training tensor set \mathbf{T} , therefore, $\mathbf{T}_i \cong \mathbf{X}_i \times_1 D_1 \times_2 D_2 \times_3 D_3 \times_4 D_4$. Then, since $D_1^i, D_2^i, D_3^i, D_4^i$ correspond to the class i , \mathbf{T}_i is expected to be well recovered by $D_1^i, D_2^i, D_3^i, D_4^i$, but not by $D_1^j, D_2^j, D_3^j, D_4^j, j \neq i$. This indicates that \mathbf{X}_i^i should have some significant entries, such that $\mathbf{T}_i \cong \mathbf{X}_i^i \times_1 D_1^i \times_2 D_2^i \times_3 D_3^i \times_4 D_4^i$, meanwhile, the entries in \mathbf{X}_i^j should be nearly zero, such that $\|\mathbf{X}_i^j \times_1 D_1^j \times_2 D_2^j \times_3 D_3^j \times_4 D_4^j\|_F$ is small. As a result, the dictionary learning model with the discriminative fidelity terms is defined as:

$$\underset{D_1, D_2, D_3, D_4}{\operatorname{argmin}} \sum_{i=1}^c (\|T_i - X_i \times_1 D_1 \times_2 D_2 \times_3 D_3 \times_4 D_4\|_F + \|T_i - X_i^i \times_1 D_1^i \times_2 D_2^i \times_3 D_3^i \times_4 D_4^i\|_F + \sum_{j=1, j \neq i}^c \|X_j^j \times_1 D_1^j \times_2 D_2^j \times_3 D_3^j \times_4 D_4^j\|_F) \text{ s.t. } \|X\|_0 \leq s \quad (13)$$

Again, c is the number of classes, and the $\|X\|_0 \leq s$ is the sparsity constraint, which means that the sparsity of tensor X is s .

These discriminative tensor dictionaries are learned in an alternating minimization rule, all other dictionaries and the sparse core tensors are fixed when learning one certain mode dictionary. Namely, D_1, D_2, D_3, D_4 are learned independently between each other. To learn the dictionary on a certain mode, we update the sub-dictionary D^i class by class. When updating D^i , all of the other sub-dictionary $D^j, j \neq i$ are fixed. Then, the objective function can be written as:

$$\underset{D_1^i, D_2^i, D_3^i, D_4^i}{\operatorname{argmin}} (\|T - X^i \times_1 D_1^i \times_2 D_2^i \times_3 D_3^i \times_4 D_4^i - \sum_{j=1, j \neq i}^c X^j \times_1 D_1^j \times_2 D_2^j \times_3 D_3^j \times_4 D_4^j\|_F + \|T_i - X_i^i \times_1 D_1^i \times_2 D_2^i \times_3 D_3^i \times_4 D_4^i\|_F + \sum_{j=1, j \neq i}^c \|X_j^j \times_1 D_1^j \times_2 D_2^j \times_3 D_3^j \times_4 D_4^j\|_F) \text{ s.t. } \|X\|_0 \leq s \quad (14)$$

Mathematically, the tensor equation can be represented in an unfolded form, the following two equations are equivalent:

$$T \cong X \times_1 D_1 \times_2 D_2 \times_3 \dots \times_n D_n \dots \times_N D_N \quad (15)$$

$$T_{(n)} \cong D_n X_{(n)} (D_N \otimes \dots \otimes D_{n+1} \otimes D_{n-1} \otimes \dots \otimes D_1)^T$$

where $T_{(n)}$ is the mode- n unfolding matrix of the tensor T , $X_{(n)}$ is the mode- n unfolded matrix of the tensor X and $n \in [1, 2, 3, 4]$. Let $\widetilde{D}_{-n} = (D_N \otimes \dots \otimes D_{n+1} \otimes D_{n-1} \otimes \dots \otimes D_1)$, Equation (14) can be rewritten into its unfolded version as:

$$\underset{D_1^i, D_2^i, D_3^i, D_4^i}{\operatorname{argmin}} (\|T_{(n)} - D_n^i X_{(n)}^i \widetilde{D}_{-n}^i - \sum_{j=1, j \neq i}^c D_n^j X_{(n)}^j \widetilde{D}_{-n}^j\|_F + \|T_{i(n)} - D_n^i X_{i(n)}^i \widetilde{D}_{-n}^i\|_F + \sum_{j=1, j \neq i}^c \|D_n^j X_{j(n)}^j \widetilde{D}_{-n}^j\|_F) \text{ s.t. } \|X\|_0 \leq s \quad (16)$$

This is a conversion to a constrained convex quadratic optimization problem, and it can be solved by the gradient algorithm in the paper [29]. Figure 3 shows the residuals of objective function (13) along dictionary set (D_1, D_2, D_3, D_4) updating.

In this way, dictionaries on the four modes are updated. In the next iteration, these new learned dictionaries are used to obtain the new sparse tensor in the sparse coding procedure. As a result, this dictionary learning processing alternates between tensor dictionary learning and sparse tensor update until a stopping criterion is reached.

Residuals

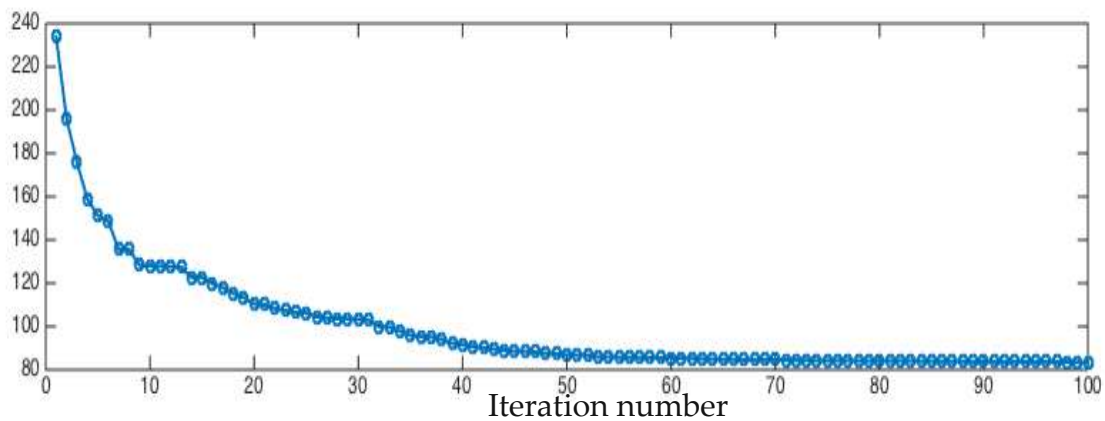


Figure 3. The reconstruction residuals of dictionary set of each iteration.

3.2.3. Tensor-Based Sparse Representation Classifier

Analogous to SRC, the class label of the test tensor \mathbf{T}_k is determined by the minimal residual:

$$label = \arg \min_{i=1,2,\dots,c} e_i \quad c = \text{number of classes}$$

where $e_i = \|\mathbf{T}_k - \mathbf{X}_k^i \times_1 D_1^i \times_2 D_2^i \times_3 D_3^i \times_4 D_4^i\|_F$.

Given a test tensor \mathbf{T}_k , its sparse tensor \mathbf{X}_k over the whole dictionary set (D_1, D_2, D_3, D_4) is calculated by TOMP, then \mathbf{X}_k^i is the subset of sparse tensor corresponding to the $D_1^i, D_2^i, D_3^i, D_4^i$ that is associated with class i . The test tensor should be well recovered by its corresponding sub-dictionary set and subset of sparse tensor, whereas the residual should be large when using other sub-dictionary sets and subsets of sparse tensor.

4. Results

4.1. Data Description

We perform the classification on eight real airborne LiDAR datasets of Vienna city. The area of each dataset is $100 \times 100 \text{ m}^2$. The densities of datasets mostly range from 8 to 75 points/ m^2 . Multiple echoes were recorded and the point clouds in all of the datasets are fully labeled. The datasets contain various kinds of objects, such as: high-rising buildings with balcony, small detached houses, single trees, grouped and low vegetation, ground with consistent height, and ground with slopes. In the classification procedure, the objects are categorized into five classes: *open ground* which is uncovered or not blocked by any objects; *building roof*; *vegetation*; *covered ground*, which is usually under the high trees or building roof; and, *façade*.

4.2. Feature Extraction

A set of 18 features are extracted from 3D LiDAR points, which contains height-based features, local plane-based features, penetrability-based features and local shape-based features. The four feature groups are detailed hereby.

4.2.1. Height-Based Features

1. Height difference. Height difference is measured between the LiDAR point and the lowest point found in a multiple scale cylindrical neighborhood. By varying the size of the local cylindrical neighborhood, height differences are calculated for each scale. The cylinder radii have been set experimentally to 10 m and 2 m, and correspondingly the height differences are denoted by ΔH_{r1} and ΔH_{r2} . The height difference ΔH is given by:

$$\Delta H = \begin{cases} \Delta H_{r1}, & \Delta H_{r1} \geq \lambda \\ \Delta H_{r2}, & \Delta H_{r1} < \lambda \end{cases}$$

The threshold $\lambda = 70\% \times \max(\Delta H_{r1})$, and the maximum is taken over the ΔH_{r1} of all the points. If the height difference value that is found in the large neighborhood is higher than the threshold, then this is considered as the reliable height difference value for this object. Otherwise, the objects may be points located on the slope, and the height difference should be calculated in a smaller neighborhood. Normally, the ground point should be selected as the lowest point and have a low height difference value. However, in the area of inclined ground, the ground points on the slope would also have relatively high height difference values for a large neighborhood selection, which can be seen in the rectangular area, as marked in Figure 4a. Figure 4a indicates that sloped ground areas have the same height difference values with roof points, which will lead to misclassification. Therefore, a small neighborhood is more suitable for sloped areas. The height difference values in the rectangular area marked in Figure 4b is much more reasonable using the multiple neighborhood selection. The ground area in the rectangle in Figure 4b shows a constant height difference values with most ground points. Thus, the height difference can be calculated correctly in both sloped and flat environments by using multiple neighborhood selections.

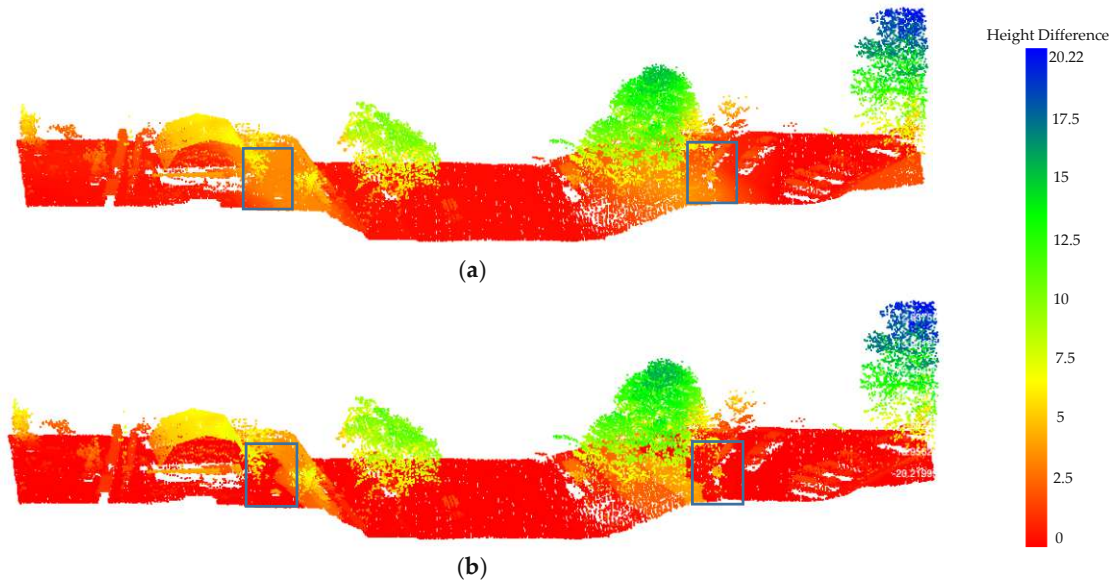


Figure 4. Height difference feature results: (a) Height difference via constant scale neighborhood; (b) Height difference via multiple scale neighborhood.

4.2.2. Local Plane-Based Features

For a given 3D point set and its k closest neighbors, the local plane-based features are exploited by estimating a local orthogonal regression plane. The local plane-based features contain:

- 2–4. Normal vector: Normal X; Normal Y; and, Normal Z. The normal vectors of local planes are estimated by k neighbor points, normal X; normal Y; normal Z are the values in X, Y, Z direction from the normal vectors.
5. NormalSigma0: the standard deviation of normal estimation. The value is high in rough areas and low in smooth areas.
6. NormalZSigma0: the standard deviation of Normal Z estimation in a cylindrical neighborhood. The value can reflect the penetrability of the object.
7. Normal planeoffset: the offset between the current point and its local estimated plane.
- 8–10. Eigenvalues: Eigenvalue1; Eigenvalue2; and, Eigenvalue3. The covariance matrix used for the normal vector computation is decomposed by eigenvalue analysis. This yields Eigenvalue1 λ_1 ; Eigenvalue2 λ_2 ; Eigenvalue3 λ_3 ($\lambda_1 > \lambda_2 > \lambda_3$). λ_2 λ_3 have low values for planar object and higher values for voluminous point clouds.

4.2.3. Echo-Based Features

11. Echo Ratio: The ER (echo ratio) is a measure for local transparency and roughness. It is defined as follows [30].

$$ER = n_{3D}/n_{2D} \times 100$$

with $n_{3D} \leq n_{2D}$, n_{3D} is the number of neighbors found in a certain search distance measured in 3D and n_{2D} is the number of neighbors found in the same distance measured in two-dimensions (2D). The ER is nearly 100% for a flat surface, whereas the ER decreases for penetrable surface parts since there are more points in a vertical search cylinder than there are points in a sphere with the same radius.

12. Echo number ratio. The echo number ratio of each point is defined as:

$$\text{Echo number ratio} = \frac{\text{echo number}}{\text{number of echoes}} \times 100$$

The echo number is q -th echo for a certain pulse. The number of echo is the maximum number of echoes that are detected for the pulse to which the echo belongs. The echo number ratio could indicate the penetrability of objects.

4.2.4. Local Shape-Based Features

The local shape-based features are obtained by the normalized eigenvalues λ_i , which include: linearity, planarity, sphericity, anisotropy, omivariance, and eigenentropy. The local shape-based features are calculated based on the paper by Niemeyer et al. [7], and are defined as follows:

$$13-18. \quad \text{Linearity} = \frac{\lambda_1 - \lambda_2}{\lambda_1}; \text{Planarity} = \frac{\lambda_2 - \lambda_3}{\lambda_1}; \text{Sphericity} = \lambda_3 / \lambda_1$$

$$\text{Anisotropy} = \frac{\lambda_1 - \lambda_3}{\lambda_1}; \text{Omivariance} = \sqrt[3]{\lambda_1 \lambda_2 \lambda_3}; \text{Eigenentropy} = - \sum_{i=1}^3 \lambda_i \ln \lambda_i$$

The nearest neighbors are selected for feature extraction, and k_f is set to 30. The radius for ER and NormalZSigma0 calculation is set to 1 m. After the feature extraction, all feature values are normalized to the interval [0,1]. Then, a feature vector of 18 dimensions corresponding to each point is obtained through the feature extraction, and attached as the 4th-order of the tensor data. Figure 5 shows a selection of features extraction results of Dataset 3.

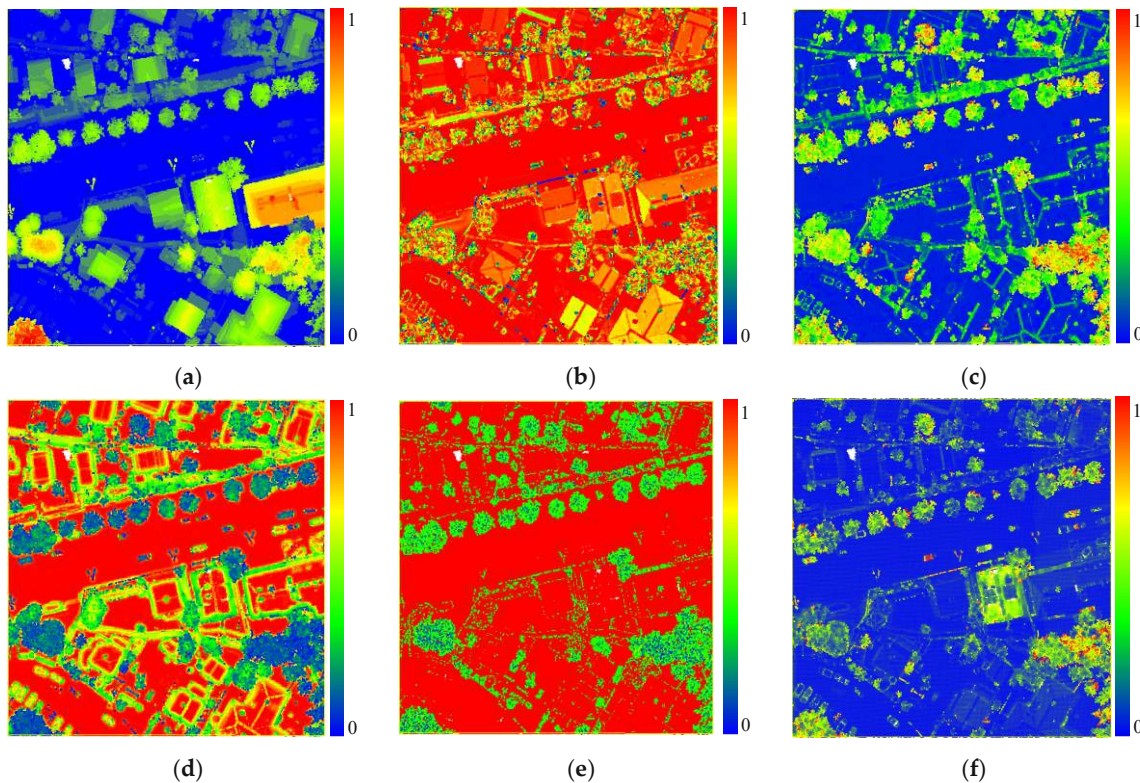


Figure 5. The feature extraction results of Dataset 3: (a) Height difference; (b) NormalZ; (c) NormalSigma0; (d) Echo Ratio; (e) Echo number ratio; and, (f) Eigenentropy.

4.3. Classification Results

Based on the 18 features described in Section 4.2, we conduct a series of experiments for TSRC. Firstly, the general behavior of TSRC is analyzed in Section 4.3.1, then, KNN (k -nearest neighbors), DT (Decision Tree), RF (Random Forest), SVM (Support Vector Machine) are used for comparison in Section 4.3.2. In each experiment, the training data is randomly selected from the whole dataset, and the remaining dataset is used as the test samples. For each class, always the same number of training samples is selected. The overall accuracy (OA) is selected to evaluate all of the classifiers.

4.3.1. Tensor-Based Sparse Representation Classification Results and Discussion

For TSRC, all 3D LiDAR points are generated as the fourth-order tensor $T \in \mathbf{R}^{5 \times 5 \times 5 \times 18}$, which means that the points are sampled into $5 \times 5 \times 5$ regular grids in the three-dimensional space, and each grid is attached with a 1×18 feature vector. The sparsity level is set to 9, and 27 training sample tensors are randomly selected from each class to learn the dictionary.

We conduct the TSRC on the eight real airborne LiDAR datasets. To avoid the biased result, we repeated TSRC 10 times on each dataset. Visual inspection indicates that most objects are classified correctly in Figure 6. The unlabeled points are objects that do not belong to any class mentioned in the Section 4.1, such as fences, cars, power lines, and others. Unlabeled points are not involved in the accuracy evaluation. The amount of points in each LiDAR dataset, percentage of training data, and mean OA of 10 classification experiments and the standard deviation of OA are summarized in Table 2. The overall accuracies of all the datasets are beyond 80%, which are rather good classification results when considering that only a few training samples are used. Moreover, the OA deviations of all datasets are less than 1%, and OA values of 10 TSRC experiments remain stable for all of the datasets. It indicates that the TSRC is barely affected by the training data selection.

Table 2. The percentage of training samples for all of the datasets.

Dataset	Test Set	Training Set	Mean OA	OA std.dev	Data Set	Test Set	Training Set	Mean OA	OA std.dev
Dataset 1	665,466	0.02%	90.57%	0.77%	Dataset 5	365,926	0.03%	82.16%	0.69%
Dataset 2	452,800	0.02%	91.85%	0.84%	Dataset 6	222,702	0.06%	87.03%	0.71%
Dataset 3	352,318	0.03%	84.98%	0.63%	Dataset 7	880,809	0.02%	85.09%	0.91%
Dataset 4	298,201	0.04%	88.44%	0.89%	Dataset 8	320,716	0.04%	87.24%	0.64%

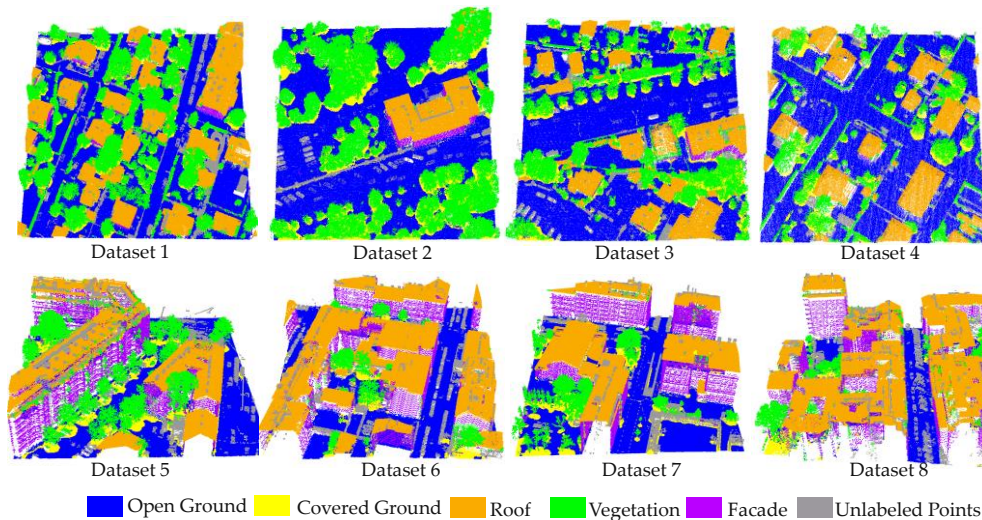


Figure 6. Three-dimensional view of the classification results of eight datasets using TSRC.

The confusion matrices in Table 3 present the correctness and incorrectness for each class of all the datasets. From the confusion matrices and qualification (Figure 7) of the classification, we can see that the major confusions occur between open ground and covered ground. 14.44% of open ground points are mislabeled as covered ground in Dataset 6, and 8.65% of covered ground points are wrongly labeled as ground in Dataset 3. This is caused by the same attributes that open and covered ground points share, such as same height difference, roughness, and local shape parameters. Moreover, open and covered ground points are easily mixed in the neighborhood when generating the tensor. Based on the Table 3, there are 4.45% of open ground points that are wrongly labeled as roof in Dataset 3. Some slope areas and low roofs are confused with each other in this site. This is due to the same feature values and geometry that they have. However, open and covered ground points are scarcely classified to other classes for other datasets. Therefore, the ground points are well distinguished from other objects by TSRC, which shows great potential ability for ground filtering. As for roof classification result, incorrect points are found essentially on building edges (as seen in Figure 7). They are labeled as vegetation, since such points behave similar for many attributes, such

as the low ER values, high NormalSigma0 values, and low planarity. Vegetation are well classified with a high accuracy. The error points do not appear on certain classes, they randomly occur in the other four classes based on the statistic in Table 3. Finally, the accuracy of façade is relatively low. Large numbers of points are labeled as vegetation. They mainly correspond to the façade where points are not sufficiently dense and co-planar. Those points will have high local-planar based feature values, which behave similarly to vegetation points. Furthermore, some façade points are very close to roof and ground, and they are often misclassified due to the neighborhood selection used for tensor generation.

In total, TSRC can lead to a good classification result on the airborne urban LiDAR points with a few training data only. Based on the statistic of the number of points per class, the accuracy has no direct relevance to the amount of training data. With 27 training samples given in each class, the performance remains stable no matter whether dealing with classes with a large or a small amount of points. Finally, when compared with the object points, the ground points are most likely to be correctly detected by TSRC, which is meaningful for the filtering and generation of DTM (Digital Terrain Model).

Table 3. The confusion matrices for each dataset and number of points per class in each dataset. (Error rates above 3% are highlighted by italic type except the mixtures between open ground and covered ground).

Predicted Class	Dataset	Reference Class					Total Points	Training Points
		Open Ground	Vegetation	Roof	Covered Ground	Facade		
Open ground	Dataset 1	89.34%	0.87%	0.97%	8.59%	0.13%	255,835	27
	Dataset 2	90.07%	0.04%	0%	9.86%	0.02%	193,715	27
	Dataset 3	85.3%	1.39%	4.45%	8.83%	0.03%	188,776	27
	Dataset 4	91.69%	1.95%	0.58%	5.68%	0.1%	124,024	27
	Dataset 5	90.11%	0.51%	0.02%	9.31%	0.04%	148,164	27
	Dataset 6	84.59%	0.08%	0.83%	14.44%	0.04%	413,762	27
	Dataset 7	87.68%	0.33%	2.67%	9.09%	0.22%	104,853	27
	Dataset 8	90.53%	0.004%	0.007%	9.06%	0.4%	55,223	27
Vegetation	Dataset 1	1.02%	94.21%	1.57%	1.85%	1.35%	157,013	27
	Dataset 2	0.83%	97.16%	1.14%	0.73%	0.15%	148,961	27
	Dataset 3	0.97%	93.03%	3.33%	1.56%	1.11	65,191	27
	Dataset 4	1.53%	94.39%	1.66%	1.41%	1.01%	35,545	27
	Dataset 5	0.24%	97.02%	0.22%	1.22%	1.3%	34,462	27
	Dataset 6	0.98%	93.01%	0.86%	3.48%	1.67%	57,892	27
	Dataset 7	1.16%	93.65%	0.75%	2.19%	2.25%	15,905	27
	Dataset 8	1.09%	92.19%	1.8%	2.34%	2.57%	20,949	27
Roof	Dataset 1	0.49%	2.63%	96.13%	0.26%	0.48%	144,671	27
	Dataset 2	0.02%	1.16%	98.55%	0.15%	0.11%	26,430	27
	Dataset 3	1.77%	5.88%	91.42%	0.59%	0.34%	45,955	27
	Dataset 4	0.14%	6.52%	92.86%	0.2%	0.28%	36,137	27
	Dataset 5	0.18%	1.19%	98.14%	0.07%	0.41%	72,738	27
	Dataset 6	0.66%	0.38%	98.41%	0.31%	0.24%	248,716	27
	Dataset 7	0.7%	1.94%	96.05%	0.34%	0.97%	112,417	27
	Dataset 8	0.42%	0.17%	97.9%	0.17%	1.35%	164,376	27
Covered ground	Dataset 1	4.47%	0.99%	0.16%	94.33%	0	53,491	27
	Dataset 2	1.98%	1.27%	0.01%	96.71%	0.02%	64,778	27
	Dataset 3	8.65%	1.69%	0.81%	88.8%	0.04%	31,373	27
	Dataset 4	5.11%	0.57%	0.02%	94.09%	0.21%	13,910	27
	Dataset 5	7.78%	2.06%	0	90.16%	0.003%	31,186	27
	Dataset 6	1.4%	1.37%	0.24%	96.99%	0.008%	49,481	27
	Dataset 7	3.35%	0.63%	0.14%	95.88%	0	11,000	27
	Dataset 8	4.41%	0.008%	0	95.52%	0.05%	11,127	27
Facade	Dataset 1	1.71%	9.61%	4.96%	1.69%	82.02%	13,707	27
	Dataset 2	3.45%	1.89%	1.84%	1.84%	90.97%	6,189	27
	Dataset 3	0.08%	18.77%	0.5%	0.92%	79.73%	4,787	27
	Dataset 4	4.68%	10.52%	0.65%	1.56%	82.60%	2,313	27
	Dataset 5	1.09%	12.01%	10.49%	0.36%	76.06%	37,636	27
	Dataset 6	1.57%	4.14%	8.9%	0.53%	84.85	47,506	27
	Dataset 7	0.94%	6.25%	4.07%	1.01%	87.73%	24,852	27
	Dataset 8	0.76%	5.22%	6.92%	0.25%	86.84%	40,815	27

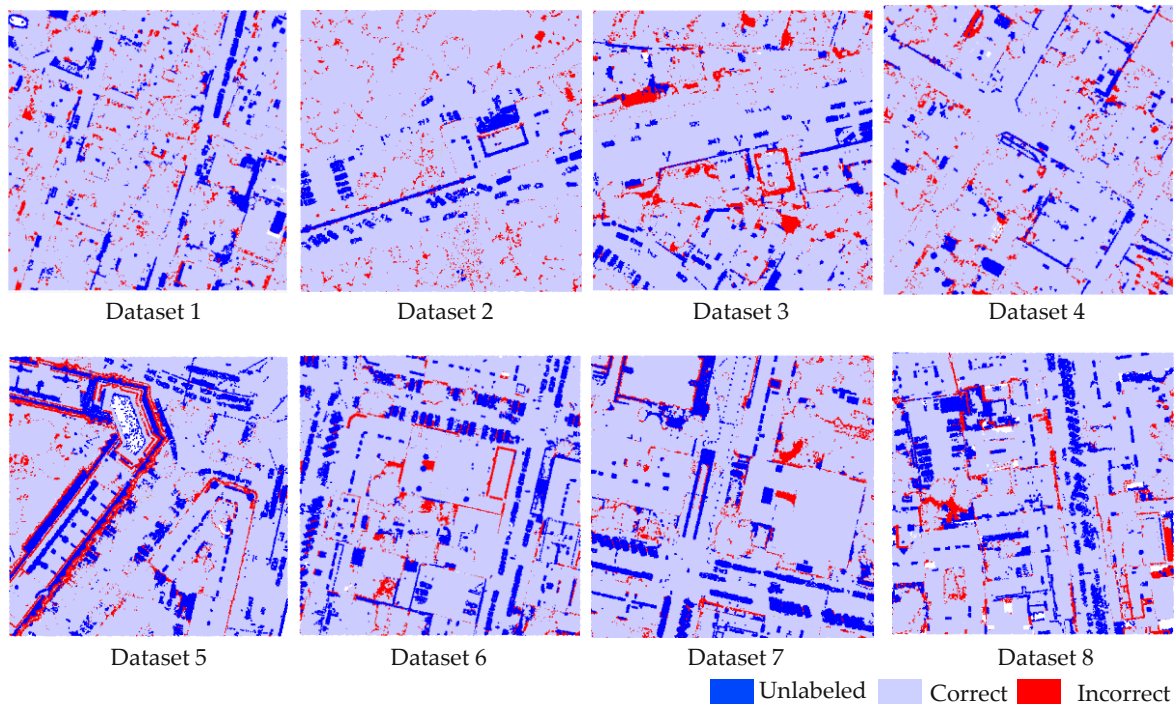


Figure 7. Qualification of the classification over eight datasets.

4.3.2. Classification Comparison

For the comparison, the classifiers KNN, DT, RF, and SVM are applied on the eight LiDAR datasets. The training data are randomly selected for 10 repetitions of the experiment, and the same training datasets are used in the TSRC dictionary learning, the training of the other classifiers, and the optimization in each experiments. To find the optimal parameters for KNN, DT, RF, and SVM, we built a misclassification rate function for each classifier based on the training dataset, then the minimum error rate is searched by parameters optimization, and the best parameters of each classifier were selected. The parameters to be optimized for each classifiers are listed in the following.

- KNN: the k nearest neighborhood points, distance computation function.
 DT: the minimum observations on each leaf, the minimum observations in each branch node, and the maximum number of branch node splits.
 RF: the number of predictors, and the parameters included for generating the decision tree, which contains the minimum observations on each leaf, the minimum observations in each branch node, and the maximum number of branch node splits.
 SVM: the kernel function, the kernel size and the box constraint which is the weight of cost of misclassification.

The mean OA and OA standard deviation of 10 experiments for all of the classifiers are summarized in Table 4. Based on Table 4, TSRC shows the best performance over all of the datasets in terms of mean OA, except for Dataset 1, where RF provides the best results. However, TSRC achieves the second best OA and it is just 0.36% lower than the best OA value for this Dataset. As for OA deviation, TSRC also has the lowest OA deviation for Datasets 2–8, while the OA deviation of TSRC for Dataset 1 is only 0.09% lower than that achieved by SVM. According to the OA deviation, the TSRC is less affected by the training data selection than the other classifiers investigated

Figure 8 shows the average accuracy per class for all eight datasets when using TSRC and other classifiers in 10 repetitions of the experiment. For the ground classification, the TSRC has the best accuracy in most cases; however, the accuracy of the TSRC is slightly lower than DT in Dataset 3 and lower than DT, RF, and SVM in Dataset 8. The accuracy gap of vegetation classification among TSRC, RF, and SVM is narrow for all of the datasets, but the vegetation classification accuracy is

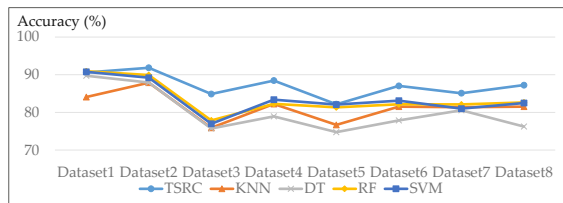
increased by 14.28% for Dataset 3 and 20.91% for Dataset 4 when compared to KNN. As displayed in Figure 8d, the accuracy of roof is significantly improved by TSRC for Dataset 4 and Dataset 5. As for covered ground classification, the accuracies of TSRC are significantly higher than that of DT, and higher than the accuracy obtained by KNN and RF for most cases. The difference between accuracies of TSRC and SVM are small. Additionally, TSRC delivers the best results among all of the classifiers for façade classification. The improvement is especially significant in façade detection for Dataset 3 and Dataset 4, the façades are badly distinguished by the other four classifiers. The accuracy of DT is only 51.03% and 43.62%, while the accuracies increase to 79.73% and 84.97% by TSRC.

KNN directly searches the similar feature vectors from the training data, and all of the attributes are used without any selection or weight assignment. Therefore, the classification results of KNN are not as good as TSRC. The optimal parameters for DT, RF, and SVM is dependent on the large amount of training data and multiple cross validation. Since only a few of training data are used to train DT, RF, and SVM in this paper, the classifiers are easily overfitting and biased. The classifiers work well for the training points, but it could not lead to a good the classification result for a large amount of test points. However, TSRC has better performance than those classifiers by using the same amount of training data.

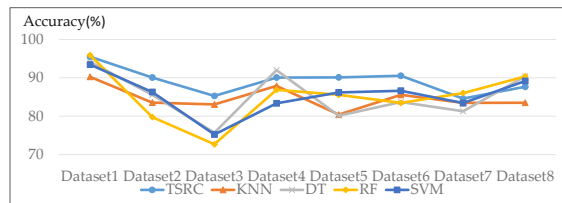
In a nutshell, the classification results for the eight LiDAR datasets demonstrate the effectiveness of TSRC in improving the classification performance, particularly enhancing the façade detection accuracy. Since façade points are influenced by their sparse density and mini-structures on the wall, and the feature values of façade points are not reliable and yield a bad detection result by the feature-based classifiers. Due to the combination of points spatial distribution and feature values, TSRC could effectively improve the façade detection accuracy.

Table 4. Mean overall accuracy (OA) and OA deviation for all dataset using different classifiers. Bold values indicate the highest overall accuracy and the lowest standard deviation with the respective classifier.

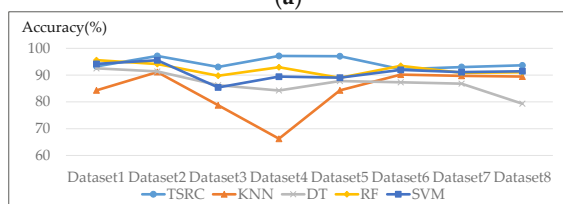
Dataset	Mean OA					OA std.dev				
	TSRC	KNN	DT	RF	SVM	TSRC	KNN	DT	RF	SVM
Dataset 1	90.57%	84.06%	89.72%	90.93%	90.15%	0.77%	1.07%	0.89%	0.77%	0.68%
Dataset 2	91.85%	87.88%	87.91%	89.91%	89.20%	0.84%	1.34%	1.94%	1.19%	0.91%
Dataset 3	84.98%	75.90%	75.72%	77.87%	76.98%	0.63%	1.58%	2.46%	1.81%	1.63%
Dataset 4	88.44%	82.21%	78.93%	82.25%	83.36%	0.89%	1.53%	2.53%	1.73%	2.17%
Dataset 5	82.16%	76.68%	74.75%	81.34%	82.07%	0.69%	0.84%	1.83%	2.81%	0.67%
Dataset 6	87.03%	81.53%	77.85%	82.14%	83.11%	0.71%	1.66%	2.84%	1.72%	1.77%
Dataset 7	85.09%	81.37%	80.55%	82.10%	80.99%	0.91%	1.70%	2.67%	1.32%	1.61%
Dataset 8	87.24%	81.52%	76.23%	82.63%	82.48%	0.64%	1.19%	2.48%	1.53%	1.24%



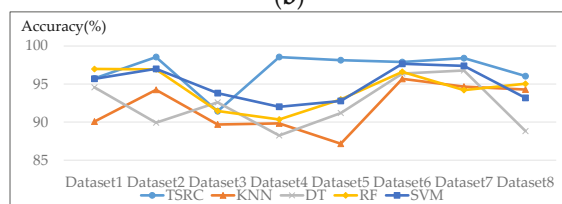
(a)



(b)



(c)



(d)

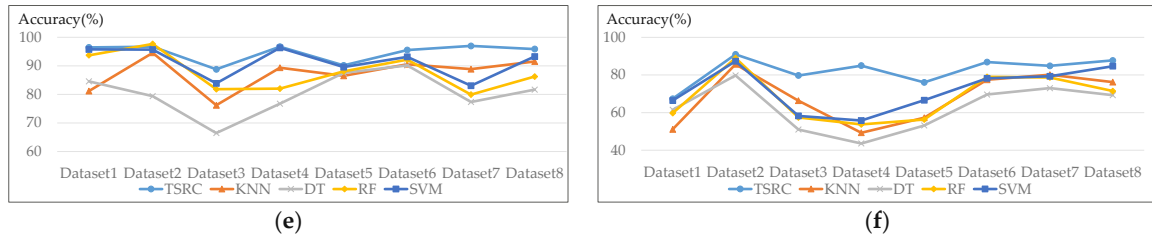


Figure 8. Accuracy per class comparisons of tensor-based sparse representation classification (TSRC) and other classifiers: (a) Overall accuracy; (b) Open ground accuracy; (c) Vegetation accuracy; (d) Roof accuracy; (e) Covered ground accuracy; (f) Facade accuracy.

5. Discussion

One of our framework's crucial parts is the tensor reconstruction. The tensor is reconstructed by each sub-dictionary and its corresponding subset of the sparse tensor. Then, the class label is determined by the minimum reconstruction error. Theoretically, it is possible that there are equal reconstruction errors. However, this tie situation never happened in our experiments. Since the bases in the sub-dictionary are different between each other, it is very unlikely that a sparse tensor contains certain subsets of tensors that could recover the same tensors. Therefore, the tie situation of reconstruction errors barely happens.

Since there are parameters need to set manually in this approach, such as the neighborhood size in tensor generation, sparsity level in TOMP, and the number of training data, we conducted a series of experiments on how those parameters influence the classification result. This is discussed below.

5.1. Impact of Neighborhood Size Selection in Tensor Generation

The impact of KNN neighborhood size in tensor processing on the classification results is assessed. The neighborhood size indicates how many points are involved in the tensor generation, and it depends on the scale parameter k_t (k nearest neighbor points). Therefore, we utilize Dataset 4 and vary k_t values over the interval between $k_t = 20$ and $k_t = 120$ with $\Delta k_t = 20$. Dataset 4 contains various types of objects, such as slopes, small detached houses, high-rising buildings, low vegetation, and high trees, so it is used to test the impact of neighborhood size selection. The classification is evaluated by OA and Kappa index in Table 5.

The OA and Kappa index slightly change by using various k_t values, the tendency is similar across the open ground, vegetation, roof and covered ground class. Only the façade objects are influenced by the k_t values; the accuracy of façade is lower when smaller k_t values are used, and façade accuracy increases when the k_t value is larger than 80. In order to achieve the high accuracies of all types of objects, k_t value is suggested setting in the range of 80–120. We use a k_t value of 80 in our classification.

Table 5. Accuracy per class, OA and Kappa index of TSRC for Dataset 4 with different k_t values.

k_t	20	40	60	80	100	120
Open Ground	91.13%	88.62%	92.81%	88.52%	93.07%	90.65%
Vegetation	85.11%	93.07%	90.13%	91.79%	91.31%	86.91%
Roof	92.98%	89.35%	91.77%	93.65%	94.08%	92.16%
Covered ground	93.00%	96.82%	93.37%	97.07%	90.84%	96.45%
Facade	68.86%	73.93%	78.57%	83.63%	85.38%	93.3%
OA	86.83%	86.69%	89.27%	87.75%	90.39%	88.39%
Kappa Index	0.7946	0.7925	0.8327	0.809	0.8502	0.8189

5.2. Impact of Sparsity Level

The sparsity level indicates that the number of bases needed to be extracted from the dictionary for data reconstruction in the sparse coding processing. As in Section 5.1 Dataset 4, which exhibits large variety within each class, is used to test the impact of various sparsity levels on the

classification result. The sparsity level s is set from $s = 5$ to $s = 18$, as shown in Table 6. The OA and Kappa index remain unchanged by using different sparsity level in the TOMP phase, which also demonstrates that the classification result is not sensitive to the sparsity level. As the initial value of $s = 9$ is only 0.1% less than the optimal value, this initial value is kept for further experiments.

Table 6. Accuracy per class, OA and Kappa index of TSRC for Dataset 4 with different sparsity level.

s	5	7	9	11	13	15	17	18
Open Ground	91.36%	91.76%	92.13%	92.29%	92.43%	91.36%	92.42%	92.52%
Vegetation	92.44%	92.42%	92.21%	92.06%	91.55%	92.44%	86.67%	85.77%
Roof	95.44%	95.48%	95.47%	95.70%	95.44%	95.44%	95.76%	95.56%
Covered ground	95.17%	94.25%	93.73%	93.35%	93.24%	95.17%	92.5%	92.09%
Facade	86.49%	86.75%	88.7%	88.7%	85.71%	86.49%	81.82%	81.82%
OA	89.50%	89.69%	89.85%	89.92%	89.94%	89.49%	89.14%	89.01%
Kappa Index	0.8363	0.8392	0.8418	0.8428	0.8431	0.8361	0.8306	0.8286

5.3. Impact of Training Data

Since learning a classifier strongly depends on the given training data, we further consider the influence of varying the amount of training data on the classification results. We focus on the impact of 10 different amount of training examples, the parameter of training data amount Nt varies from $Nt = 9$ to $Nt = 90$, with a step size of $\Delta Nt = 9$. The general behavior of the TSRC and other classifiers under various numbers of training samples is analyzed. KNN, DT, RF, and SVM are chosen for classification comparison. Again, the LiDAR Dataset 4 is used for classification, which contains 352318 points.

The overall accuracy values are given in Figure 9. Generally, the TSRC performs better than the other four classifiers independent of the number of training samples. The overall accuracy tends to increase for all of the classification methods. The overall accuracy remains steady from $Nt = 27$ to $Nt = 90$ by TSRC. Therefore, the training data amount is set as 27 in the classification experiments.

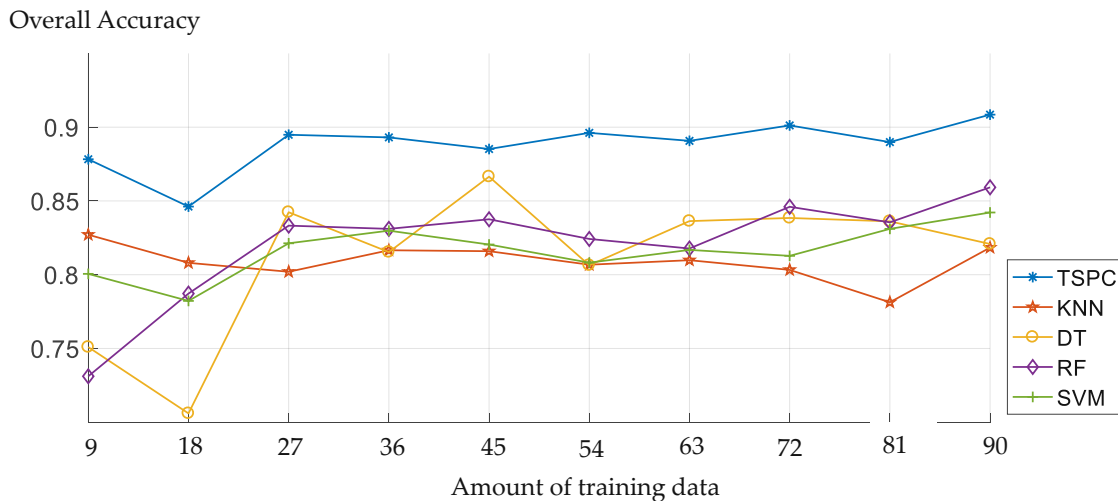


Figure 9. The OA of dataset 3 with different amount of training tensors by different classifiers.

6. Conclusions

In this paper, a tensor-based sparse representation classification frame work is proposed for 3D LiDAR point cloud classification. In this framework, each point is considered as the fourth-order tensor in order to make full use of geometry and feature information. 18 features per point are extracted from the 3D LiDAR points, and all features are utilized for classification without any feature selection procedure. Based on the Tucker Decomposition, the structured and discriminative

dictionaries along each mode are learned for tensor data classification. Then, the test tensor data is projected onto the dictionary set to get its sparse tensor. After that, using different class-specific dictionary sets and its corresponding subsets of the sparse tensor to recover the test tensor data, meanwhile the residuals per class are determined. Finally, the label of the test tensor is determined by the minimal residual.

A series of experiments of TSRC suggest that the TSRC is barely dependent on the neighborhood size of tensor generation and the sparsity level. The TSCR can be successfully conducted by using only a few training samples. Based on the eight real airborne LiDAR points classification result, the OAs of TSRC are beyond 80%, with only 27 training tensors being used per class. Additionally, TSRC achieves a good classification when compared with other classifiers. TSRC has respectable performance in identifying objects with less distinguishable features, such as façade.

Acknowledgments: The LiDAR data comes from Vienna city government. The project is supported by National Natural Science Foundation of China (Project No. 41371333 and Project No. 41771481). Norbert Pfeifer was partially supported by the Ludwig Boltzmann Institute for Archaeological Prospection and Virtual Archaeology (archpro.lbg.ac.at), funded by the Ludwig Boltzmann Gesellschaft.

Author Contributions: Nan Li designed the study, conducted the experiments and wrote the manuscript. Norbert Pfeifer supported the study design and contributed to the manuscript. Chun Liu provided the initial idea and basic concepts.

Conflicts of Interest: The authors declare that there is no conflict of interests regarding the publication of this article.

References

1. Secord, J.; Zakhor, A. Tree detection in urban regions using aerial lidar and image data. *IEEE Geosci. Remote Sens. Lett.* **2007**, *4*, 196–200.
2. Lodha, S.K.; Fitzpatrick, D.M.; Helmbold, D.P. Aerial lidar data classification using adaboost. In Proceedings of the Sixth International Conference on 3-D Digital Imaging and Modeling, Montreal, QC, Canada, 21–23 August 2007; pp. 435–442.
3. Garcia-Gutierrez, J.; Gonçalves-Secob, L.; Riquelme-Santosa, J. Decision trees on lidar to classify land uses and covers. In Proceedings of the ISPRS Workshop: Laserscanning, Paris, France, 1–2 September 2009; pp. 323–328.
4. Guo, L.; Chehata, N.; Mallet, C.; Boukir, S. Relevance of airborne lidar and multispectral image data for urban scene classification using random forests. *ISPRS J. Photogramm. Remote Sens.* **2011**, *66*, 56–66.
5. Najafi, M.; Namin, S.T.; Salzmann, M.; Petersson, L. Non-associative higher-order markov networks for point cloud classification. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; Springer: Cham, Switzerland, 2014; pp. 500–515.
6. Anguelov, D.; Taskarf, B.; Chatalbashev, V.; Koller, D.; Gupta, D.; Heitz, G.; Ng, A. Discriminative learning of markov random fields for segmentation of 3D scan data. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, 20–25 June 2005; pp. 169–176.
7. Niemeyer, J.; Rottensteiner, F.; Soergel, U. Contextual classification of lidar data and building object detection in urban areas. *ISPRS J. Photogramm. Remote Sens.* **2014**, *87*, 152–165.
8. Im, J.; Jensen, J.R.; Hodgson, M.E. Object-based land cover classification using high-posting-density lidar data. *GISci. Remote Sens.* **2008**, *45*, 209–228.
9. Renard, N.; Bourennane, S. Dimensionality reduction based on tensor modeling for classification methods. *IEEE Trans. Geosci. Remote Sens.* **2009**, *47*, 1123–1131.
10. Vasilescu, M.A.O.; Terzopoulos, D. Multilinear image analysis for facial recognition. In Proceedings of the 16th International Conference on Pattern Recognition, Quebec City, QC, Canada, 11–15 August 2002; pp. 511–514.
11. Kuang, L.; Hao, F.; Yang, L.T.; Lin, M.; Luo, C.; Min, G. A tensor-based approach for big data representation and dimensionality reduction. *IEEE Trans. Emerg. Top. Comput.* **2014**, *2*, 280–291.
12. Huang, K.; Aviyente, S. Sparse representation for signal classification. In Proceedings of the Advances in Neural Information Processing Systems 19, Vancouver, BC, Canada, 4–7 December 2006; pp. 609–616.

13. Wright, J.; Yang, A.Y.; Ganesh, A.; Sastry, S.S.; Ma, Y. Robust face recognition via sparse representation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 210–227.
14. Chen, Y.; Nasrabadi, N.M.; Tran, T.D. Hyperspectral image classification using dictionary-based sparse representation. *IEEE Trans. Geosci. Remote Sens.* **2011**, *49*, 3973–3985.
15. Sun, Y.; Liu, Q.; Tang, J.; Tao, D. Learning discriminative dictionary for group sparse representation. *IEEE Trans. Image Process.* **2014**, *23*, 3816–3828.
16. Rubinstein, R.; Bruckstein, A.M.; Elad, M. Dictionaries for sparse representation modeling. *Proc. IEEE* **2010**, *98*, 1045–1057.
17. Ophir, B.; Lustig, M.; Elad, M. Multi-scale dictionary learning using wavelets. *IEEE J. Sel. Top. Signal Process.* **2011**, *5*, 1014–1024.
18. Smith, L.N.; Elad, M. Improving dictionary learning: Multiple dictionary updates and coefficient reuse. *IEEE Signal Process. Lett.* **2013**, *20*, 79–82.
19. Sadeghi, M.; Babaie-Zadeh, M.; Jutten, C. Dictionary learning for sparse representation: A novel approach. *IEEE Signal Process. Lett.* **2013**, *20*, 1195–1198.
20. Yang, M.; Zhang, L.; Feng, X.; Zhang, D. Fisher discrimination dictionary learning for sparse representation. In Proceedings of the International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 543–550.
21. Roemer, F.; Del Galdo, G.; Haardt, M. Tensor-based algorithms for learning multidimensional separable dictionaries. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 4–9 May 2014; pp. 3963–3967.
22. Caiafa, C.F.; Cichocki, A. Block sparse representations of tensors using kronecker bases. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 25–30 March 2012; pp. 2709–2712.
23. Lee, Y.K.; Low, C.Y.; Teoh, A.B.J. Tensor kernel supervised dictionary learning for face recognition. In Proceedings of the Signal and Information Processing Association Annual Summit and Conference (APSIPA), Hong Kong, China, 16–19 December 2015; pp. 623–629.
24. Peng, Y.; Meng, D.; Xu, Z.; Gao, C.; Yang, Y.; Zhang, B. Decomposable nonlocal tensor dictionary learning for multispectral image denoising. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 2949–2956.
25. Kolda, T.G.; Bader, B.W. Tensor decompositions and applications. *SIAM Rev. Soc. Ind. Appl. Math.* **2009**, *51*, 455–500.
26. Yang, S.; Wang, M.; Li, P.; Jin, L.; Wu, B.; Jiao, L. Compressive hyperspectral imaging via sparse tensor and nonlinear compressed sensing. *IEEE Trans. Geosci. Remote Sens.* **2015**, *53*, 5943–5957.
27. Zhang, H.; Nasrabadi, N.M.; Zhang, Y.; Huang, T.S. Multi-view automatic target recognition using joint sparse representation. *IEEE Trans. Aerosp. Electron. Syst.* **2012**, *48*, 2481–2497.
28. Phillips, P.J. Matching pursuit filters applied to face identification. *IEEE Trans. Image Process.* **1998**, *7*, 1150–1164.
29. Yang, M.; Zhang, L.; Yang, J.; Zhang, D. Metaface learning for sparse representation based face recognition. In Proceedings of the 17th IEEE International Conference on Image Processing (ICIP), Hong Kong, China, 26–29 September 2010; pp. 1601–1604.
30. Höfle, B.; Mücke, W.; Dutter, M.; Rutzinger, M.; Dorninger, P. Detection of building regions using airborne lidar—A new combination of raster and point cloud based gis methods. In Proceedings of the GI-Forum 2009—International Conference on Applied Geoinformatics, Salzburg, Austria, 2009; pp. 66–75.



© 2017 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).

Publication III

- Title: Improving LiDAR classification accuracy by contextual label smoothing in post-processing
- Authors: Li, N., Liu, C. and Pfeifer, N
- Published in: ISPRS Journal of Photogrammetry and Remote Sensing, 148, 13-31
DOI: 10.1016/j.isprsjprs.2018.11.022

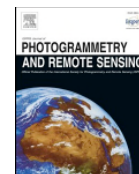


Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.



Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing



Improving LiDAR classification accuracy by contextual label smoothing in post-processing

Nan Li^{a,b,*}, Chun Liu^a, Norbert Pfeifer^b

^a College of Survey and Geoinformation, Tongji University, 200092 Shanghai, China

^b Department of Geodesy and Geoinformation, Technische Universität Wien, 1040 Vienna, Austria



ARTICLE INFO

Keywords:

Optimal neighborhood
Probabilistic label relaxation
Point cloud
Neighborhood dependency

ABSTRACT

We propose a contextual label-smoothing method to improve the LiDAR classification accuracy in a post-processing step. Under the framework of global graph-structured regularization, we enhance the effectiveness of label smoothing from two aspects. First, each point can collect sufficient label-relevant neighborhood information to verify its label based on an optimal graph. Second, the input label probability set is improved by probabilistic label relaxation to be more consistent with the spatial context. With this optimal graph and reliable label probability set, the final labels are computed by graph-structured regularization. We demonstrate the contextual label-smoothing approach on two separate urban airborne LiDAR datasets with complex urban scenes. Significant improvements in the classification accuracies are achieved without losing small objects (such as façades and cars). The overall accuracy is increased by 7.01% on the Vienna dataset and 6.88% on the Vaihingen dataset. Moreover, most large, wrongly labeled regions are corrected by long-range interactions that are derived from the optimal graph, and misclassified regions that lack neighborhood communications in terms of correct labels are also corrected with the probabilistic label relaxation.

Airborne LiDAR (Light Detection And Ranging) is a useful and efficient technique for acquiring 3D information in urban areas regardless of relief displacements, tree-canopy penetration and lighting conditions (Yan et al., 2015). For many applications, point-cloud classification is a basic step in LiDAR processing. Because of the complex combination of artificial and natural objects in cities, the automated classification of 3D point clouds is a very challenging task in urban areas (Niemeyer et al., 2014; Vosselman et al., 2017; Yan et al., 2015).

A comprehensive review of LiDAR classification techniques can be found in Yan et al. (2015), which summarizes pixel-based and point-based classification methods. To eliminate characteristic point loss and losses in height accuracy, most of the proposed classification algorithms focus on individual point classification (Yastikli and Cetin, 2016). Various supervised statistical approaches exist for point-based classification, such as decision trees (Tran et al., 2015), Bayesian discriminant classifiers (Khoshelham et al., 2013), support vector machines (SVM) (Mallet et al., 2011; Secord and Zakhor, 2007), AdaBoost (Lodha et al., 2007; Xu et al., 2014) and random forests (Guo et al., 2011; Niemeyer et al., 2011). In these methods, each point is represented by a set of

handcrafted features, and then the feature vectors and corresponding labels of training data serve as the input to build a decision rule or probabilistic function as the classifier. The benefits of the common supervised statistical methods are their ease of application and availability in free software. On the other hand, feature vectors might be unstable at the boundaries of objects or in areas of lower point density. This phenomenon may result in inhomogeneous classification in complex scenes because each feature vector is treated individually without considering the correlated labels in the neighborhood. Thus, adding contextual information could support achieving smooth classification results.

One way to incorporate contextual information is to add a smoothness constraint in the classifiers, which is called contextual classification. The contextual classification of 3D points is usually based on probabilistic graphical models, such as Markov random fields (MRF) (Angelov et al., 2005) and conditional random fields (CRF) (Niemeyer et al., 2012). Both of these models require the specification of a posterior energy function, which is a linear combination of a unary potential and a pairwise potential. Unary potentials express the label probabilities given the observed features of the node, and pairwise potentials incorporate spatial contextual information in a local

* Corresponding author at: Department of Geodesy and Geoinformation, Technische Universität Wien, Vienna, Austria; College of Survey and Geoinformation, Tongji University, Shanghai, China.

E-mail address: nan.li@geo.tuwien.ac.at (N. Li).

<https://doi.org/10.1016/j.isprsjprs.2018.11.022>

Received 9 August 2018; Received in revised form 26 November 2018; Accepted 27 November 2018

Available online 13 December 2018

0924-2716/© 2018 International Society for Photogrammetry and Remote Sensing, Inc. (ISPRS). Published by Elsevier B.V. All rights reserved.

neighborhood. By generating an adjacency graph for the 3D points, MRFs and CRFs can be successfully applied to contextual classification to achieve smoother results than classifications that are based on individual independent features (Niemeyer et al., 2014; Shapovalov et al., 2010). Furthermore, contextual classification that is based on segments has also been conducted to obtain long-range interactions (Niemeyer et al., 2016; Vosselman et al., 2017). These contextual classification methods are usually computationally extensive in terms of training time. However, one suggestion in terms of these methods is to add interaction features to distinguish semantic classes.

Another approach to achieve smooth classification results is to develop a label-smoothing technique that considers the neighborhood context in post-processing. Compared to MRFs and CRFs, label-smoothing techniques do not require the extensive computation of contextual classifiers or the extraction of useful interaction features. Label-smoothing methods usually incorporate neighborhood dependencies in post-processing, which is time efficient and flexible to conduct on points regardless of the acquisition techniques. Equivalent smoothing classification results can also be achieved by exploring the interactions of neighborhood labels during label-smoothing processing (Landrieu et al., 2017; Schindler, 2012). Therefore, we concentrate on label-smoothing approaches to improve the classification accuracy in urban areas.

Based on our observations of feature-independent classification results (here, a random-forest classifier is used), three types of label errors exist: small wrongly labeled regions (Fig. 1(a)), large wrongly labeled regions (Fig. 1(b)) and wrongly labeled regions that lack sufficient correct contextual information (Fig. 1(c)); in the last case, the neighborhood consistency from a poor initial classification is not reliable enough to deduce the correct label interactions. Small wrongly labeled regions can be solved by common local smoothing methods, such as majority voting or Gaussian filters. However, to simultaneously correct large wrongly labeled regions, an adaptive neighborhood is required for the acquisition of long-range interactions. For wrongly labeled regions that lack sufficient correct contextual information, a modification of the initial label probability set of points is conducted to be more logical with the spatial context in a real scene.

Our goal is to improve the initial classification results by applying a label-smoothing method in post-processing. In this paper, the suggested label-smoothing approach is based on the graph-structured regularization framework that was proposed in Landrieu et al. (2017). First, we design an optimal graph based on the spatial coordinates of each point. Then, to improve the reliability of the initial label probabilities, we introduce probabilistic label relaxation (PLR) to iteratively update the label probability over each class of each point. Finally, the optimal graph and improved label probabilities are used as the input to the graph-structured regularization to achieve the final labels of the points. Our main contributions are as follows:

- We propose an optimal-graph estimation method for label smoothing. Instead of finding the optimal searching range or the number of nearest points, this optimal neighborhood involves selecting neighboring points that have a high possibility to belong to the same object with the concerned points and exclude neighboring points that have significant geometric changes. Thus, large wrongly labeled regions can be corrected from the sufficient reliable neighborhood label information, while small objects are also preserved and over-smoothing is avoided.
- We use PLR to improve the reliability of the initial label probability set. Based on the neighborhood compatibility, the label probabilities of neighboring points reinforce the weights to each class probability of the concerned point. After relaxation, the resulting probability values are more consistent with the prior knowledge of neighborhood dependencies. By using PLR, the label probability set can provide more reliable information for the graph-structured regularization.

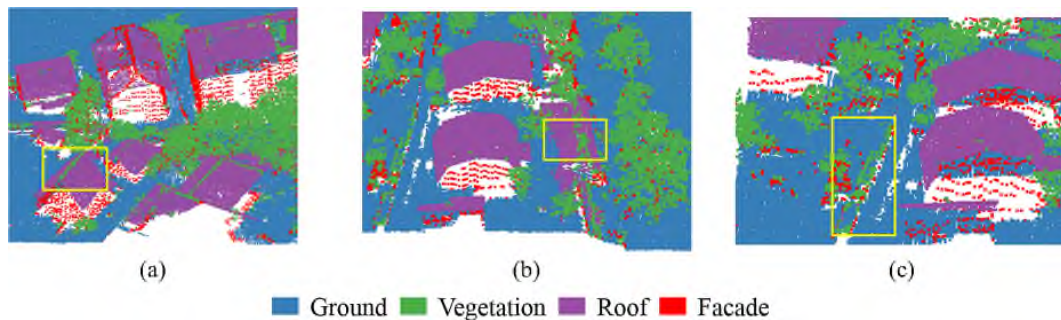
This paper is organized as follows. We begin with an introduction of related work in Section 2, focusing on a review of the related literature and the idea of graph-structured regularization for label smoothing. In Section 3, we explain the optimal graph estimation and label-probability improvements based on PLR in detail. Section 4 presents the performance of the proposed label-smoothing approach, which is demonstrated and evaluated on two airborne LiDAR point sets over complex urban scenes. Afterwards, Section 5 discusses the advantages of our label-smoothing method and the effects of the initial parameter selections. The paper concludes with Section 6.

2.1. Literature review

2.1.1. Contextual classification for point clouds

MRFs and CRFs are two commonly used models for the contextual classification of 3D points. MRFs are generative models and encourage that the local neighborhood shares the same label. For instance, an associative Markov network (AMN) was applied for the contextual segmentation of terrestrial laser-scanning points (Angelov et al., 2005). Munoz et al. (2008) extended the AMN model to consider local directional information and produce an anisotropic model. The major problem of such approaches is that AMNs only assume that neighboring points likely belong to the same object and that objects of different classes are independent. The occurrence probabilities of different objects are neglected.

To overcome the drawbacks of the MRF model, a more general discriminative mode is provided by CRFs. Label dependencies between adjacency nodes are inferred in CRFs by incorporating neighborhood spatial interactions among labels and the observed features of edges in



Different types of label errors: (a) small wrongly labeled regions; (b) large wrongly labeled regions; (c) wrongly labeled regions that lack sufficient correct contextual information.

the graph (Kumar and Hebert, 2006). Moreover, CRFs can directly estimate conditional probabilities from the given features, so CRFs has been more widely used for point-cloud contextual classification. Laible et al. (2013) compared CRFs and MRFs during fused 3D LiDAR and camera-data classification, and the results confirmed that the CRFs had better performance than the MRFs for this classification task. Rusu et al. (2009) robustly segmented a point cloud into different geometric surface classes based on CRFs. A hybrid conditional random field was proposed to classify ground and non-ground points for automatic DTM generation (Lu et al., 2009). In this model, hidden random variables contain discrete variables that refer to class labels and continuous random variables that represent the height of the underlying ground surface at that point. For more complex urban-scene classification, Niemeyer et al. (2014) conducted a point-wise contextual classification based on CRFs. These authors compared three versions of CRF-based classifiers: the first version trained the unary and pairwise potentials independently by using a linear model; the second version was still based on the linear model, but the weights for both potentials were determined simultaneously; the third version used two independent random forest classifiers for the unary and pairwise potentials. CRFs that are based on random forests can achieve accurate classification results with the fastest training speed. However, the interactions only happen at a very local level in these methods. Thus, the results may maintain isolated clusters of points that are classified to the wrong labels. Therefore, long-range interactions need to be considered.

Generating an adjacency graph that links long-range neighbors is a commonly used strategy to consider long-range interactions. A multi-range CRF that was proposed by Luo and Sohn (2014) constructs two independent graphical models: one is a short-range CRF for enhancing local smoothing within a short range, and the other is a long-range CRF for favoring a global and asymmetric regularity of spatial arrangement between classes within a long range. Jung et al. (2016) estimated a multi-range CRF by considering directional information, and three types of pairwise contextual terms were included in this CRF model: short-range pairwise potentials, horizontal long-range pairwise potentials and vertical long-range pairwise potentials. A second popular strategy in terms of long-range interactions is based on segmentation. A multi-scale CRF is performed on the “over-segments” of 3D terrestrial LiDAR data (Lim and Suter, 2009); this multi-scale CRF includes node potentials, local edge potentials and regional edge potentials, which represent the long-range connectivity between segments. Niemeyer et al. (2016) proposed a two-layer hierarchical CRF model that consists of a point-based layer and a segment-based layer. Segments are generated from the labelling results of the first point-based layer. Then, the classification result of the segment-based layer is introduced as an energy term for the next iteration of the point-based layer. The classification accuracy is improved compared to pure point-based CRF classification and pure segment-based CRF classification. Shapovalov et al. (2010) used over-segments to achieve long-range interactions. Vosselman et al. (2017) conducted a segment-based CRF, and the interaction features of segments were derived from a local analysis of the points near segment boundaries. In this study, large segments of ground or standing water bodies were defined as fixed nodes because of their valuable context information, and the segments were considered to provide more representative features than single points for classification tasks.

2.1.2. Label-smoothing techniques

Label-smoothing methods after classification are usually conducted in a local neighborhood or are based on a global regularization. A summary and comparison of local label-smoothing techniques can be found in Schindler (2012). Most research on local label smoothing focused on the assignments of weights to neighboring entities. In the local filter that was designed in Lu et al. (2016), weights of neighbors were calculated based on the spatial distance and a geometric similarity. An edge-preserving filter was used in Kang et al. (2014) to recalculate the

probability of each class; the weights in this edge-preserving filter consisted of color values from the guidance images, which are generated by the first principal component or the first three principal components of the hyperspectral image. These weigh based approaches can achieve a good smoothing result when only small wrongly labeled regions exist in the initial classification and greatly depend on the definition of the neighborhood. In addition to weighted local filters, other local smoothing methods aim to reduce variations in the label probabilities in a neighborhood for local label smoothing. Li et al. (2014) employed a smoothing filter that was based on sparse gradient minimization to eliminate the inherent variations within a small neighborhood. However, this method only works in small neighborhoods. Local graph-cut segmentation was used to optimize the initial classification results (Guo et al., 2015). For every reliable object, points that belonged to this object were defined as the foreground seed points, while points of other reliable objects nearby were defined as background seed points. After graph-cut based segmentation, the labels of unreliable points that were segmented into the foreground were then optimized. However, the threshold that was used to identify the reliable objects was hard to determine, and large wrongly labeled regions still could not be fixed. A relearning strategy for label smoothing was adopted by Huang et al. (2014). The class-frequency histogram and a primitive co-occurrence matrix were learned and updated iteratively to estimate the class spatial arrangement and correlation in a certain direction. Consequently, erroneous labels were corrected and classification was smoothed by this relearning procedure. The pixels in the images were arranged in a highly regular format, so spatial direction information could be easily found from the grid, but defining such a directional constraint in an irregular point cloud was difficult.

Apart from local smoothing, some researchers explored global optimization by considering the labels of all points in a scene simultaneously. A global graph-structured regularization framework for the label smoothing of point-cloud classification was proposed in Landrieu et al. (2017). This method simultaneously considered the spatial relationship and label probabilities of 3D points by using a cost function, and smooth classification was achieved by finding the optimal solution for this cost function. The graph is simply built based on a KNN (k nearest neighbors) in this paper, so large wrongly labeled regions and regions that lack consistent label correlations still remain after smoothing. Li et al. (2016) globally optimized the initial classification by adopting a multi-label graph that was cut at the first stage, and then the optimized classification was further refined by a local optimization approach that was based on an object-oriented decision tree. This object-oriented decision tree was estimated by prior knowledge: for instance, the average height of a building above the ground should be greater than a certain value. However, the use of this threshold to distinguish confused categories is often hard to define in complex urban areas.

2.1.3. Graph structure design

Generally, the smoothness terms of contextual models are structured by the adjacency graph, which is constructed by connecting neighborhood points. The neighborhood definitions of 3D points are usually characterized by spherical neighborhoods (Lee and Schenk, 2002), cylindrical neighborhoods (Filin and Pfeifer, 2005) and a fixed number of nearest points (Linsen and Prutzsch, 2001). However, defining a constant-scale neighborhood requires prior knowledge of the 3D points in the scene. Moreover, the neighborhood size might vary from the density and geometric structures, so many researchers have favored data-driven adaptive neighborhood-selection approaches. Lalonde et al. (2005) determined the neighborhood scale by iteratively computing the curvature, density and noise. Demantke et al. (2011) computed three-dimensionality features (linear, planar and volumetric) on a spherical neighborhood at various sizes; one-dimensionality features should dominate over the others for a neighborhood of optimal size. Weinmann et al. (2015) selected the optimal number of nearest

points by the minimal entropy of eigenvalues. He et al. (2017) divided points into scatter regions and regular regions based on the local surface variation. Based on minimal entropy neighborhood selection, the neighborhood was recovered by the adaptive k nearest neighbors and an adaptive sphere neighborhood in scatter regions and regular regions, respectively.

Neighborhood for segments can be defined in different ways. Vosselman et al. (2017) determined the neighborhood of segments by searching along the segment boundary points' neighborhood. Najafi et al. (2014) constructed a clique structure from segments that were located in the same vertical structures in the point cloud and suggested that the vertical placements of objects can convey more geometric information. Wang et al. (2018) defined a search radius within the 99th quantile of the point spacing to find adjacent segments, and the point spacing for a single point was defined as the distance to its nearest point.

2.2. Graph-structured regularization preliminaries

Graph-structured regularization (GSR), which was proposed by Landrieu et al. (2017), has achieved good results for point-cloud label smoothing compared to local label-smoothing methods. Given the initial labeling probability set, GSR finds a new probability set with increased spatial smoothness while remaining as close as possible to the input initial labeling probability set. Thus, GSR considers label smoothing as the solution of an optimization problem, the objective function of which is structured by an adjacency graph that represents the spatial relationships among 3D points.

First, we consider $G = (V, E)$, where V are the nodes (vertices) and E are the edges. Let $C = [c_1, \dots, c_N]$ be the set of classes, with N being the number of classes. With every node $i \in V$, a random variable l_i is associated, specifying the label of node i . l_i is determined by the probability distribution over the class set on node i , which is denoted by P_i . Usually, the predicted l_i is the label that has the highest probability value. Then, let Pr denote the initial labeling probability set over all nodes, with Pr being the objective probability set. Thus, the GSR problem can be described with the following energy function, which finds Pr by minimizing the energy:

$$Pr = \arg \min_{Pr} \{\Phi(Pr, Pr) + \lambda \Psi(Pr)\} \quad (1)$$

where $\Phi(Pr, Pr)$ is the fidelity term, which enforces the influence of the initial probability set Pr . $\Psi(Pr)$ is the graph-structured regularizer that encourages solutions that are spatially smooth. $\lambda > 0$ is the regularization strength. The fidelity term can be written as

$$\Phi(Pr, Pr) = \sum_{i \in V} \varphi(P_i, Pr_i) \quad (2)$$

The fidelity function $\varphi(P_i, Pr_i)$ should be minimal when $P_i = Pr_i$ and increases by the difference between P_i and Pr_i . Thus, the fidelity term is defined as the following linear fidelity in this paper, although other functions can be chosen, such as linear-logarithmic fidelity or quadratic fidelity, as mentioned in Landrieu et al. (2017).

$$\varphi_{linear}(P_i, Pr_i) = -(P_i, Pr_i) = - \sum_{c_k \in C} P_i(l_i = c_k) \cdot Pr_i(l_i = c_k) \quad (3)$$

For a deviation from a very confident classification (i.e., one class probability close to 1 and the other class probabilities close to zero), the fidelity energy increase for a change in the probabilities is larger, whereas for an uncertain classification (i.e., all the probabilities have similar values), the fidelity energy increase for a probability change is moderate.

The regularizer $\Psi(Pr)$ is structured by the adjacency graph $G = (V, E)$. The goal of the regularizer function is to favor adjacent nodes in the graph G sharing the same label. Therefore, $\Psi(Pr)$ can be factorized over all edges of G in the following form:

$$\Psi(Pr) = \sum_{(i,j) \in E} \varphi(l_i, l_j) \quad (4)$$

In this paper, we use the Potts model to define the regularizer function:

$$\varphi(l_i, l_j) = \begin{cases} 0 & l_i = l_j \\ 1 & l_i \neq l_j \end{cases} \quad (5)$$

Finally, the algorithms for minimizing the objective function Eq. (1) depend on the respective properties of the fidelity and regularizer functions. For the Potts model, α -expansion, which is a variation of the graph-cut optimization algorithm (Boykov and Kolmogorov, 2004), is used to solve the objective function Eq. (1). The α -expansion algorithm solves the multi-label energy-minimization problem by iterative expansion (Boykov et al., 2001). During one expansion operation, only movement to label α is considered. Thus, in one configuration for label α , nodes are only allowed to keep their old labels or switch to a new label α . Each expansion operation is converted to a binary graph-cut problem that can be solved by the max-flow/min-cut algorithm.

Under the framework of GSR, we focus on the optimal estimation of input data, namely, input labeling probabilities and the adjacency graph. The general pipeline of our label-smoothing approach is shown in Fig. 2. In Stage 1, the adjacency graph is constructed by optimal neighborhood selection so that the regularization is only influenced by the labels of relevant neighbors. In Stage 2, probabilistic label relaxation (PLR) is used to enhance the reliability of the input label probabilities. PLR relocates the probability over each label by incorporating spatial contextual information to achieve a new reasonable probability set. Finally, the optimal graph and reliable label probability set are fed into the global regularization function. The final smooth labels are achieved by the application of α -expansion.

Section 3.1 describes the details in Stage 1, which includes the definition and estimation of the optimal neighborhood for label smoothing. The principals of PLR are introduced in Section 3.2, which also corresponds to the content of Stage 2.

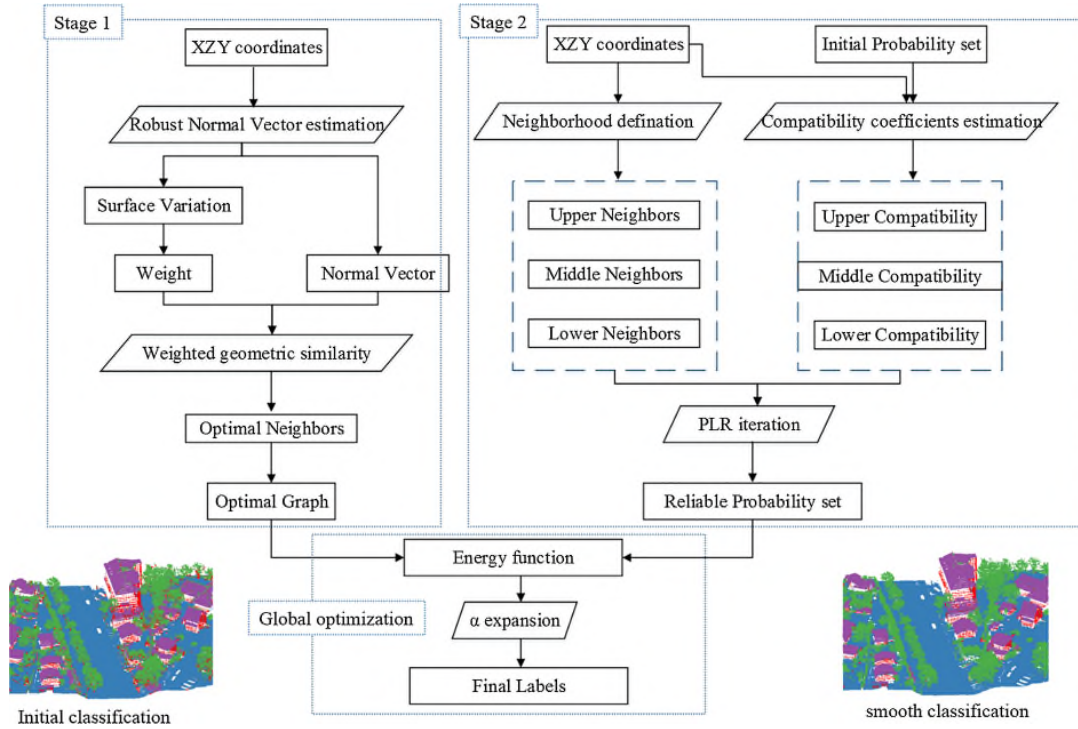
3.1. Optimal-neighborhood estimation

The optimal neighborhood in this approach intends to find all neighbors that have high possibilities of belonging to the same object with the considered point from within an initial search space. According to this optimal-neighborhood definition, each point can receive adequate influence from its neighbors. Thus, highly smooth classification results are achieved without losing any detail information for small objects.

The optimal neighborhood is estimated based on a weighted geometric similarity. The geometric similarity is measured by normal vectors and surface variations, as described in Sections 3.1.1 and 3.1.2 presents the weight-value allocation of each neighbor, which depends on the local surface variation in the considered point and the candidate neighboring points.

3.1.1. Fast and robust normal estimation based on a maximum consistent set

To provide reliable geometric attributes for the optimal-neighborhood selection, a robust normal estimation method is utilized based on a maximum consistency with minimum distance (MCMD). MCMD, which was proposed by Nurunnabi et al. (2015), couples the idea of using a point-to-plane orthogonal distance (OD) and surface variation along the normal (σ). However, MCMD is very time consuming because it requires a large number of repetitions of random sampling to obtain a sufficient initial subsets of points, which are used to achieve the most reliable and homogenous subsets in the local neighborhood. To reduce



General pipeline diagram of label smoothing.

Algorithm for the fast computation of robust normal vectors.

Step 1	Uniform sampling on the local point set to achieve the initial subsets of points; one subset can be recorded as $S(P)$
Step 2	For each initial subset $S(P)$, a plane is fitted by PCA. The OD of each remaining point p_i in the entire local neighborhood to this fitted plane is calculated and denoted as $OD(p_i)$. Then, the ODs are sorted as follows: $ OD(p_1) \leq OD(p_2) \leq \dots \leq OD(p_h) \leq \dots \leq OD(p_n) $
Step 3	Points with the h smallest ODs are selected as the h -subset, and plane fitting is performed again over points in this h -subset. Afterwards, the surface variation along the normal of the fitting plane is calculated (σ) and added to the list of $S(\sigma)$, and the h -subset points are also recorded as $S(p_h)$
Step 4	Repeat Step 2 to 3 until all the initial subsets are involved in the computation. The minimal σ is selected from the $S(\sigma)$, and the corresponding h -subset $S(p_h)$ is considered as the maximum consistent dataset in this local neighborhood
Step 5	Finally, the robust normal vector and surface variation along the normal (σ) of this local neighborhood is calculated from the selected h -subset points

the repetition times, we use uniform sampling that divides 3D points into voxels instead of random sampling. Each candidate initial subset consists of points within each voxel, and the number of repetitions is the number of voxels. The algorithm for the robust normal estimation can be found in Table 1.

Reliable normal and local surface variations in each point are computed based on fast MCMD and then used for the optimal-neighborhood estimation in the next section.

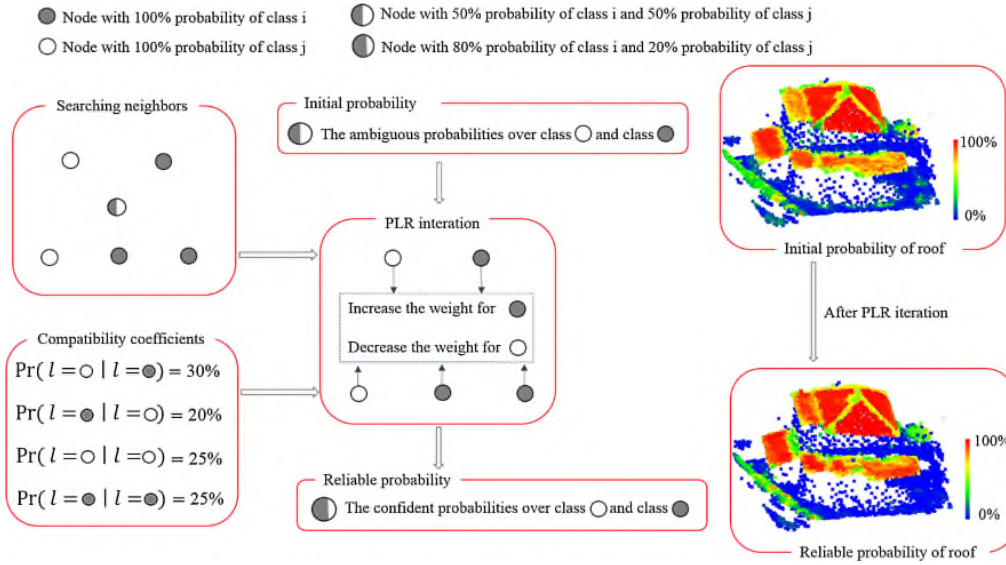
3.1.2. Optimal neighborhood

For the label-smoothing task, the optimal neighborhood is estimated by weighted geometric similarity to involve all neighboring points that potentially belong to the same object with the concerned points. Here, the geometric similarity is measured by the angle between the normal vectors and point-to-plane orthogonal distances, while the weights are determined by the local surface variations (σ). Points in vegetation areas have rather arbitrary normal vectors and relatively random spatial positions, so using equal-weight geometric similarity measurements produces a poor neighborhood for vegetation points, resulting in a lack of sufficient consistent neighborhood information for label smoothing. Inspired by the scaled normal-vectors technique that was proposed in Vosselman (2013), we assign different weights for points based on their

roughness.

Let p_0 denote the point of interest. At the beginning, a rough initial and relatively large neighborhood N_{p_0} of p_0 is estimated, which can be performed by the k nearest neighbors or fixed-range searching method. Each point $p_i \in N_{p_0}$ is a candidate for the optimal neighbors of p_0 . $Weight(p_0, p_i)$ denotes the weight that is jointly determined by p_0 and its neighboring candidate point p_i and is defined as in Eq. (6). $\sigma(p_0)$ is the local surface variation in the point p_0 . If $\sigma(p_0)$ is smaller than a certain threshold THR_σ , which means that p_0 may belong to a planar object, the weight values for all neighbors of p_0 equal 1. Otherwise, if $\sigma(p_0)$ is larger than the threshold THR_σ , the point p_0 is assumed to belong to non-planar objects. Thus, the weight value will be larger than 1 to increase the geometric similarity between the point p_0 and its neighbors. Additionally, the weight values vary by the $\sigma(p_i)$ value of neighbors in non-planar areas. Planar points are avoided during selection of the optimal neighborhood of the non-planar point by Eq. (6).

Given the robust normal vector and surface variation in each point, for $p_i \in N_{p_0}$, the weighted geometric similarity between p_i and p_0 is measured by Eq. (7). nv_p denotes the normal vector of point p and $p = [x_p, y_p, z_p]$ denotes the 3D coordinates of point p . THR_σ is the threshold of the normal vector-angle change and THR_d is the threshold of the local point-to-plane orthogonal distance. The candidates from the



Workflow for PLR.

initial neighborhood that satisfy Eq. (7) are collected as the optimal neighbors of the concerned point:

$$Weight(p_0, p_i) = \begin{cases} 1 & \sigma(p_0) \leq THR_\sigma \\ e^{\sigma(p_0) \cdot e^{\sigma(p_i)}} & \sigma(p_0) > THR_\sigma \end{cases} \quad (6)$$

$$Weight(p_0, p_i) \cdot nv_{p_0} \cdot nv_{p_i} \geq \cos(THR_\alpha) \text{ and } Weight(p_0, p_i) \cdot |(p_0 - p_i) \cdot nv_{p_0}| \leq THR_d \quad (7)$$

3.2. Probabilistic label relaxation

Point clouds that are scanned from real scenes exhibit statistical spatial rules, and the human visual system appears to have evolved to exploit such statistical regularities. Thus, we introduce probabilistic label relaxation (PLR) to embed spatial contextual constraints in label-smoothing processing.

PLR is an iterative procedure that updates the label probabilities on each point by using the label probabilities from the neighborhood; the PLR workflow is displayed in Fig. 3. In the initial probabilistic classification results, the ambiguous probability over each class cannot provide a confident label for the node (shown as a half-filled circle in the “Initial probability” block in Fig. 3). By considering the neighborhood context, more confident and reliable label probabilities can be iteratively updated based on PLR (shown as mostly filled circles in the “Reliable probability” block in Fig. 3). The neighborhood influences are numerically computed based on the neighborhood function, which requires defining the neighborhood and corresponding compatibility coefficients. An example of real LiDAR points’ probabilities before and after PLR is presented in the right column of Fig. 3.

The concept of PLR is described in Section 3.2.1. Then, the neighborhood function for the point cloud is defined in Sections 3.2.2 and 3.2.3 presents how to estimate compatibility coefficients from the initial classification.

3.2.1. Probabilistic label-relaxation definition

Given an initial probabilistic classification of points, let the probability for a point p that belongs to class c_i be denoted by

$$Pr_p^k(l_p = c_i) \quad c_i \in C \quad C = [c_1, \dots, c_N] \quad (8)$$

where l_p is the label of point p and N is the total number of classes. Based on PLR, an iterative probability-updating process is conducted by the following rule (John and Xiuping, 2006):

$$Pr_p^{k+1}(l_p = c_i) = \frac{Pr_p^k(l_p = c_i) \cdot Q_p^k(l_p = c_i)}{\sum_{c_j \in C} Pr_p^k(l_p = c_j) \cdot Q_p^k(l_p = c_j)} \quad (9)$$

where k is the iteration counter. $Q_p^k(l_p = c_i)$ is the neighborhood function of the point p , which allows neighbors to numerically influence the label probability over each class c_i . Let N_p denote the neighborhood of point p . Then, $Q_p^k(l_p = c_i)$ is defined by Eq. (10):

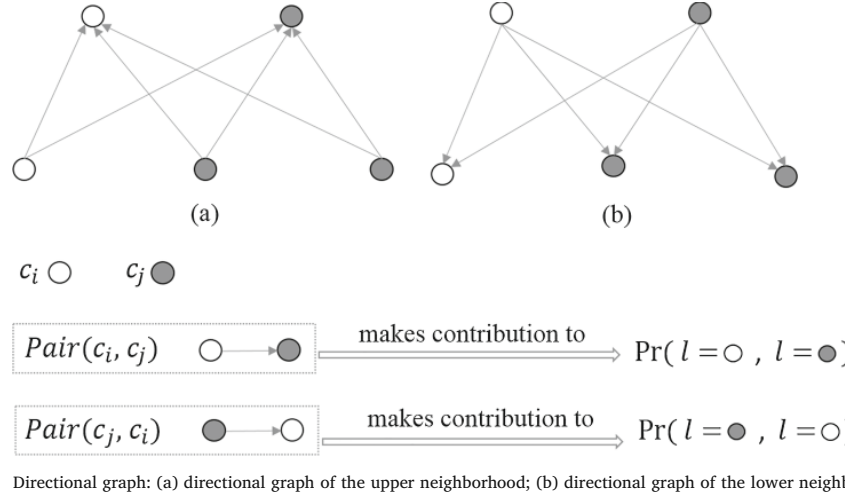
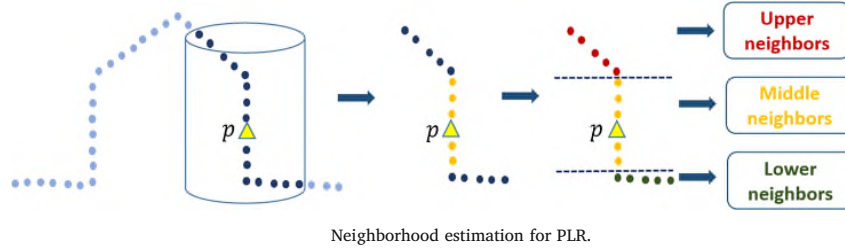
$$Q_p^k(l_p = c_i) = \sum_{q \in N_p, l_q^0 \neq l_p^0} \sum_{c_j \in C} Pr_{pq}(l_p = c_i | l_q = c_j) \cdot Pr_q^k(l_q = c_j) \quad (10)$$

where $Pr_{pq}(l_p = c_i | l_q = c_j)$ is the compatibility coefficient, which is measured by the conditional probability that point p is assigned to class c_i given that a neighboring point q belongs to class c_j . l_p^0 and l_q^0 denote the initially assigned labels of points p and q , respectively. Thus, only neighbors whose initially assigned labels differ from that of point p are allowed to influence the label probabilities of point p . This approach prevents the probability confidences of large wrongly labeled regions from becoming enhanced. The estimation of the neighborhood function in Eq. (10) and the contextual neighborhood definition are introduced in Section 3.2.2.

3.2.2. Neighborhood function

Differently from 2D images, the logical spatial relationships between a point and its neighbors must be clearly defined. A cylindrical neighborhood that consists of upper neighbors, middle neighbors and lower neighbors is selected for the neighborhood function. The neighborhood in the vertical direction can provide useful contextual information, e.g., the points beyond a vegetation point are likely vegetation rather than belonging to other classes.

The estimation for this neighborhood is described in Fig. 4. Given a point p , all the points within its cylindrical neighborhood are chosen, and the middle neighbors are selected based on the optimal-neighborhood estimation in Section 3.1. For the remaining points, the upper neighbors and lower neighbors are determined by their height differences compared to the point p . Thus, the neighborhood function in Eq. (10) can be written as



$$Q_p^k(l_p = c_i) = \sum_{q \in N_p^u, l_q^0 \neq l_p^0} \sum_{c_j \in c} Pr_{pq}^u(l_p = c_i | l_q = c_j) \cdot Pr_q^k(l_q = c_j) + \sum_{q \in N_p^m, l_q^0 \neq l_p^0} \sum_{c_j \in c} Pr_{pq}^m(l_p = c_i | l_q = c_j) \cdot Pr_q^k(l_q = c_j) + \sum_{q \in N_p^l, l_q^0 \neq l_p^0} \sum_{c_j \in c} Pr_{pq}^l(l_p = c_i | l_q = c_j) \cdot Pr_q^k(l_q = c_j) \quad (11)$$

where N_p^u, N_p^m , and N_p^l denote upper neighbors, middle neighbors and lower neighbors, respectively. $Pr_{pq}^u(l_p = c_i | l_q = c_j), Pr_{pq}^m(l_p = c_i | l_q = c_j)$, and $Pr_{pq}^l(l_p = c_i | l_q = c_j)$ are the compatibility coefficients between the point p and its upper, middle and lower neighbors, respectively. Based on Eq. (11), the neighbors' label probabilities are propagated through the corresponding compatibility coefficients. A reliable label-probability set can then be deduced from the neighborhood.

3.2.3. Compatibility coefficients

The compatibility coefficients between label pairs are expressed by the matrix R_{pq} :

$$R_{pq} = \begin{bmatrix} Pr_{p,q}(l_p = c_1 | l_q = c_1) & \cdots & Pr_{p,q}(l_p = c_N | l_q = c_1) \\ \vdots & \ddots & \vdots \\ Pr_{p,q}(l_p = c_1 | l_q = c_N) & \cdots & Pr_{p,q}(l_p = c_N | l_q = c_N) \end{bmatrix}$$

where q belongs to a certain type of neighborhood of point p . R_{pq}^u, R_{pq}^m , and R_{pq}^l denote the compatibility matrix for point p and its upper, middle and lower neighbors (N_p^u, N_p^m, N_p^l), respectively.

If $q \in N_p^m$, we assume that points within the optimal neighborhood likely belong to the same class. R_{pq}^m is defined as the unit matrix:

$$R_{pq}^m = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}$$

When $q \in N_p^u$ or $q \in N_p^l$, in each row of the compatibility matrix, we allocate the equal probabilities to $Pr_{pq}(l_p = c_i | l_q = c_j)$ where the label

pair (c_i, c_j) is compatible; the remaining $Pr_{pq}(l_p = c_i | l_q = c_j)$ values are assigned to 0 where the label pair (c_i, c_j) is not constrained by each other. Thus, each neighboring compatible label pair has equal influence on a certain type of class. The sum of each row of R_{pq}^u or R_{pq}^l equals 1.

Based on the research by Kumar and Dikshit (2017), compatible label pairs can be explored by mutual information (MI). MI estimates the contribution from the information of label c_i to the label c_j and is denoted by $MI(c_i, c_j)$. A high value of $MI(c_i, c_j)$ means a high positive correlation between two labels (c_i, c_j) . The MI is computed by Eq. (12):

$$MI(c_i, c_j) = \ln \left(\frac{Pr(l = c_i, l = c_j)}{Pr(l = c_i) \cdot Pr(l = c_j)} \right) \quad (12)$$

$Pr(l = c_i)$ is the probability of labelling any point as c_i and $Pr(l = c_i, l = c_j)$ is the joint probability of a pair of neighboring points with labels c_i and c_j . Based on the initial labels, these probabilities are determined through Eq. (13):

$$Pr(l = c_i, l = c_j) = \frac{n_{c_i, c_j}}{n}, \quad Pr(l = c_i) = \frac{n_{c_i}}{n}, \quad Pr(l = c_j) = \frac{n_{c_j}}{n} \quad (13)$$

In Eq. (13), n is the total number of points, n_{c_i} is the number of points with label c_i , and n_{c_i, c_j} is the number for the directional pair (c_i, c_j) that connects a node of label c_i to a node of label c_j . The directionality of pair (c_i, c_j) in the upper and lower neighborhood space is illustrated in Fig. 5, and different labels c_i and c_j are represented as unfilled and filled circles, respectively. Obviously, the number for the directional pair (c_i, c_j) (denoted by n_{c_i, c_j}) is different from the number of pair (c_j, c_i) (denoted by n_{c_j, c_i}) in Fig. 5(a) and (b). Moreover, n_{c_i, c_j} and n_{c_j, c_i} contribute to the joint probability $Pr(l = c_i, l = c_j)$ and $Pr(l = c_j, l = c_i)$, respectively. Thus, $Pr(l = c_i, l = c_j) \neq Pr(l = c_j, l = c_i)$.

After the estimation of the joint probability of label pairs, the corresponding MI of the label pairs can be achieved through Eq. (12). Then, the compatibility coefficients are calculated as in Eq. (14) based on MI. For a certain class c_j , $n_{clp}(c_j)$ is the number of compatible label

pairs of class c_j .

$$Pr_{pq}(l_p = c_i ||_q = c_j) = \begin{cases} \frac{1}{n_{clp}(c_j)} & MI(c_i, c_j) > 0 \\ 0 & MI(c_i, c_j) \leq 0 \end{cases} \quad (14)$$

We applied this contextually based label-smoothing method to two urban 3D point clouds. The datasets are described in Section 4.1, and the initial classification and label-smoothing method are conducted in Section 4.2. The results are evaluated in Section 4.3.

4.1. Datasets

To assess the applicability of our approach, two airborne LiDAR datasets were used. The first was a fully labelled airborne LiDAR dataset of Vienna, Austria (Tran et al., 2018). The selected area is $100 \times 100 \text{ m}^2$, and the density varies from 5 to 75 points/ m^2 . This area represents a complex urban scene, including a mixture of high and low vegetation, high-rise and small detached houses, and flat and sloped ground. Four domain classes were categorized for the Vienna dataset: *ground, vegetation, roofs, and façades*.

The second was an airborne LiDAR dataset of Vaihingen, Germany, which is a component of the ISPRS 3D semantic labelling context benchmark (Rottensteiner et al., 2012). The point density is between 4 and 7 points/ m^2 . Multiple echoes and intensities are recorded on each point. Three test areas with reference labels are available. Test area 1, “Inner City”, is located in the center of Vaihingen and contains dense historic buildings with complex shapes alongside some trees. Test area 2, “High Rises”, consists of a few high-rising residential buildings that are surrounded by trees. Test area 3, “Residential Area”, has small detached houses. For the Vaihingen dataset, the following 8 classes were discerned: *grassland, roads, cars, low vegetation, gable roofs, flat roofs, high vegetation, and façades*.

In order to get an impression of the datasets, the original datasets of Vienna and Vaihingen colorized by intensity are displayed in Fig. 6. Meanwhile, the percentage distributions of each class in both datasets are shown in Table 2.

4.2. Initial classification and label smoothing

We used a random forest (RF) as the initial probabilistic classifier, but our label-smoothing approach is not limited to RFs, and any probabilistic classifier could be applied to obtain the initial classification. A set of 18 features were extracted from each dataset for initial classification; a detailed description of the features can be found in Li et al. (2017). For the Vaihingen dataset, the extra feature “Intensity” was included in the feature vector to differentiate grassland and roads. To train the RF classifiers for each dataset, we randomly selected 100 and 2000 training samples per class from the Vienna dataset and Vaihingen dataset, respectively.

For the PLR method, a cylindrical neighborhood was built with a radius of 1 m and height of 4 m. The optimal neighbors that were selected from this cylindrical neighborhood were used to estimate the middle-neighborhood function. The remaining neighboring points with heights beyond the concerned point were used to generate the upper-neighborhood function; likewise, a lower-neighborhood function was built. The number of iterations was empirically set as four. After the PLR, the new labeling probabilities were expected to be more consistent with the prior knowledge of neighborhood dependencies.

For the robust normal estimation, the voxel size was set as 0.5 m in the X, Y, and Z directions. To construct the optimal adjacency graph G_{opt} , k ($k = 30$) nearest points were first selected as the initial candidate neighbors, the threshold of the orthogonal distance between the point to the local plane was set as 0.1 m, and the threshold of the angle

between normal vectors was set as 10° . Then, the optimal neighbors were chosen from the initial candidate neighbors as described in Section 3.1. The optimal graph was created by connecting each point to its optimal neighbors.

The new reliable probabilities and the optimal graph were fed into the regularization optimization function, and final smooth labels were obtained by the solution of α -expansion.

To separately test the effectiveness of PLR and the optimal graph, two label-smoothing techniques were applied on both LiDAR datasets, as described in Table 3:

4.3. Results

Figs. 7 and 8 show the label-smoothing results from GSR- G_{opt} and PLR + GSR- G_{opt} on the Vienna and Vaihingen airborne LiDAR datasets alongside the initial classification results (Figs. 7(a) and 8(a)–(c)) and the references (Fig. 7(d) and 8(j)–(l)). The quantitative evaluations of the label-smoothing results are shown in Tables 4 and 5.

From a visual inspection of the initial classification, both small and large wrongly labeled regions existed in the initial classification of the Vienna and Vaihingen datasets. Small wrongly labeled regions occurred on the edges of roofs and in vegetation areas with lower density in both datasets. Because of the feature similarity, the RF failed to assign correct labels to sections of sloped ground points, low flat-vegetation points, and small low-roof points. Thus, large wrongly labeled regions and wrongly labeled regions that lacked sufficient correct contextual interactions existed in the mentioned areas in both datasets. Moreover, eight classes were considered in the Vaihingen dataset. Scenarios with various types of labeling errors and various sizes and shapes of objects can be considered a rather challenging label-smoothing task.

We first present the GSR- G_{opt} label-smoothing results in Section 4.3.1 to verify the optimal graph’s effectiveness. Then, the PLR + GSR- G_{opt} label-smoothing results are explained in Section 4.3.2 to examine the necessity of adding PLR.

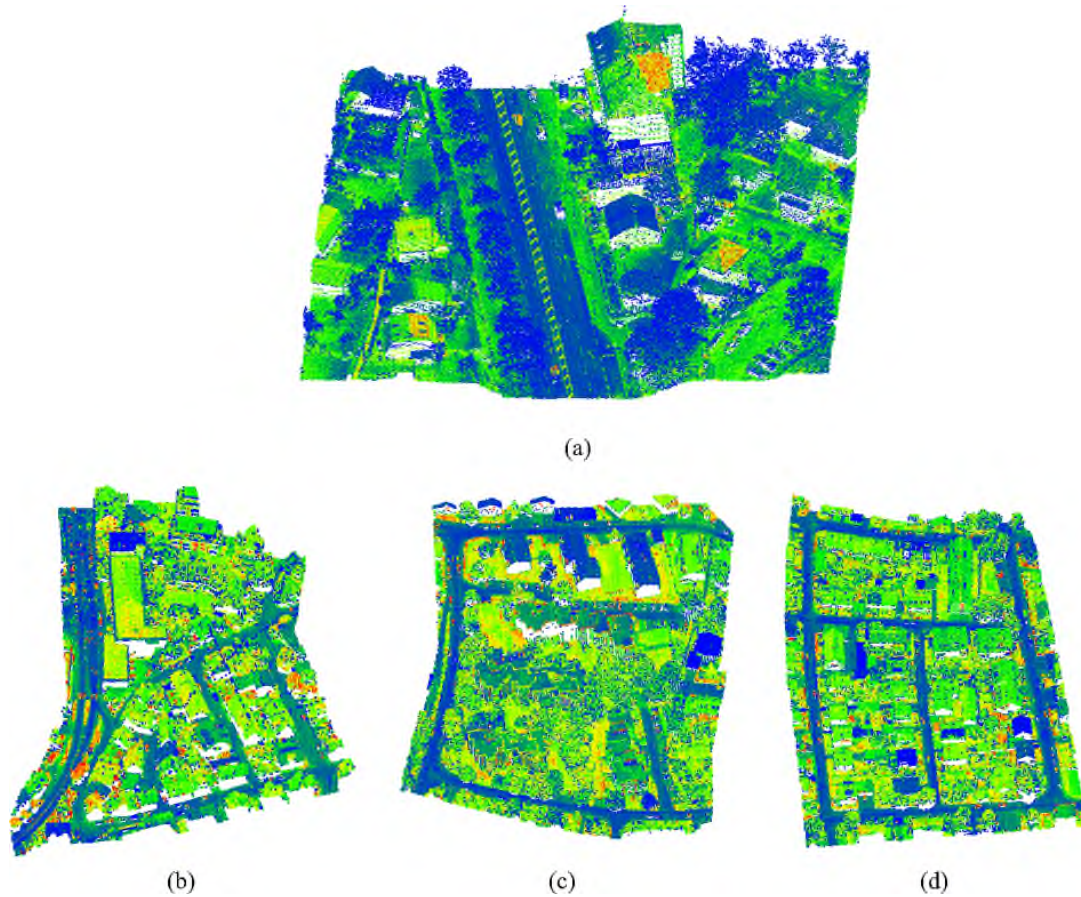
4.3.1. GSR- G_{opt} labeling-smoothing results

From Tables 4 and 5, GSR- G_{opt} improved the overall accuracy (OA) to 95.31% and 83.52% for the Vienna and Vaihingen datasets, respectively. According to the optimal adjacency graph, most small and large wrongly labeled regions were assigned to the correct labels based on Fig. 7(b) and 8(d)–(f). After the GSR- G_{opt} label smoothing of the Vienna dataset, the classification accuracy of ground, vegetation and roofs increased by 6.20%, 3.42% and 5.21%, respectively. A slight accuracy loss (-0.52%) was observed for the façade class in the Vienna dataset. No entire façade object was lost; only façade points that were located on conjunctions with ground were inevitably smoothed to ground points. Correct façade objects could still be extracted and the façade accuracy remained unchanged in terms of object level. The accuracies of all classes for the GSR- G_{opt} label smoothing on the Vaihingen dataset improved regardless of the size of the objects. Thus, the optimal adjacency graph could produce a better smooth classification result without losing any detailed information of small objects.

4.3.2. PLR + GSR- G_{opt} label-smoothing results

After GSR- G_{opt} label smoothing, wrongly labeled regions still remained because of a lack of correct label interactions from the neighborhood consistency. By updating the labeling probabilities based on the neighborhood dependencies, the correct label probability would increase and the probabilities of other labels would decrease. Thus, the label probabilities became more reliable and logical within the spatial context. Therefore, PLR + GSR- G_{opt} effectively solves the lack of a reliable neighborhood consistency. The PLR + GSR- G_{opt} label-smoothing results are shown in Fig. 7 (c) and 8(g)–(i).

Fig. 9 provides more details of the object classification after label smoothing. Fig. 9(a) is from the Vienna dataset’s RF classification. The wrongly labeled regions, which are shown in three rectangles, are



Airborne LiDAR dataset colorized by intensity: (a) Vienna airborne LiDAR dataset; (b)–(d) test area 1, test area 2 and test area 3 from the Vaihingen airborne LiDAR dataset.

supposed to be low vegetation, but most of these points were misclassified as ground or roofs by the RF. After GSR- G_{opt} label smoothing, the wrongly labeled regions retained their wrong labels of ground (Fig. 9(d), left rectangle) and roofs (Fig. 9(d), middle rectangle); only some of the regions were correctly modified to low vegetation (Fig. 9(d), right rectangle) because of the higher probabilities over ground and roofs in the neighborhood. In contrast, the wrongly labeled regions in the above three sites were all correctly classified into vegetation by PLR + GSR- G_{opt} label smoothing (Fig. 9(g)). Another example is from test area 3 in the Vaihingen dataset. Fig. 9(b) shows that half of the low flat-roof points were misclassified into low vegetation (Fig. 9(b), left rectangle), with some misclassified into high vegetation (Fig. 9(b), right rectangle) in the RF classification result. GSR- G_{opt}

classified these points into low vegetation (Fig. 9(e)) because this solution is the best for the regularization function given the input RF probabilities. By replacing the input probabilities with PLR, the wrongly labeled regions in the rectangles were corrected to flat roofs (Fig. 9(h)). The nearby gable roof objects favored a higher weight value over the probabilities of gable roofs and flat roofs, so the probabilities over gable roofs and flat roofs would be enhanced during PLR for these wrongly labeled points. Therefore, the label probabilities were more consistent with the contextual knowledge and could provide more reliable probability inputs for graph-structured regularization optimization. Similar examples can be found in Fig. 9(c), (f), and (i), which show test area 1 from the Vaihingen dataset.

From the classification evaluation in Tables 4 and 5, the OA

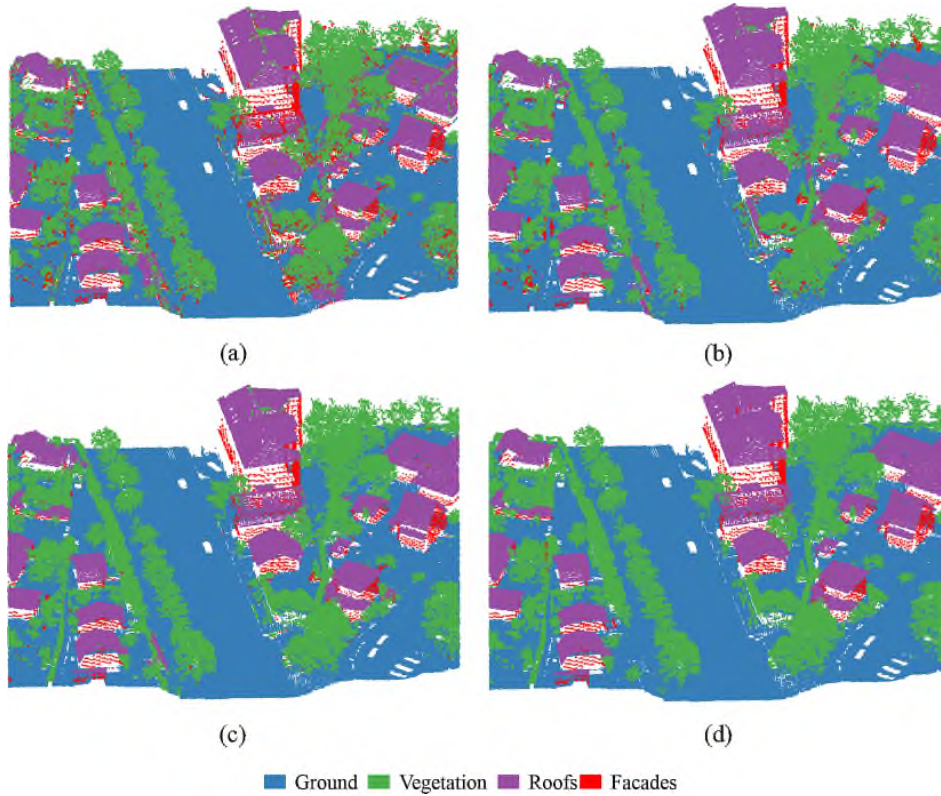
Percentage distribution of each class in the dataset.

(a) Vienna airborne LiDAR dataset				
Class	Ground	Vegetation	Roofs	Façades
Percentage	65.50%	19.40%	13.67%	1.43%

(b) Vaihingen airborne LiDAR dataset								
Class	Grassland	Roads	Cars	Low vegetation	Gable roofs	Flat roofs	High vegetation	Façades
Percentage	22.68%	28.01%	0.77%	6.21%	15.76%	5.59%	17.55%	3.44%

List of label-smoothing methods.

Label-smoothing method	Input probability set	Graph estimation		Regularization
		Initial neighborhood selection	Optimization	
GSR-G _{opt}	Derived from RF	KNN with k of 30	Proposed optimal neighborhood estimation	α -expansion
PLR + GSR-G _{opt}	Derived from RF and modified by PLR	KNN with k of 30	Proposed optimal neighborhood estimation	α -expansion



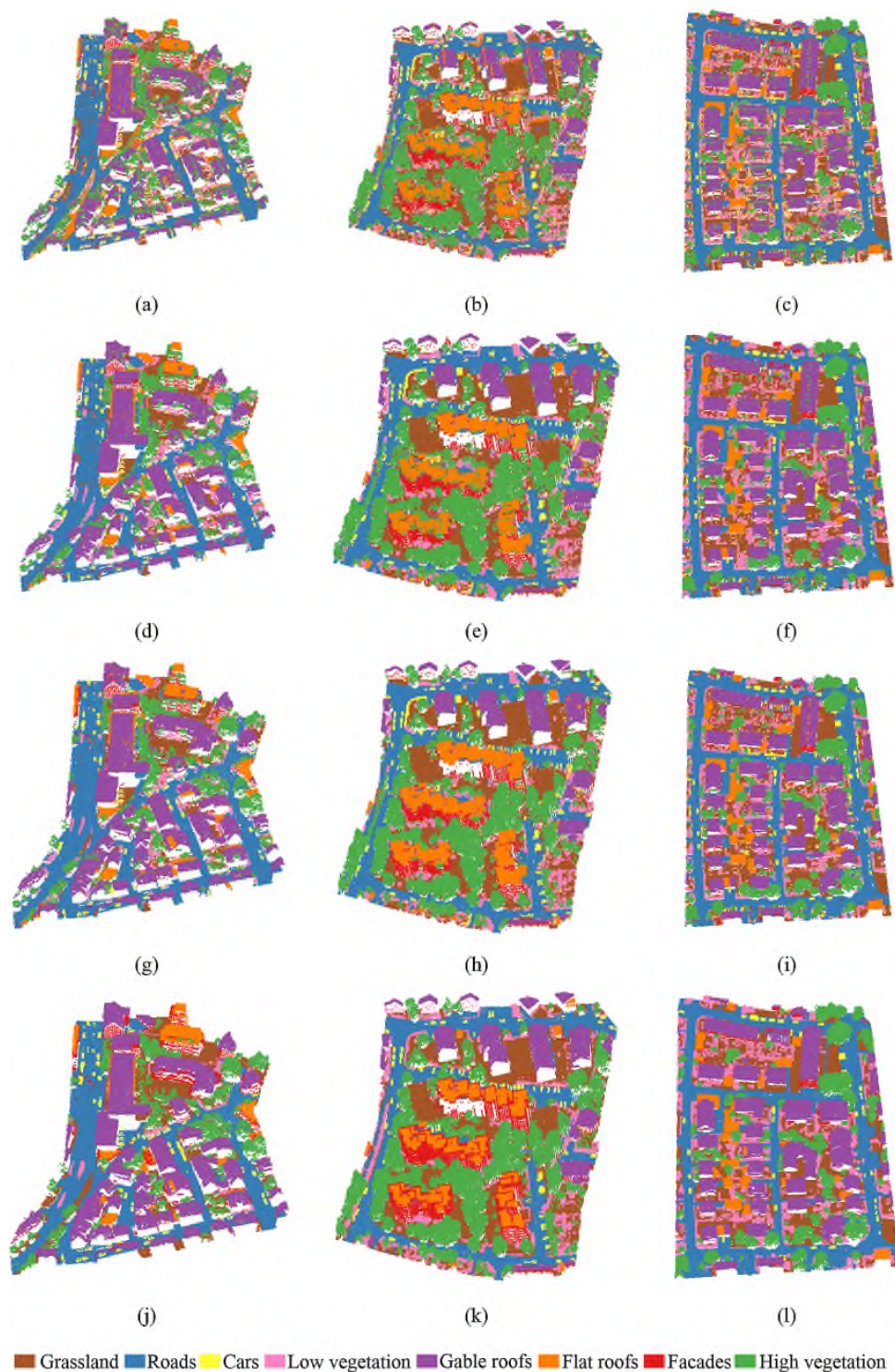
Classification results of the Vienna dataset: (a) initial classification result from the RF; (b) label-smoothing result from GSR-G_{opt}; (c) label-smoothing result from PLR + GSR-G_{opt}; (d) reference.

increased by 7.01% and 6.88% after PLR + GSR-G_{opt} label smoothing on the Vienna and Vaihingen datasets, respectively. For the Vienna dataset’s label smoothing, all the classes’ accuracies significantly improved except for façades; the reason was explained in Section 4.3.1. The vegetation classification-accuracy improvement (9.95%) from PLR + GSR-G_{opt} was much higher than the accuracy improvement (3.42%) from GSR-G_{opt}. For the more complicated classification task of Vaihingen, most classes achieved obvious improvements after PLR + GSR-G_{opt} label smoothing; in particular, high vegetation’s accuracy increased by 11.35%, 6.55% higher than the GSR-G_{opt} label-smoothing result. The accuracy of the less dominant class of façades had a slight improvement of 0.96%. Low vegetation’s classification accuracy decreased by 5.54%, worse than the GSR-G_{opt} label-smoothing result. According to the confusion matrix after PLR + GSR-G_{opt} label smoothing, which can be found in the supplement, a large number of low-vegetation points were wrongly assigned into grassland (12.87%) and high vegetation (14.86%). The complex urban environment in the areas of interest in Vaihingen city also inhibits us from distinguishing these points even by manual classification. A few erroneous labels may exist in the Vaihingen dataset, so some of the accuracy decrease in the Vaihingen-dataset classification could be explained by a few of the

errors in the reference. Another reason is that low vegetation usually lies very close to grassland or high vegetation, so these points are often confused with each other.

Given the GSR-G_{opt} label-smoothing results, large wrongly labeled regions could be corrected by collecting a large number of relevant neighbors, while small objects such as façades were preserved by the adaptive neighborhood. The accuracy can be further improved in terms of the object level by using this approach alongside PLR. Visual inspection and the quantitative evaluation of PLR + GSR-G_{opt} both indicated that more points were corrected based on the prior neighborhood dependencies. Good improvement could be achieved by PLR + GSR-G_{opt} label smoothing, regardless of the number of target classes.

In this section, we compare the PLR + GSR-G_{opt} label-smoothing method with GSR by using graphs that were constructed by KNN and a graph of minimal entropy in Section 5.1. Comparisons with local-smoothing methods are presented in Section 5.2. Our classification results are also compared with three of the results that were submitted to



Classification results of the Vaihingen dataset: (a)–(c) initial classification results from the RF; (d)–(f) label-smoothing results from GSR- G_{opt} ; (g)–(i) label-smoothing results from PLR + GSR- G_{opt} ; (j)–(l) references.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
 The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Classification accuracy of the Vienna dataset.

Class	Initial accuracy	Accuracy after label smoothing			Accuracy changes
	RF	GSR-G _{opt}	PLR + GSR-G _{opt}	GSR-G _{opt}	PLR + GSR-G _{opt}
Ground	90.65%	96.85%	97.33%	+6.20%	+6.68%
Vegetation	86.40%	89.82%	96.35%	+3.42%	+9.95%
Roofs	93.36%	98.57%	98.70%	+5.21%	+5.34%
Façades	88.03%	87.51%	87.22%	-0.52%	-0.81%
OA	90.16%	95.31%	97.17%	+5.15%	+7.01%

Classification accuracy of the Vaihingen dataset.

Class	Initial accuracy	Accuracy after label smoothing			Accuracy changes
	RF	GSR-G _{opt}	PLR + GSR-G _{opt}	GSR-G _{opt}	PLR + GSR-G _{opt}
Grass	70.52%	76.78%	77.98%	+6.27%	+7.46%
Roads	86.40%	93.52%	93.63%	+7.12%	+6.96%
Cars	80.61%	80.80%	80.80%	+0.19%	+0.19%
Low vegetation	63.17%	64.57%	57.63%	+1.40%	-5.54%
Gable Roofs	83.36%	89.62%	90.53%	+6.26%	+7.17%
Flat Roofs	84.65%	90.09%	91.75%	+5.45%	+7.10%
High vegetation	74.92%	79.47%	86.27%	+4.80%	+11.35%
Façades	57.94%	59.19%	58.90%	+1.25%	+0.96%
OA	77.74%	83.52%	84.62%	+5.78%	+6.88%

the ISPRS benchmark in Section 5.3. Section 5.4 discusses the effect of the initial neighborhood selection. Finally, the effect of the compatibility coefficients is discussed in Section 5.5.

5.1. Comparison of the graph-estimation methods

The effectiveness of PLR was already presented in Section 4.3.2 by a comparison of the PLR + GSR-G_{opt} and GSR-G_{opt} label-smoothing methods, so we investigate the following graph-structured regularization methods in Table 6 to assess the advantage of the optimal graph.

Fig. 10 shows an accuracy comparison of regularization label smoothing that was structured by various graphs for the Vienna and Vaihingen datasets. Two different sites were selected from the Vienna (Fig. 11(a)–(d)) and Vaihingen datasets (Fig. 11(e)–(h)) to provide detailed information of the classification improvements. Neighborhoods with small k values failed to modify large wrongly labeled regions (Fig. 11(a) and (e)), where k refers to the number of nearest points. However, excessively large neighborhoods without selection in PLR + GSR-G_{k30} sacrificed the accuracy of small object (Fig. 11(b) and (f)). Neighborhoods that were estimated from the minimal entropy also could not build long-range interactions for large wrongly labeled regions (Fig. 11(c) and (g)).

As shown in Fig. 10(a), the improvement in the ground classification increased with the value of k ; in contrast, the loss in the façade classification accuracy increased with the value of k . Similar trends were also observed for the label-smoothing task of the Vaihingen dataset. The neighborhood that was computed from the minimal entropy also failed to avoid over-smoothing small objects. Neighborhoods that were based on the minimal entropy selected the optimal k so that the local neighborhood had the best planarity but could not exclude points of different objects. Therefore, the accuracies of small objects such as cars decreased after PLR + GSR-G_{entropy}. Nevertheless, PLR + GSR-G_{opt} slightly increased the accuracy of cars (+0.19%). Another significant accuracy loss was related to low vegetation; in contrast, our approach obtained a much lower accuracy loss for low vegetation (5.54%) compared to PLR + GSR-G_{k20} (36.40%), PLR + GSR-G_{k30} (50.73%) and PLR + GSR-G_{entropy} (18.26%).

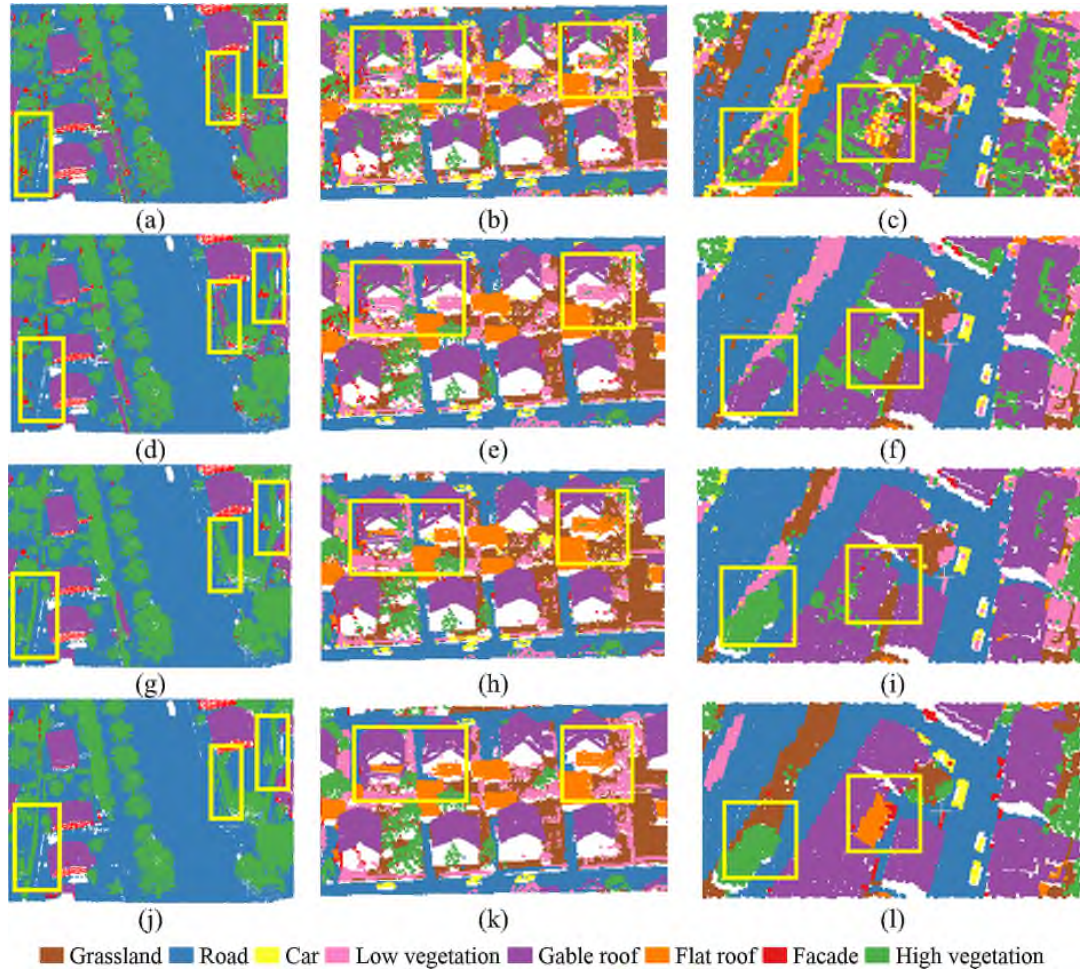
Additionally, the accuracy comparisons in Fig. 10 suggest that finding an exact optimal number of nearest points to build the graph is

difficult. When various types of label errors exist in a complex urban environment, regularizations that are structured by a KNN graph cannot correct large wrongly labeled regions and preserve small objects at the same time; however, PLR + GSR-G_{opt} exhibited much better performance with respect to all classes in both datasets.

5.2. Comparison to local label-smoothing methods

We consider five local-smoothing methods for comparison, namely, a majority filter, Gaussian filter, bilateral filter, edge-aware filter and local segmentation. Detailed explanations of the first four smoothing filters can be found in Schindler (2012), and local segmentation optimization based on graph-cutting is described in Guo et al. (2015). The neighborhoods for the local smoothing filters were defined as KNNs with k of 10.

According to the comparison in Fig. 12, PLR + GSR-G_{opt} outperformed all the local smoothing filters in terms of the OA. For the Vienna dataset, PLR + GSR-G_{opt} achieved much better improvements for the ground and vegetation classification than the other local smoothing filters. Local segmentation had a higher accuracy improvement over façades than the other methods, including PLR + GSR-G_{opt}; however, a trade-off occurred with an accuracy loss for roofs after local segmentation. In contrast to local-segmentation smoothing, all the other methods obtained an obvious enhancement for roofs. For the label smoothing on the Vaihingen dataset, PLR + GSR-G_{opt} only performed worse than the other local optimization methods in terms of cars and low vegetation because points that surrounded cars or low vegetation were mostly modified as cars or low vegetation by the local smoothing methods whether these points truly belonged to cars or low vegetation. Thus, fewer improvements were obtained in terms of the other classes. Moreover, except for local segmentation, the other local smoothing filters lost accuracy for façades, whereas PLR + GSR-G_{opt} slightly enhanced the façade classification accuracy, as did the local segmentation optimization. In particular, PLR + GSR-G_{opt} obtained a much higher increase in the high vegetation's accuracy (11.35%) than the local smoothing filters (2.80% from the majority filter, 3.68% from the Gaussian filter, 3.59% from the bilateral filter, 3.48% from the edge-aware filter, and 4.48% from local segmentation).



Details of the classification improvements: (a)–(c) initial classification results from the RF for the three sites; (d)–(f) classification results after GSR- G_{opt} label smoothing; (g)–(i) classification results after PLR + GSR- G_{opt} label smoothing; (j)–(l) references for the three sites.

5.3. Comparison with other published methods in the ISPRS benchmark

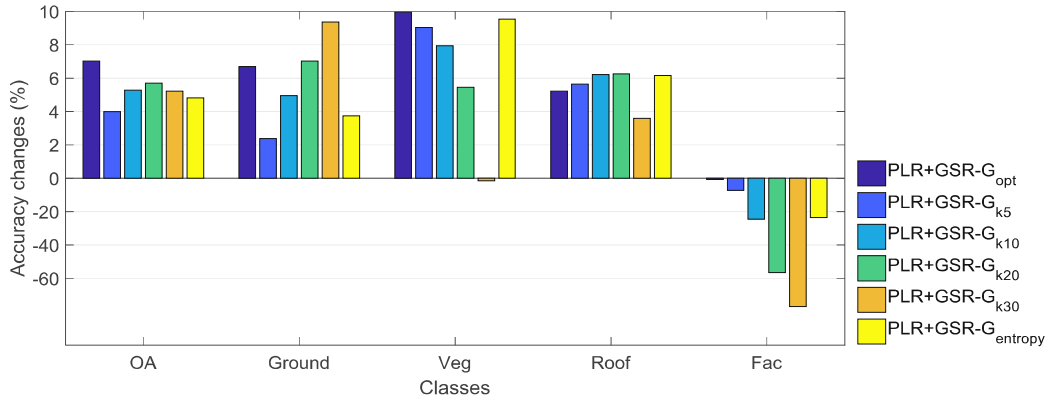
In this section, we compare our classification results with three of the results that were submitted to the ISPRS benchmark. The dataset that was used for the ISPRS benchmark evaluation is shown in Fig. 13. Obviously, the evaluation dataset was not completely covered by the three areas of interest that were used in our previous evaluation; a

nearby area was not included in our previous dataset. Thus, we used the RF model in Section 4.2 for the initial classification and PLR + GSR- G_{opt} to apply label smoothing for this area.

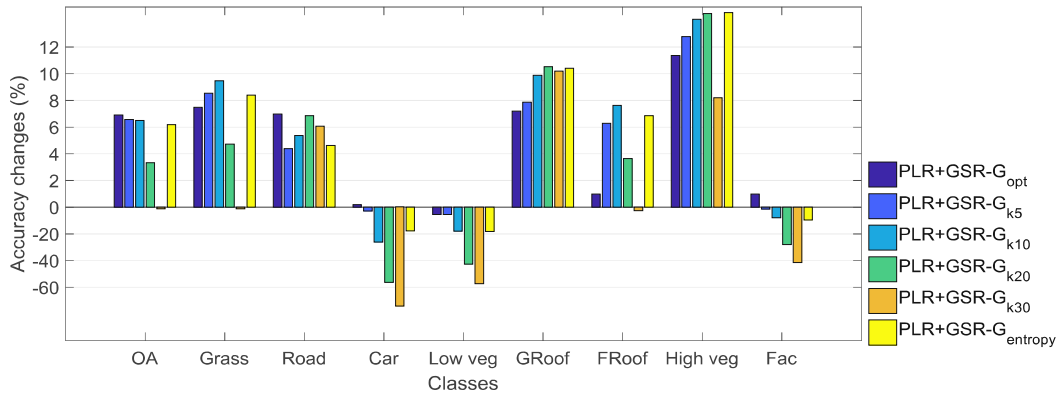
Three methods that had the best classification performances and attached papers to provide a detailed explanation were selected from the ISPRS 3D semantic labeling benchmark. The overall accuracy and correctness of each class are listed in Table 7. The methods for

GSR with the graphs that were estimated by various methods.

Label-smoothing method	Input probability set	Graph estimation		Regularization
		Initial neighborhood selection		
		Neighborhood definition	Parameters	
PLR + GSR- G_{k5}	Modified by PLR	KNN	$k = 5$	α -expansion
PLR + GSR- G_{k10}	Modified by PLR	KNN	$k = 10$	α -expansion
PLR + GSR- G_{k20}	Modified by PLR	KNN	$k = 20$	α -expansion
PLR + GSR- G_{k30}	Modified by PLR	KNN	$k = 30$	α -expansion
PLR + GSR- $G_{entropy}$	Modified by PLR	KNN	k ranges from 5 to 100 with an interval of 1	α -expansion
PLR + GSR- G_{opt}	Modified by PLR	KNN	$k = 30$	Proposed optimal neighborhood estimation



(a)

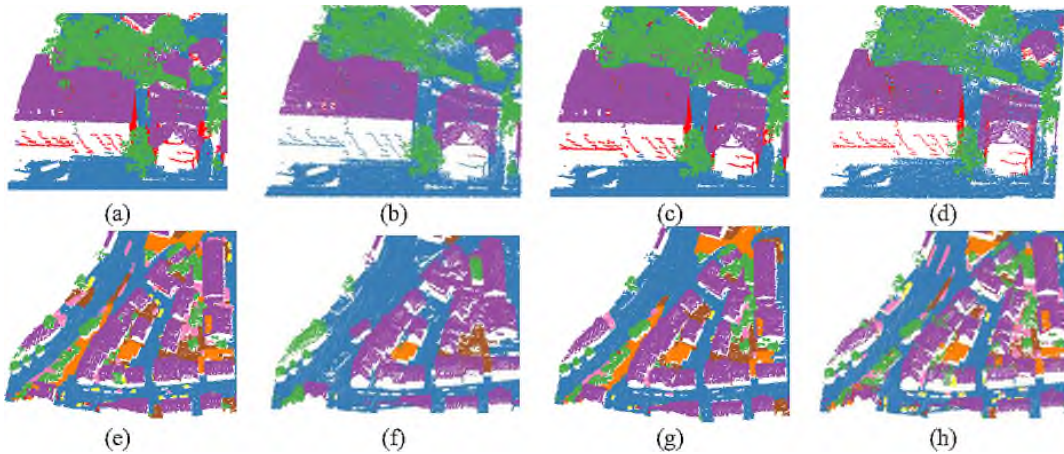


(b)

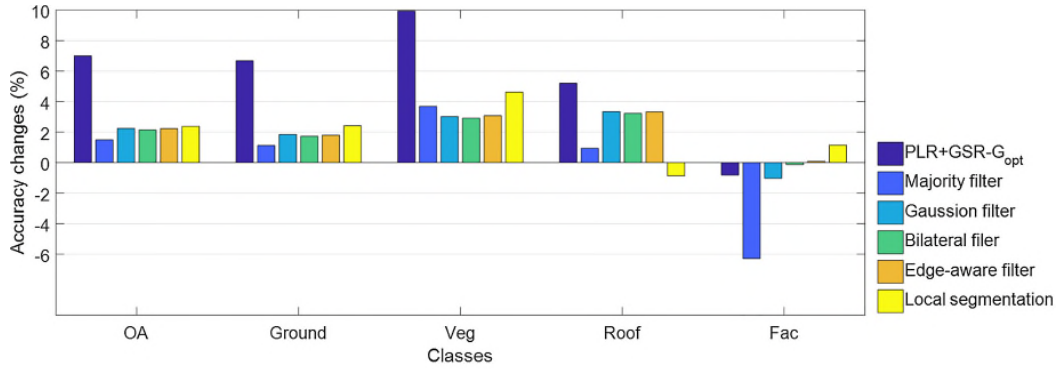
Comparison with various graph-estimation methods: (a) classification-accuracy changes in the Vienna dataset; (b) classification-accuracy changes in the Vaihingen dataset (OA: overall accuracy; Grass: grassland; Road: roads; Car: cars; Low veg: low vegetation; GRoof: gable roofs; FRoof: flat roofs; High veg: high vegetation; Fac: façades).

comparison were NANJ2, WhuY3, and LUH. NANJ2 (Zhao et al., 2018) designs a multi-scale convolutional neural network for classification. WhuY3 (Yang et al., 2017) uses a convolutional neural network-based method to extract high-level features and label 3D points. LUH

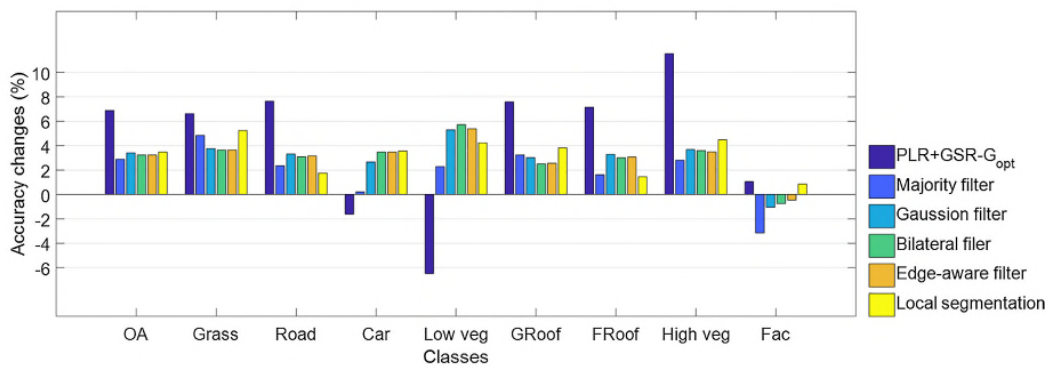
(Niemeyer et al., 2016) adopts hierarchical conditional random fields (HCRF) for classification. By concentrating on the common categories of the classes that we considered and the classes that were provided by the ISPRS benchmark, six classes were involved in this comparison



Comparison of different graph-construction methods: (a) and (e) GSR of the graph from KNN ($k = 5$); (b) and (f) GSR of the graph from KNN ($k = 30$); (c) and (g) GSR of the graph from KNN with k determined by the minimal eigenvalue entropy; (d) and (h) GSR of the graph from the optimal neighborhood.

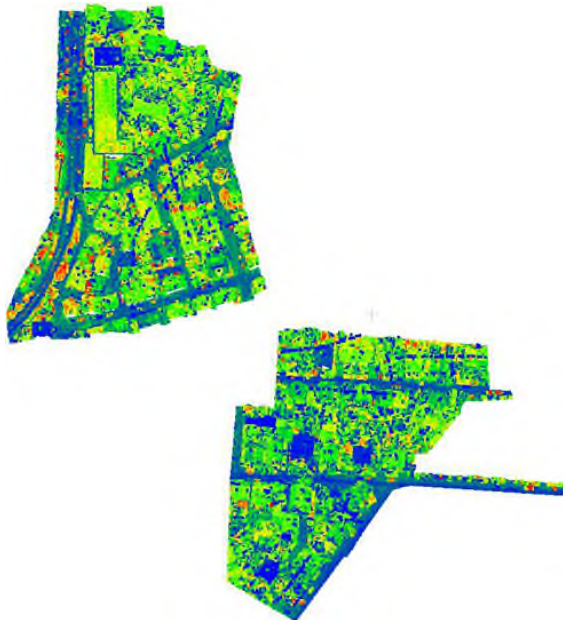


(a)



(b)

Comparison of local-smoothing methods: (a) classification-accuracy changes in the Vienna dataset; (b) classification-accuracy changes in the Vaihingen dataset.



Dataset that was used for evaluation in the ISPRS benchmark, colored by intensity.

Classification-accuracy comparison with other published methods in the ISPRS benchmark.

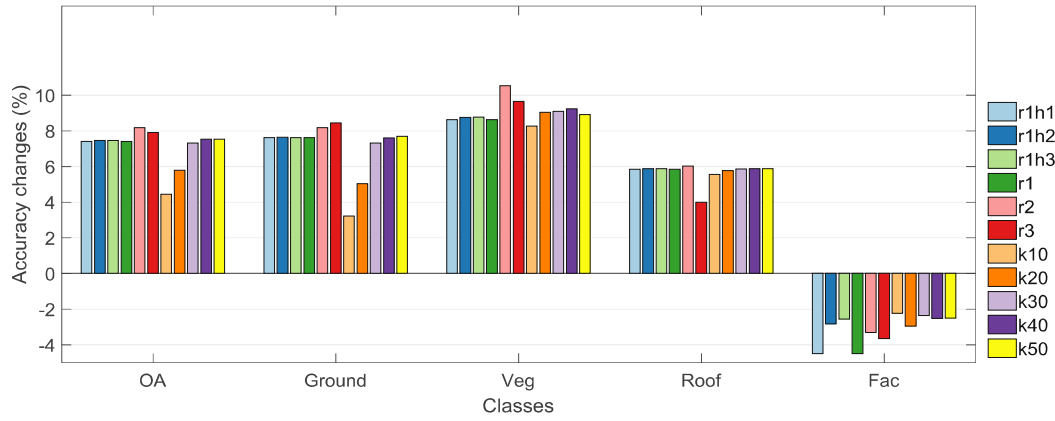
Class	NANJ2	WhuY3	LUH	PLR + GSR-G _{opt}
Grass	90.0%	80.9%	83.0%	69.3%
Roads	89.2%	88.4%	91.8%	94.6%
Cars	83.4%	58.4%	86.4%	67.0%
Roofs	95.7%	91.4%	97.3%	94.9%
Façades	47.6%	56.6%	52.4%	56.9%
High vegetation	88.3%	77.5%	87.4%	80.6%
OA	85.2%	82.3%	81.6%	81.5%

Classification-accuracy changes over the ISPRS evaluation dataset.

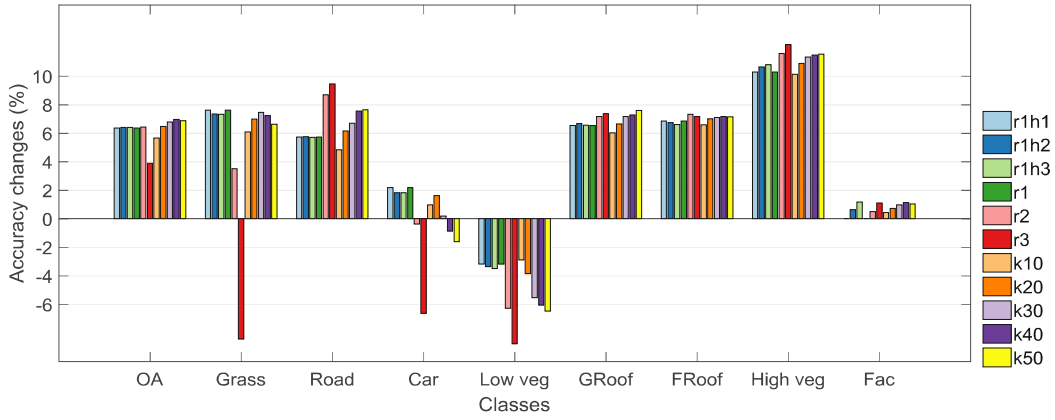
Class	Initial accuracy	Accuracy after label smoothing	Accuracy changes
			RF
Grass	58.7	69.3%	10.6
Roads	86.8	94.6%	7.8
Cars	63.5	67.0%	3.5
Roofs	84.4	94.9%	10.5
Façades	52.8	56.9%	4.1
High vegetation	69.1	80.6%	11.5
OA	73.2	81.5%	8.3

Graph-structured regularization with optimal graphs that were estimated from various initial neighborhood selections.

Label-smoothing method	Input probability set	Graph estimation			Regularization
		Initial neighborhood selection		Optimization	
		Neighborhood definition	Parameters		
r1h2	Modified by PLR	Cylinder	Radius = 1; Height = 2	Proposed optimal neighborhood estimation	α -expansion
r1h4	Modified by PLR	Cylinder	Radius = 1; Height = 4	Proposed optimal neighborhood estimation	α -expansion
r1h6	Modified by PLR	Cylinder	Radius = 1; Height = 6	Proposed optimal neighborhood estimation	α -expansion
r1	Modified by PLR	Sphere	Radius = 1	Proposed optimal neighborhood estimation	α -expansion
r2	Modified by PLR	Sphere	Radius = 2	Proposed optimal neighborhood estimation	α -expansion
r3	Modified by PLR	Sphere	Radius = 3	Proposed optimal neighborhood estimation	α -expansion
k10	Modified by PLR	KNN	$k = 10$	Proposed optimal neighborhood estimation	α -expansion
k20	Modified by PLR	KNN	$k = 20$	Proposed optimal neighborhood estimation	α -expansion
k30	Modified by PLR	KNN	$k = 30$	Proposed optimal neighborhood estimation	α -expansion
k40	Modified by PLR	KNN	$k = 40$	Proposed optimal neighborhood estimation	α -expansion
k50	Modified by PLR	KNN	$k = 50$	Proposed optimal neighborhood estimation	α -expansion



(a)



(b)

Comparison of different initial neighborhood-selection methods: (a) classification-accuracy changes in the Vienna dataset; (b) classification-accuracy changes in the Vaihingen dataset.

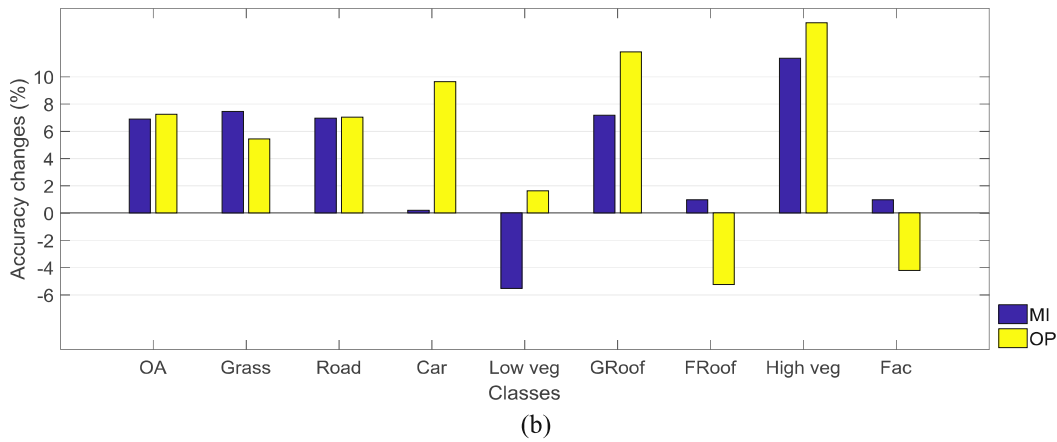
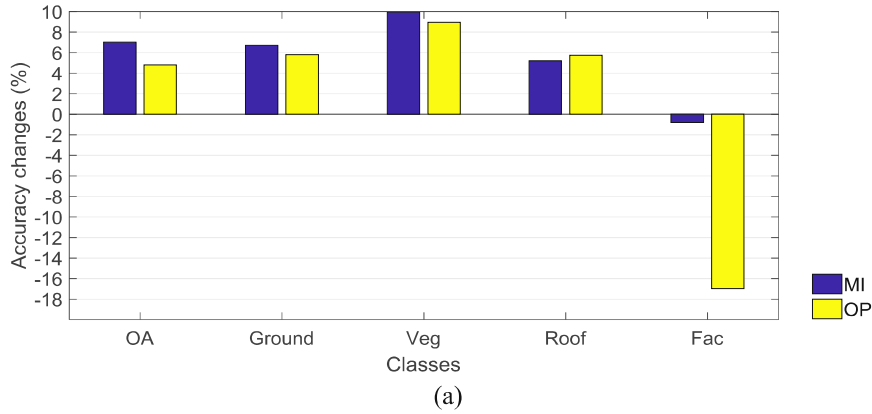
(grass, roads, cars, roofs, façades, and high vegetation).

With respect to the accuracy, our approach only performs better than the other three methods for the road and façade classifications. These three methods are based on CNNs, so high-level features that are achieved by deep convolutional neural networks would be beneficial for further classification. These results for our label-smoothing approach with “only” handcrafted features are satisfactory. Moreover, our main task was to improve the classification accuracy from an initially

poorer classification. As shown in Table 8, significant enhancements were achieved after label smoothing.

5.4. Effect of the initial neighborhood definition on the optimal graph

The optimal neighbors had to be chosen from the rough initial neighborhood. Thus, we designed a series of initial neighborhood-selection methods to test the effect of the initial neighborhood on the



Comparison of different compatibility-coefficient estimations: (a) classification-accuracy changes in the Vienna dataset; (b) classification-accuracy changes in the Vaihingen dataset.

smoothing results. The following label-smoothing techniques in Table 9 only vary in terms of the initial neighborhood definition; the same optimal graph-estimation method was used, and the same input-probability modification from PLR was applied.

The accuracy changes were computed to evaluate the initial-neighborhood effect on the label-smoothing results, which can be found in Fig. 14. The OA change differences among all the initial neighborhood-definition methods were within 1% for the Vienna dataset, except for the initial neighborhoods that were defined by k10 and k20. This result was caused by a lack of sufficient neighboring label information to modify large wrongly labeled regions, given the small initial neighborhood. More specifically, the ground-accuracy improvements of the initial neighborhoods that were constructed by k10 and k20 were much lower than those of the other initial neighborhood-definition methods for the Vienna dataset. For the Vaihingen dataset’s label smoothing, the initial neighborhood that was defined by r3 had the smallest OA increase, losing a large amount of accuracy in terms of grassland, cars and low vegetation. Given such a large initial neighborhood, a large amount of grassland points were changed into roads because of their identical geometric attributes. The intensity feature, which could potentially distinguish between grassland and roads, apparently did not support the discrimination, possibly because of the uncalibrated nature of these measurements.

5.5. Effect of the compatibility coefficients

In addition to MI, another approach to estimate compatibility

coefficients is to compute the occurrence probability (OP) of the label pair (c_i, c_j) . This factor is the probability of one node belonging to class c_i , given that another node in the pair belongs to class c_j . Given the initial classification result, any label c_i that appears in a certain type of neighborhood of class label c_j is counted as n_{c_i, c_j} , and the number of class label c_j is recorded as n_{c_j} . The compatibility coefficient is computed as

$$R(c_i, c_j) = \frac{n_{c_i, c_j}}{n_{c_j}} \tag{15}$$

We used the compatibility coefficients from the OP to build the neighborhood function and compared this function with the results when using compatibility coefficients from MI, which is shown in Fig. 15. The compatibility coefficients from the OP heavily depended on the number of class appearances in the scene, so the neighborhood function could favor one type of class. For the Vienna dataset, the upper compatibility coefficient favored vegetation rather than façades in terms of ground, so more points were classified as vegetation, and the façade class achieved less accuracy. For the Vaihingen dataset, the accuracy of flat roofs and façades also decreased for the same reason. In Section 3.2.3, we assigned equal conditional probabilities to the correlative label pairs to avoid favoring classing that occur in large quantities.

Compatibility coefficients are more object driven than data driven, so urban scenes can be characterized by the composition of fixed types of objects. Compatibility coefficients can also be computed from other sources of urban LiDAR points, such as mobile laser scanning, image matching or 3D models of cities without restriction in terms of the

manner in which the concerned point cloud is generated. Additionally, compatibility coefficients are unrelated to their corresponding sites. Thus, compatibility coefficients that are computed from other point-cloud sites can also be applied to the concerned site as long as the type and number of target classes are consistent in these sites.

In this paper, we presented a contextual label-smoothing technique to improve the classification accuracy of 3D point clouds. The optimal neighborhood-selection method was performed on each point based on the weighted geometric similarity. Thus, label-relevant neighbors for each point were collected. Then, the optimal graph of 3D points could be structured by linking each point and its optimal neighbors. Additionally, the initial label probabilities were improved by PLR (Probabilistic Label Relaxation). The spatial dependencies of objects could be represented by three types of compatibility coefficients (upper compatibility coefficients, middle compatibility coefficients and lower compatibility coefficients), which decide the directional influences on each class. Based on the neighborhood dependencies, the label probability set of each point was iteratively updated to a reliable probability set that was more logical with the spatial contextual knowledge. Finally, given the optimal graph and improved label probability set, smooth labels were achieved by GSR (Graph-Structured Regularization).

Our evaluation showed that the contextual label smoothing performed quite effectively on two selected airborne LiDAR datasets with complex urban scenes. The overall accuracy increased by 7.01% for the Vienna dataset when discerning four general target objects. For a more complex classification task, an OA improvement of 6.88% was achieved for the Vaihingen dataset when detecting eight target objects. Therefore, a classification improvement could be achieved independently of the targeted detail degree that was desired in the classification task. Furthermore, most large wrongly labeled regions were correctly classified after contextual label smoothing in both datasets. As a benefit from PLR, the contextual label smoothing also assigned correct labels to wrongly labeled regions where correct label interactions were missing from the neighborhood.

The only inputs for our contextual label-smoothing approach were the label probability set and 3D coordinates. Any probabilistic classifiers such as RFs (Random Forest) or SVMs (Support Vector Machine) can be used for the initial classification. Instead of adding contextual constraints in the classifiers, we incorporated contextual information in post-processing. With a few given thresholds, the spatial-correlation parameters (such as compatibility coefficients or robust normal vectors) were automatically learned from the xyz coordinates and initial probabilities per class. The LiDAR classification accuracy could be significantly improved through the contextual label-smoothing framework without additional reference data.

The Vienna airborne LiDAR dataset was provided by the 'Stadtvermessung Wien', the Magistrate of the City of Vienna.

The Vaihingen airborne LiDAR dataset was provided by the German Society for Photogrammetry, Remote Sensing and Geoinformation (DGPF) (Cramer, 2010): <http://www.ifp.uni-stuttgart.de/dgpf/DKEF-Allg.html>.

This research was also supported by the National Natural Science Foundation of China (Project No. 41771481).

Supplementary data to this article can be found online at <https://doi.org/10.1016/j.isprsjprs.2018.11.022>.

- Anguelov, D., Taskarf, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., Ng, A., 2005. Discriminative learning of markov random fields for segmentation of 3d scan data. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE, pp. 169–176.
- Boykov, Y., Kolmogorov, V., 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Trans. Pattern Anal. Mach. Intell. 26, 1124–1137.
- Boykov, Y., Veksler, O., Zabih, R., 2001. Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. 23, 1222–1239.
- Cramer, M., 2010. The DGPF-test on digital airborne camera evaluation—overview and test design. In: Photogrammetrie-Fernerkundung-Geoinformation, pp. 73–82.
- Demantke, J., Mallet, C., David, N., Vallet, B., 2011. Dimensionality based scale selection in 3D lidar point clouds. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. 38, W12.
- Filin, S., Pfeifer, N., 2005. Neighborhood systems for airborne laser data. Photogramm. Eng. Remote Sens. 71, 743–755.
- Guo, B., Huang, X., Zhang, F., Sohn, G., 2015. Classification of airborne laser scanning data using JointBoost. ISPRS J. Photogramm. Remote Sens. 100, 71–83.
- Guo, L., Chehata, N., Mallet, C., Boukir, S., 2011. Relevance of airborne lidar and multispectral image data for urban scene classification using Random Forests. ISPRS J. Photogramm. Remote Sens. 66, 56–66.
- He, E., Chen, Q., Wang, H., Liu, X., 2017. A curvature based adaptive neighborhood for individual point cloud classification. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. 42.
- Huang, X., Lu, Q., Zhang, L., Plaza, A., 2014. New postprocessing methods for remote sensing image classification: a systematic study. IEEE Trans. Geosci. Remote Sens. 52, 7140–7159.
- John, A.R., Xiuping, J., 2006. Remote Sensing Digital Image Analysis. Springer-Verlag, Berlin Heidelberg, New York, pp. 55.
- Jung, J., Chen, L., Sohn, G., Luo, C., Won, J.-U., 2016. Multi-range conditional random field for classifying railway electrification system objects using mobile laser scanning data. Remote Sens. 8, 1008.
- Kang, X., Li, S., Benediktsson, J.A., 2014. Spectral-spatial hyperspectral image classification with edge-preserving filtering. IEEE Trans. Geosci. Remote Sens. 52, 2666–2677.
- Khoshelham, K., Elberink, S.O., Xu, S., 2013. Segment-based classification of damaged building roofs in aerial laser scanning data. IEEE Geosci. Remote Sens. Lett. 10, 1258–1262.
- Kumar, B., Dikshit, O., 2017. Spectral contextual classification of hyperspectral imagery with probabilistic relaxation labeling. IEEE Trans. Cybern. 47, 4380–4391.
- Kumar, S., Hebert, M., 2006. Discriminative random fields. Int. J. Comput. Vision 68, 179–201.
- Laible, S., Khan, Y.N., Zell, A., 2013. Terrain classification with conditional random fields on fused 3d lidar and camera data. In: 2013 European Conference on Mobile Robots (ECMR). IEEE, pp. 172–177.
- Lalonde, J.-F., Unnikrishnan, R., Vandapel, N., Hebert, M., 2005. Scale selection for classification of point-sampled 3D surfaces. In: Fifth International Conference on 3-D Digital Imaging and Modeling, 2005. 3DIM 2005. IEEE, pp. 285–292.
- Landrieu, L., Raguet, H., Vallet, B., Mallet, C., Weinmann, M., 2017. A structured regularization framework for spatially smoothing semantic labelings of 3D point clouds. ISPRS J. Photogramm. Remote Sens. 132, 102–118.
- Lee, I., Schenk, T., 2002. Perceptual organization of 3D surface points. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. 34, 193–198.
- Li, N., Pfeifer, N., Liu, C., 2017. Tensor-based sparse representation classification for Urban Airborne LiDAR points. Remote Sens. 9, 1216.
- Li, W., Hu, W., Ran, Q., Zhang, F., Du, Q., Younan, N., 2014. Improving hyperspectral image classification using smoothing filter via sparse gradient minimization. In: 2014 8th IAPR Workshop on Pattern Recognition in Remote Sensing, pp. 1–4.
- Li, Z., Zhang, L., Tong, X., Du, B., Wang, Y., Zhang, L., Zhang, Z., Liu, H., Mei, J., Xing, X., 2016. A three-step approach for TLS point cloud classification. IEEE Trans. Geosci. Remote Sens. 54, 5412–5424.
- Lim, E.H., Suter, D., 2009. 3D terrestrial LIDAR classifications with super-voxels and multi-scale conditional random fields. Comput. Aided Des. 41, 701–710.
- Linsen, L., Prautsch, H., 2001. Local versus global triangulations. In: Proceedings of EUROGRAPHICS, pp. 257–263.
- Lodha, S.K., Fitzpatrick, D.M., Helmbold, D.P., 2007. Aerial lidar data classification using adaboost. In: The Sixth International Conference on 3-D Digital Imaging and Modeling. IEEE, Montreal, QC, Canada, pp. 435–442.
- Lu, Q., Huang, X., Liu, T., Zhang, L., 2016. A structural similarity-based label-smoothing algorithm for the post-processing of land-cover classification. Remote Sens. Lett. 7, 437–445.
- Lu, W.-L., Murphy, K.P., Little, J.J., Sheffer, A., Fu, H., 2009. A hybrid conditional random field for estimating the underlying ground surface from airborne lidar data. IEEE Trans. Geosci. Remote Sens. 47, 2913–2922.
- Luo, C., Sohn, G., 2014. Scene-layout compatible conditional random field for classifying terrestrial laser point clouds. ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci. 2, 79.
- Mallet, C., Bretar, F., Roux, M., Soergel, U., Heipke, C., 2011. Relevance assessment of full-waveform lidar data for urban area classification. ISPRS J. Photogramm. Remote Sens. 66, S71–S84.
- Munoz, D., Vandapel, N., Hebert, M., 2008. Directional associative markov network for 3-d point cloud classification.
- Najafi, M., Namin, S.T., Salzmann, M., Petersson, L., 2014. Non-associative higher-order

- markov networks for point cloud classification. In: European Conference on Computer Vision. Springer, pp. 500–515.
- Niemeyer, J., Mallet, C., Rottensteiner, F., Sörgel, U., 2011. Conditional random fields for the classification of LiDAR point clouds. In: International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences: [ISPRS Hannover Workshop 2011: High-Resolution Earth Imaging For Geospatial Information]. Copernicus GmbH, Göttingen, pp. 209–214.
- Niemeyer, J., Rottensteiner, F., Soergel, U., 2012. Conditional random fields for lidar point cloud classification in complex urban areas. ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci. 3, 263–268.
- Niemeyer, J., Rottensteiner, F., Soergel, U., 2014. Contextual classification of lidar data and building object detection in urban areas. ISPRS J. Photogramm. Remote Sens. 87, 152–165.
- Niemeyer, J., Rottensteiner, F., Sörgel, U., Heipke, C., 2016. Hierarchical higher order crf for the classification of airborne lidar point clouds in urban areas. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.-ISPRS Arch. 41, 655–662.
- Nurunnabi, A., West, G., Belton, D., 2015. Outlier detection and robust normal-curvature estimation in mobile laser scanning 3D point cloud data. Pattern Recognition 48, pp. 1404–1419.
- Rottensteiner, F., Sohn, G., Jung, J., Gerke, M., Baillard, C., Benitez, S., Breitzkopf, U., 2012. The ISPRS benchmark on urban object classification and 3D building reconstruction. ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. 1, 293–298.
- Rusu, R.B., Holzbach, A., Blodow, N., Beetz, M., 2009. Fast geometric point labeling using conditional random fields. In: IROS, pp. 7–12.
- Schindler, K., 2012. An overview and comparison of smooth labeling methods for land-cover classification. IEEE Trans. Geosci. Remote Sens. 50, 4534–4545.
- Secord, J., Zakhor, A., 2007. Tree detection in urban regions using aerial lidar and image data. IEEE Geosci. Remote Sens. Lett 4, 196–200.
- Shapovalov, R., Velizhev, E., Barinova, O., 2010. Nonassociative markov networks for 3d point cloud classification. Int. Arch. Photogramm. Remote Sensing Spatial Inf. Sci. XXXVIII (Part 3A) Citeseer.
- Tran, G., Nguyen, D., Milenkovic, M., Pfeifer, N., 2015. Potential of full waveform airborne laser scanning data for urban area classification—transfer of classification approaches between missions. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. 40, 441–448.
- Tran, T.H.G., Ressel, C., Pfeifer, N., 2018. Integrated change detection and classification in urban areas based on airborne laser scanning point clouds. Sensors 18, 448.
- Vosselman, G., 2013. Point cloud segmentation for urban scene classification. ISPRS Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 1, 257–262.
- Vosselman, G., Coenen, M., Rottensteiner, F., 2017. Contextual segment-based classification of airborne laser scanner data. ISPRS J. Photogramm. Remote Sens. 128, 354–371.
- Wang, D., Brunner, J., Ma, Z., Lu, H., Hollaus, M., Pang, Y., Pfeifer, N., 2018. Separating tree photosynthetic and non-photosynthetic components from point cloud data using dynamic segment merging. Forests 9, 252.
- Weinmann, M., Jutzi, B., Hinz, S., Mallet, C., 2015. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. ISPRS J. Photogramm. Remote Sens. 105, 286–304.
- Xu, S., Vosselman, G., Elberink, S.O., 2014. Multiple-entity based classification of airborne laser scanning data in urban areas. ISPRS J. Photogramm. Remote Sens. 88, 1–15.
- Yan, W.Y., Shaker, A., El-Ashmawy, N., 2015. Urban land cover classification using airborne LiDAR data: a review. Remote Sens. Environ. 158, 295–310.
- Yang, Z., Jiang, W., Xu, B., Zhu, Q., Jiang, S., Huang, W., 2017. A convolutional neural network-based 3D semantic labeling method for ALS point clouds. Remote Sens. 9, 936.
- Yastikli, N., Cetin, Z., 2016. Classification of LiDAR data with point based classification methods. Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 441–445.
- Zhao, R., Pang, M., Wang, J., 2018. Classifying airborne LiDAR point clouds via deep features learned by a multi-scale convolutional neural network. Int. J. Geographical Inf. Sci. 32, 960–979.



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.



Publication IV

- Title:** A Comparison of Deep Learning Methods for Airborne Lidar Point Clouds Classification
- Authors:** Li, N. and Pfeifer, N
- Published in:** IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 14, 6467-6468
DOI: 10.1109/JSTARS.2021.3091389



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

A Comparison of Deep Learning Methods for Airborne Lidar Point Clouds Classification

Nan Li , Olaf Kähler, and Norbert Pfeifer 

Abstract—The success achieved by deep learning techniques in image labeling has triggered a growing interest in applying deep learning for three-dimensional point cloud classification. To provide better insights into different deep learning architectures and their applications to ALS point cloud classification, this article presents a comprehensive comparison among three state-of-the-art deep learning networks: PointNet++, SparseCNN, and KPConv, on two different ALS datasets. The performances of these three deep learning networks are compared w.r.t. classification accuracy, computation time, generalization ability as well as the sensitivity to the choices of hyper-parameters. Overall, we observed that PointNet++, SparseCNN, and KPConv all outperform Random Forest on the classification results. Moreover, SparseCNN leads to a slightly better classification result compared to PointNet++ and KPConv, while requiring less computation time and memory. At the same time, it shows a better ability to generalize and is less impacted by the different choices of hyper-parameters.

Index Terms—ALS point clouds, classification, comparison, deep learning.

I. INTRODUCTION

ALS is one of the most important techniques for data collection of real-world scenes. Extraction of meaningful information from ALS point clouds is fundamental for many applications, which makes the automatic classification of three-dimensional (3-D) point clouds a crucial task.

For the past years, tremendous progress in the automatic classification of ALS point clouds has been achieved in the community of remote sensing and photogrammetry. The general methods compute a set of handcrafted features for each point and conduct the automatic classification by means of machine learning, such as decision tree (DT) [1], support vector machines (SVM) [2], [3], and random forest (RF) [4], [5]. To achieve smoother classification results, many studies employ graph-based models to introduce contextual information [5] or execute the classification on segments to ensure a local consistency of predicted labels [6]. Nevertheless, the performance of these methods still heavily relies on the representation ability of the handcrafted features and the generalization abilities of machine learning models are limited due to the shallow architectures.

Manuscript received February 12, 2021; revised May 18, 2021; accepted June 8, 2021. Date of publication June 23, 2021; date of current version July 14, 2021.
(Corresponding author: Nan Li.)

Nan Li and Norbert Pfeifer are with the Department of Geodesy and Geoinformation, Technische Universität Wien, 1040 Vienna, Austria (e-mail: nan.li@geo.tuwien.ac.at; norbert.pfeifer@geo.tuwien.ac.at).

Olaf Kähler is with the Research Group Active Vision Technologies, Siemens AG Österreich, 1210 Vienna, Austria (e-mail: olaf.kaehler@siemens.com).

Digital Object Identifier 10.1109/JSTARS.2021.3091389

In recent years, deep learning (DL) has shown superior performance to other machine learning models across a wide range of areas, such as speech recognition [7], image semantic segmentation [8], and natural language processing [9]. Unlike other machine learning approaches, deep learning can directly learn features during the training process, and this ability of learning features is considered as the major reason for those outstanding results. More recently, researchers have also developed various deep learning architectures for 3-D point cloud classification, such as PointNet++ [10] and SparseCNN [11]. Considering the impressive results achieved on indoor point clouds, those deep learning frameworks are further applied on the outdoor point clouds in the field of remote sensing and photogrammetry. Compared to indoor point clouds, outdoor point clouds like ALS normally contain much larger scale scenes with large size objects, which leads to a larger data volume. Some studies [12], [13] utilized DL models derived from the computer vision field for the ALS classification by using adapted parameters. Other studies [14], [15] proposed new architectures or built modified models upon the PointNet architecture, in order to embed characteristics of ALS point clouds.

However, a firm comparison of DL for the classification of ALS remains difficult. Most reported results were obtained with different experimental setups, such as the generation of training patches in preprocessing, class balancing strategies, and learning rate schedule [16]. Moreover, the models are mostly trained and evaluated on the ISPRS Vaihingen 3-D dataset [14], [15], [17], [18]. Although these benchmark point clouds provide the possibility for performance comparison of different models, it only covers a small area with limited diversity in the scenes [12]. Thus, our aim is to present a comprehensive comparison among three state-of-the-art DL models, namely PointNet++, SparseCNN, and KPConv. Unlike normal comparisons of approaches that are conducted on only one benchmark, we performed a more comprehensive comparison on two large-scale ALS datasets with very different application purposes, and therefore different semantics. One dataset is the ALS point cloud of Vienna city and the tiles we use for training and testing in total cover an area of approximately 9 km² with different urban scenes. Note that the tiles used in this article are published on ZENODO and can serve further as benchmark data for ALS point cloud classification [19]. Another dataset consists of the ALS point clouds of power line facilities, which are used for the power line inspection and management.

Through the inspection of the classification results of two different datasets and a group of further investigations, this

studies reveal the strengths and weaknesses of the selected networks in the classification of ALS point clouds. To provide insights into choosing networks and tuning hyper-parameters for the classification of ALS point clouds, we conducted the comparison in the following aspects:

- 1) The accuracy of the classification results achieved on two different datasets. Here, RF is used as a baseline machine learning model so that the effectiveness of three selected architectures can be validated by the classification performance improvement.
- 2) The computational cost spent on two datasets. This includes the GPU memory demands and the computation time used for preprocessing, training, and evaluation.
- 3) The generalization ability is assessed on the point clouds of power line facilities, as this dataset contains point clouds collected from four different campaigns located in different regions and with different acquisition parameters.
- 4) The sensitivity to the choices of hyper-parameters. Based on the impacts of different selections of hyper-parameters on the classification performances, we attempted to find the crucial parameters for PointNet++, SparseCNN, and KPConv. Concerning the long training process and the time budget for performing research, experiments are only conducted on a subset of the dataset of the power lines facilities.

The rest of this article is organized as follows: In Section II, we review the classical methods for ALS point cloud classification and deep learning models for the classification of 3-D points. In Section III, we present the principals and implementations of the three selected models, together with the descriptions of the two datasets and the evaluation metrics. The experiments are analyzed in Section IV. Finally, the conclusions are given in Section V.

II. RELATED WORK

Relevant works of ALS point cloud classification can be generally divided into two main groups: classification based on handcrafted features followed by classical machine learning and classification based on deep learning with learned features. We will review both groups in the following with a special focus on deep learning.

A. Classification Based on Classical Machine Learning

The classical approach for the classification of 3-D point clouds starts with the extraction of handcrafted features and then utilizes machine learning models to label input points according to those features. The handcrafted features normally refer to the geometric features that are generated from a 2-D or 3-D neighborhood, such as eigenvalue-based features [20], height features [21], and echo features [22]. A comprehensive description of handcrafted features derived from 2-D and 3-D neighborhoods can be found in the work of Weinmann *et al.* [23]. As for the classifiers, various classical machine learning models are available for ALS point clouds, such as DT [1], Bayesian discriminant classifiers [24], SVM [2], [3], Adaboost [25], [26], and RF [4], [5]. However, those aforementioned methods suffer

from inhomogeneous classification results caused by the unstable features at the boundaries of objects. To achieve smoother and more accurate results, some studies make use of the contextual information by employing graph-based models, such as Markov random fields (MRF) [27] and the more frequently used conditional random fields (CRF) [5], [28], [29]. The main difference is that CRFs take the local context into account, when selecting the co-occurrence probabilities of different objects, whereas MRFs assume a fixed co-occurrence matrix irrespective of the input context. To incorporate further context from a larger region, several studies proposed to use a multiscale neighborhood. Luo and Sohn [30] built the adjacency graph by using two different range parameters. In the work of Jung *et al.* [31] three types of the directional neighborhood were included. Since larger segments of point clouds are considered more capable of providing representative features for classification tasks than single points, some studies apply CRF models on such segments. Niemeyer *et al.* [32] proposed a two-layer hierarchical CRF model that consists of a point-based layer and a segment-based layer. Vosselman *et al.* [6] first conducted a segmentation on point clouds, then the interaction features of segments are used in the CRF model. Regardless of the classification and label smoothing methods, well-designed handcrafted features need to be fed into these models, and the overall accuracy (OA) still mostly depends on the effectiveness of handcrafted features. An additional challenge for graph-based models and methods based on segments are small and thin objects, which often tend to be misclassified.

B. Classification Based on Deep Learning

In contrast to the above, numerous deep learning architectures have been developed for the classification of 3-D points in the field of computer vision. Those architectures can be categorized into three groups: deep learning based on regular representations, point-wise deep learning by multilayer perceptron (MLP) and point-wise deep learning by convolution.

1) *Deep Learning Based on Regular Representations*: These methods usually use 3-D grids or seek a compact structure to represent unordered point clouds, so that the classical convolutional neural network (CNN) can be applied on the regular representations of point clouds. VoxNet [33] applied a 3-D voxel-based convolution on the voxel grids of point clouds. Although encouraging results were achieved, this method is unlikely to scale to large point clouds as the computation and memory demand grow cubically with the resolution. To address this inefficiency, SparseCNN [11] took only nonempty voxels as input, and also restricted the output of the convolution to the set of nonempty voxels. In this way, the submanifold dilation can be avoided. Apart from voxels, various sparse data structures are also employed to apply CNNs to 3-D point clouds, such as Octrees used in OctNet [34] and Kd-trees used in KD-Net [35]. Regular representations can also be achieved by projections or interpolations of input points. Tangent convolution [36] projected local neighborhoods of points onto a tangent plane for each point, resulting in a set of tangent images. Then planar convolutions are performed on each tangent image to compute

the features of every point. SPLATNet [37] interpolated input points onto a sparse lattice by means of bilateral convolution layers (BCLs) [38], [39]. The convolution of BCLs can then be applied on this sparsely populated lattice and at last learned features are interpolated back to the original input 3-D points.

In the last few years, hybrid models, where a mixture of voxel-based, projection-based, and point-wise operations are used to learn features, have been investigated in order to take advantage of different representations of point clouds. For example, PVCNN [40] and SPVConv [41] use a point-voxel fusion scheme, in which voxels generate large-scale features, while point-wise MLPs are responsible for preserving fine geometrical features. FusionNet [42] also works on the point-voxel interaction. It proposes a voxel-based mini-PointNet that directly aggregates features between neighboring voxels and corresponding points, in this way the complexity in neighborhood search can be reduced. The hybrid feature fusion module presented in (AF)²-S3Net [43] focuses on point-based, medium-scale voxel-based, and large-scale voxel-based features, and it further designs a unique adaptive feature selection module with feature map re-weighting in the decoder. Besides point-wise networks and voxel-wise networks, RPVNet [44] adds a range-based image network and builds the framework to learn mutual information interactions among these three branches. AMVNet [45] considered the class-prediction uncertainty between different view's networks, and these uncertain points are passed into an extra network to obtain more robust predictions. 3-D-MiniNet [46] learns a 2-D representation from the raw points through a projection that extracts local and global information from the 3-D data, and then feeds it to a 2-D fully CNN to generate semantic predictions.

2) *Point-Wise Deep Learning Based on MLP*: A second range of methods employs MLPs on groups of local points to compute features and then aggregates these features using a symmetric operation to achieve permutation invariance for 3-D point clouds. This idea was pioneered in PointNet [47], but the original PointNet lacks a consideration of local features. Therefore, the authors further proposed PointNet++ [10] that takes both local and global features into account. Several modifications or variations based on PointNet or PointNet++ were developed, such as PointSIFT and PointWeb. Inspired by the Scale Invariance Feature Transform (SIFT) feature descriptor, PointSIFT [48] added a series of orientation-encoding units in the architecture of the PointNet++, in order to combine features from different orientations. Since the simple max or average pooling operations used in PointNet++ do not make the best use of local contextual information, PointWeb [49] formulates an adaptive feature adjustment module to learn the impact value of each on other points in the given local region for a better feature aggregation. SO-Net [50] built a self-organizing map (SOM) to model the spatial distribution of point clouds, its separate-and-assemble process is more efficient than the grouping strategy used in PointNet++ because the SOM explicitly reveals the spatial distribution of the input point cloud. PointASNL [51] innovatively designed a local-nonlocal module, in which the key component is the Point Nonlocal Cell. It allows the computation of the response of a sampled point as a weighted sum of the

influences of the entire point clouds, instead of just within a limited neighbor range.

MLP-based techniques normally treat points independently at a local scale to maintain permutation invariance, and this lacks the consideration of dependencies between semantic context and geometric relationships between points. Some studies work on learning feature relationships in 3-D point clouds, since the high-level features learned by deep learning can be considered as category responses, which should have dependencies regarding to context. SPG [52] first geometrically partitioned the point cloud into homogenous superpoints, upon which it built a graph convolution network to learn and pass the contextual information between adjacent superpoints. Liu *et al.* [53] proposed a Point Context Encoding (PointCE) module to learn a set of scaling factors, so that the network can selectively focus on a few important intermediate features. Similarly, LAE-Conv [54] learned the coefficients that represent the importance of neighbors to the central point by MLP, then aggregates the central point features as a weighted sum of its neighbors. PointGCR [55] module aims to capture global contextual information along the channel dimension by utilizing a graph structure. It automatically generated a neighborhood matrix of the graph convolution operation, and finally passes the information to capture the relationship between these channel feature maps. DGCNN [56] is proposed to incorporate edge convolution layers (EdeConv) into the PointNet architecture. EdgeConv explicitly constructed a local graph and learns the relationships between a point and its neighbors.

Besides the aforementioned research that explores local regions to leverage correlations between unordered points, there are also some studies that focus on exploiting the geometrical correlations of the local neighborhood points. RandLA-Net [57] uses a local spatial encoding unit that explicitly embeds the x - y - z coordinates of all neighboring points, such that the corresponding point features are always aware of their relative spatial locations. RS-CNN [58] aims to learn the relationships among neighboring points, i.e., the geometric topology constraint among points, by stacking multiple MLPs.

3) *Point-Wise Deep Learning Based on Convolution*: As an alternative approach, suitable convolutional operations can be defined, such that they operate directly on unordered point clouds. To achieve permutation invariance, A-CNN [59] first ordered input points locally in a clockwise/counterclockwise manner before feature encoding, then applied a 1-D annular convolution. PointCNN [60] designed a transformation matrix for the local input points, which is able to simultaneously weight and permute the input features. In the works of PointConv [61], SpiderCNN [62], and Flex-convolution [63], the convolution kernels are treated as parametric functions of 3-D coordinates. PointConv [61] approximated the parametric functions by MLP, and an inverse density scale is added to each point in order to balance nonuniform point density. SpiderCNN [62] parametrized the convolution operator as a product of a simple step function that captures local geodesic information and a Taylor polynomial that ensures the expressiveness. Flex-convolution [63] used a linear function to approximate the convolution operator in a 3-D continuous space. To perform the convolution in a 3-D space,

some studies employed a set of 3-D spatial points to define the area where each kernel weight is applied, such as PCNN [64], ConvPoint [65], and KPConv [66]. PCNN [64] defined a 3-D convolution operator whose weights are carried by a group of fixed grid points and the weights of grid kernel points are computed by the radial basis function. ConvPoint [65] defined the spatial convolutional kernel as a set of 3-D points that are randomly distributed in a unit sphere, and the impact of kernel points on raw 3-D points are learned by an MLP. KPConv [66] proposed a deformable kernel, i.e., weights and the positions of the kernel points are both learnable so the kernel can be adapted to different shapes and point densities. FKACConv [67] explicitly separates the estimation of geometry-less kernel weights and their alignment to the spatial support of features, and its key idea is to learn a linear transformation to align the neighboring points with the grid-like kernel and then perform kernel operation.

4) *Deep Learning for the Classification of ALS Point Clouds:* Since the aforementioned architectures are derived from the computer vision field, the training and evaluation are typically performed on publicly available indoor point clouds, such as ShapeNet [68] and ScanNet [69]. For the classification of outdoor point clouds, as typically encountered in the field of remote sensing and photogrammetry, some popular deep learning models were successfully adapted to ALS point clouds and achieved satisfactory results, like PointNet++ used by Winiwarter *et al.* [13] and SparseCNN used by Schmohl *et al.* [12]. Yousefhusien [18] utilized the same scheme of combining local and global information as PointNet, but replaced the MLP-based architecture with a 1-D fully convolutional architecture. Li [14] proposed a geometry-aware convolution, which aims to learn the high-level features from the low-level handcrafted features, so that the geometric awareness can be emphasized by the prior knowledge of the neighborhood. To avoid overfitting, Arief [17] refined a trained PointCNN model by combing the strict pairwise penalties that are used in the CRF procedure on the unseen data. To address the challenges of uneven density distribution, Li [70] introduced a density-aware convolutional module which adds an inverse density function to reweight the convolutional kernel. Wen [15] constructed the local receptive field by selecting the directionally constrained nearest neighbors, then the orientation-aware features can be aggregated by following the order of each angular segment when formulating the receptive field. In the study of [71], handcrafted features were integrated into a simple neural network model. Contrary to real deep learning, this approach lacks the ability to learn local features automatically, and the network is not able to learn more effective features through optimization in the training step.

In this study, we selected one representative architecture from each category mentioned above, namely PointNet++, SparseCNN, and KPConv, to present a comprehensive comparison. Besides the successful performance of these three networks achieved in indoor point clouds, they also showed powerful capability in the ALS point cloud classification in the previous works [12], [13], [71]. PointNet++ is a point-wise network in which features are learned by successive levels of MLPs followed by order-invariant pooling operations. In contrast, SparseCNN uses voxels as internal representation and applies

convolutional operations directly on this regular grid. KPConv designs a 3-D filter by using 3-D points as kernel points to carry weights, and the convolution is performed on 3-D point without any voxelization.

In comparison to MLP based approaches, both point-wise and voxel-wise convolutional methods are theoretically considered advantageous, because they can exploit the spatial arrangement of points inherently, whereas MLP takes flattened vectors as input, which leads to a loss of spatial information. Voxels provide a natural way of defining convolution masks. They also offer much faster lookup of neighboring points compared to point-wise methods, and the required matrix multiplications are well suited for implementation on GPUs. However, voxel-wise methods intuitively suffer from defects in accuracy by predicting labels only on quantized grids. Instead of voxelization, the convolutional operators can also be represented as a group of 3-D spatial points with learnable weights in a sphere around the given point. However, this 3-D kernelization might be memory-demanding, since all the neighboring points within the sphere are involved in the convolution. In summary, the CNNs employed in SparseCNN and KPConv are expected to be superior to MLP used in PointNet++, while the resolution loss by SparseCNN may decrease classification accuracy and KPConv may have a difficulty in large-scale dense point clouds. Thus, we conducted a number of experiments to examine the capabilities of the different classification approaches for ALS point clouds.

III. METHODS AND MATERIALS

In this section, we introduce the two ALS datasets used for training and evaluation in Section III-A. The subsequent Section III-B briefly reviews the architectures of the PointNet++, SparseCNN, and KPConv, followed by the training setups for the investigated models in Section III-C. Finally, we present our evaluation metrics in Section III-D.

A. Datasets

We evaluate the selected deep learning models on two ALS LiDAR datasets. The first one is an ALS LiDAR point cloud of Vienna. The LiDAR system recorded up to 10 echos per emitted pulse, and the point density within 2-D and 3-D search spaces is presented in Fig. 1. The point density of first returns is mostly between 20 and 50 points/m², and the number of points within a 3-D sphere of radius 0.5 m is approximately between 10 and 25.

The dataset has been organized into a number of 1270 m × 1020 m tiles (including an overlap of 20 m), from which we selected 4 tiles for the training and 5 tiles for the evaluation. The locations of the chosen tiles are presented in blue and red, respectively, in Fig. 2. Training tiles are situated in four different districts of Vienna, which contain various land cover and terrain such that sufficient representative samples can be provided to the classifiers.

For the evaluation, test tiles with different scenes are considered. Test tile 1 is characterized by a large wooded area with significant slopes, along with several spacious detached houses and a small part of the Danube River. Test tiles 2 and 3 are located

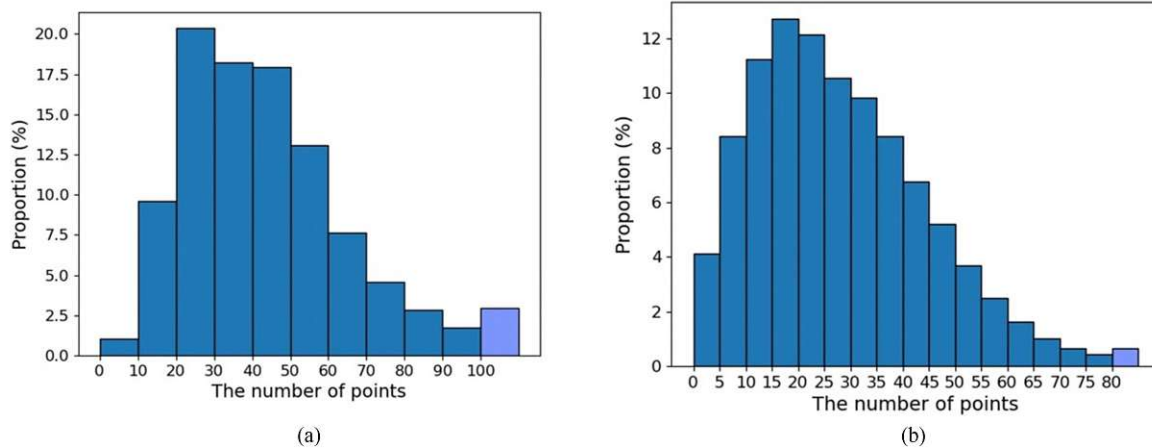


Fig. 1. Histograms of point density for the Vienna point cloud. (a) Histogram of point counts of first returns in 2-D cells of 1 m². Cells with point counts more than 100 pts/m² are labeled light blue. (b) Histogram of point counts within 3-D sphere with radius of 0.5 m. Point counts higher than 80 are labeled light blue.

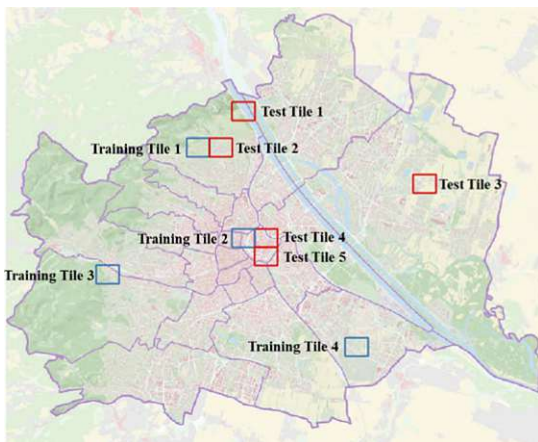


Fig. 2. Locations of training and test tiles.

in suburban areas with mixed landscape. Test tile 2 has a large number of residential houses surrounded by dense trees, areas of vineyards and a small cemetery. Test tile 3 covers a large area of agricultural land, along with buildings of different heights. Test tiles 4 and 5 are located in the center of Vienna, which are dominated by various historic and tall office buildings.

For the purpose of Vienna city administration, five classes are considered, namely *ground*, *buildings*, *vegetation*, *water*, and *others*. All common street objects are categorized as *others*, such as streetlights, benches, shrubs, cars, construction sites, and garbage bins. The reference labels are generated semi-automatically: a rough filtering of main objects is first conducted by the software “Terrasolid,” and the final classification is refined by manual labeling. After the label generation, the quality of the labeling is manually checked. 20 sites of 100 m × 100 m tiles are manually verified, and the average accuracy of reference labels is 95%. In this article, only four main classes are used: *ground*, *buildings*, *vegetation*, and *others*.

The difficulty of this dataset is the variety of objects within the classes. *Vegetation* includes various types of plants such as trees

and shrubs that usually appear around houses, but also grass. The highest ambiguity is found in the class *others*.

The second dataset is provided by Siemens and consists of ALS point clouds of power line facilities. The point clouds cover four different power line corridors located in different regions, named Area_1, Area_2, Area_3, and Area_4 respectively, and they were collected in different measurement campaigns. Again, we assess the point density by counting the number of points within 3-D spheres of radius 0.5 m. Places with more than 300 points in such a sphere are excluded from the density statistics in order to avoid disturbing artifacts caused by flight geometry deviations, such as variability in aircraft pitch. As shown in Fig. 3, the densities of the four areas vary from each other, especially the point density in Area_3 is much lower than in the other three areas.

Unlike the labeling schema used in the field of land use, the targets of the power line point clouds are specifically restricted to four classes, namely (surrounding) *environment*, *conductors* (i.e., the wires), *pylons*, and *insulators* (connecting wires and pylons). This corresponds to the needs for the inspection and management of the power lines facilities. The surrounding environment refers to all points that are in a certain range of the pylons. Other objects such as ground, building, or vegetation that are far away from pylons, are not considered and eliminated before the classification task. An example of such labeled point clouds of power line facilities is shown in Fig. 4. The reference labels for this dataset are manually generated. The challenges of this dataset are the specific semantic class definitions, the variations in terms of point density, shapes of pylons among the four different areas, and the unbalanced class distribution.

The dataset is split into tiles of 250 m × 250 m and there are 55, 37, 13, and 31 tiles for Area_1, Area_2, Area_3, and Area_4, respectively. We selected 6 tiles each from Area_1, Area_2, and Area_4 and 4 tiles from Area_3 for the evaluation, the remaining tiles are used for training. Thus, there is a total 114 tiles from 4 areas for training, and 22 tiles from 4 areas for the evaluation.

The number of points in the training and test tiles are presented in Table I for the two datasets. Although fewer tiles are used in

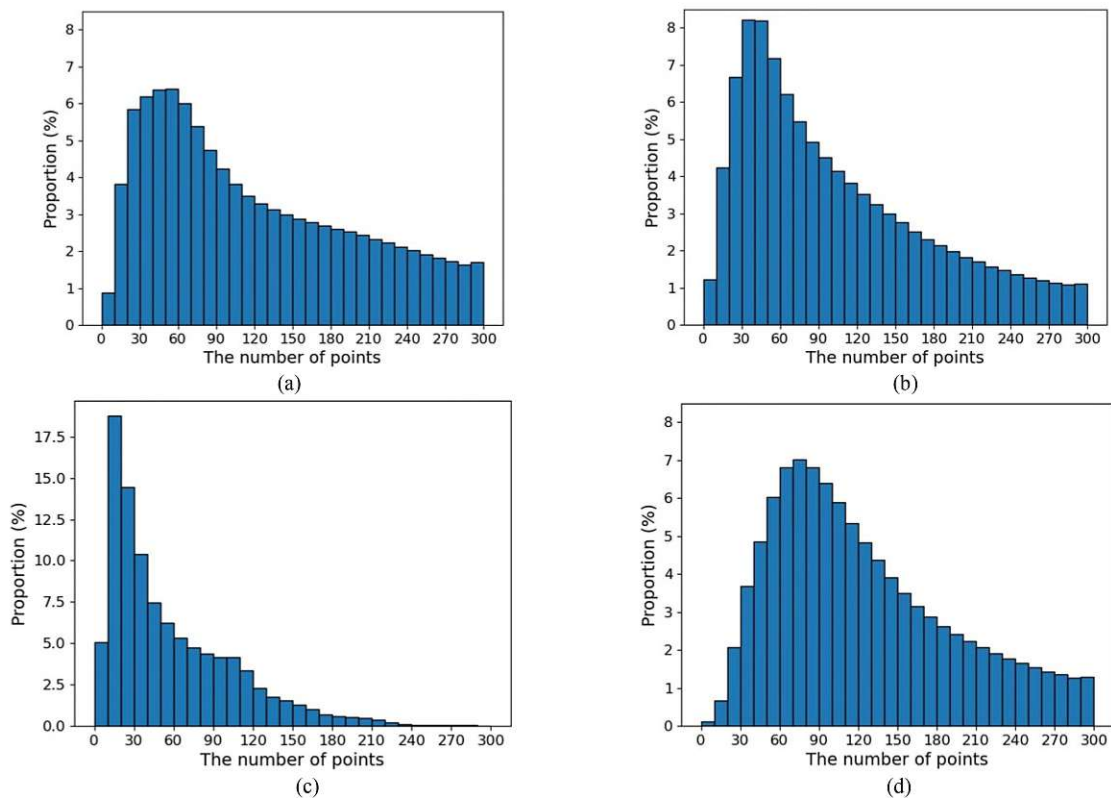


Fig. 3. Histograms over the number of neighboring points within 3-D spheres of radius 0.5 m for the various areas of the power line data set. The percentage of points with more than 300 neighbors is also noted. Note the different vertical range used in (c). (a) Area_1, 30.3% points excluded. (b) Area_2, 19.8% points excluded. (c) Area_3, 0.3% points excluded. (d) Area_4, 21.2% points excluded.

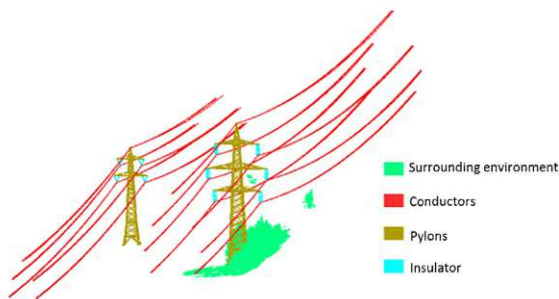


Fig. 4. Labeled point cloud of power line facilities.

TABLE I
NUMBER OF POINTS IN THE TRAINING AND TEST TILES OF THE TWO DATASETS

Dataset	Training	Test
Vienna dataset	447,208,684	321,182,616
Power line facilities dataset	97,162,457	19,875,256

the Vienna dataset than in the power line facilities dataset for training and test, much more points are contained in the training and test tiles of the Vienna dataset compared to the power line facilities dataset. This is because each tile in the Vienna dataset covers a much large area than the tiles in the power line facilities dataset. Consequently, we only selected 9 tiles for the training

and test on the Vienna dataset, as this amount of point clouds is sufficient to perform deep learning experiments and costs much less time than using all tiles of the entire city. In terms of the point numbers and covered areas, the Vienna dataset is much larger than the ISPRS Vaihingen dataset [72] in which there are 750 000 points for training and 400 000 points for test.

B. Description of PointNet++, SparseCNN, and KPConv

PointNet++, SparseCNN, and KPConv all employ the classic U-Net structure [73] for semantic classification. U-Net consists of two parts, the encoding part and a symmetric decoding part. The encoding part progressively downsamples the input data so that large context features can be learned by increasing the receptive field. And then, in the decoding stage, these high-level features are propagated back to the original resolution of the input data by upsampling.

In PointNet++, features are computed in a multiresolution hierarchy by using set abstraction. At each level, the input points are first subsampled to a set of centroid points (illustrated in red in Fig. 5) using farthest point sampling (FPS). Then, the local neighborhood region of each centroid point (dotted circles in Fig. 5) is constructed by selecting its neighboring points in the previous abstraction level. Subsequently, MLPs with max or average pooling is used on these groups of points to extract new high-level features. Each set abstraction contains fewer

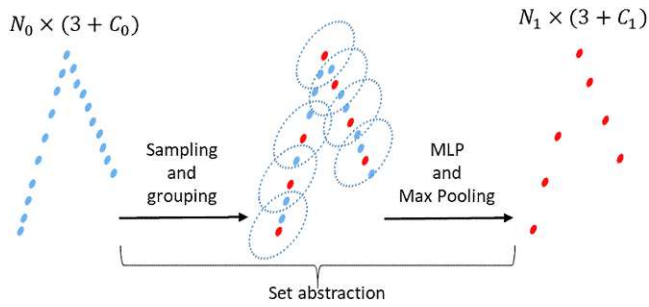


Fig. 5. Schematic illustration of the feature learning in PointNet++. N_0 and N_1 are the number of points in the respective input and output of one hierarchy level and C_0 and C_1 are the respective dimensions of feature vectors.

points but has more informative features attached to each point. Through this subsampling and the increasing neighborhood, the features in deeper levels can progressively cover and describe larger areas of the original point cloud [10].

Unlike the point-wise architecture used in PointNet++, SparseCNN [11] is a voxel-based network that rasterizes irregular point clouds into voxel grids for convolution. In order to avoid unnecessary computations, SparseCNN performs the convolutional operations only on active voxels but also restricts the output of the convolution to a set of active input voxels. Here active voxels are defined as nonempty voxels where at least one point is contained. Internally, SparseCNN converts the convolutional operations to matrix multiplications by building hash tables for input and hidden layers. To give an example, the implementation on a slice of voxels is described in Fig. 6, where active sites are labeled in orange and blue in input and output layers, respectively. The input layer is stored by a hash table and a feature matrix F_{in} . The matrix F_{in} has a size of $a \times m$, where a is the number of active sites in the input layer and m is the number of dimensions of input features. Meanwhile, each row in the hash table records a tuple of spatial coordinates of an active site and its corresponding row index in the matrix F_{in} . Likewise, the output layer also consists of a hash table and a feature matrix F_{out} . Each element in F_{out} is the result of the convolution of the input active sites that are located in the receptive field of the corresponding output active site. Since the features of the input and output active sites are accessed by the index in the matrix F_{in} and F_{out} , an index rule is built as a matrix that records the indices in F_{out} and the corresponding indices of active input sites that are used for convolution in F_{in} . That is to say, when an active input site with a row index of $index_x$ is situated at the spatial spot i in the receptive field of an output active site with a row index of $index_y$, a row $(index_x, index_y)$ will be added to the rule matrix R_i . Thus, for the 3×3 convolution kernel, at each spatial location i an index rule matrix R_i is established, together with a learnable weight matrix W_i of size $m \times n$, where n is the dimension of output features. In this manner, at each spatial location i of the kernel, the convolution is operated as a multiplication of the input feature matrix and the corresponding weight matrix W_i with learnable parameters.

Note that in the architecture of SparseCNN the convolutional block also includes pooling and batch normalization, which are

not depicted in Fig. 2 for the sake of simplicity. These act similar to the subsampling and increasing neighborhood in PointNet++ in the sense of successively increasing the overall receptive field of voxels in the later layers, i.e., aggregating more and more complex information over the larger and larger neighboring context of the input data.

KPConv directly performs 3-D convolution operations on the raw point clouds without any intermediate representations. In KPConv, a set of 3-D points is used as kernel points to carry kernel weights (illustrated in Fig. 7), thus the positions of kernel points define the area where each kernel weight is applied.

KPConv takes a spherical neighborhood centered on a given point as input. For each neighboring point, its kernel value is calculated as a weighted sum of all the kernel weights carried by the kernel points. The influence of each kernel weight is defined by a correlation function based on the distance between the kernel point and the neighboring point. Therefore, the new representation of the given center point is calculated as the sum of all the neighboring points' features multiplied by their correlation to each kernel point and the corresponding kernel weight.

The kernel points can be positioned regularly in a sphere around the given center point, where each kernel point applies a repulsive force on the others. On the other hand, the positions of kernel points can also be learned by the network. In this case, a shift for each kernel point is generated to determine its new location.

For the architecture of KPConv, a grid subsampling is applied in each layer, and the convolution radius is doubled in every layer to incrementally increase the receptive field.

C. Setup of Selected Models

For the implementation of the selected deep learning networks, the point clouds are first cut into small patches with identical amount of points. For PointNet++ and SparseCNN, each patch contains 80 k points and 600 k points for the power lines dataset and the Vienna dataset, respectively. However, for KPConv, fewer points (50 k points) are used to generate the training patches in order to avoid out of memory (OOM) problems. Details on the implementation can be found in [13]. For class balancing, only patches that contain all concerned classes are involved in the training procedure and a weighted loss function mentioned in the study of [12] is utilized. In the case of the Vienna dataset classification by KPConv, the same patch filtering would lead to a very limited number of training patch due to the small area covered by the patches. Thus, for the consideration of class balance for KPConv, an equal number of center points are selected from each class, and 50 k nearest neighboring points within these center points are further generated as the training patches. We pass the 3-D coordinates of point cloud along with two echo attributes (number of returns and return number) into the networks for semantic classification. Before training, various hyper-parameters need to be chosen appropriately for the deep learning algorithms. These include learning rate schedule, mini-batch size, duration of training, and architecture variables that determine the network structure such

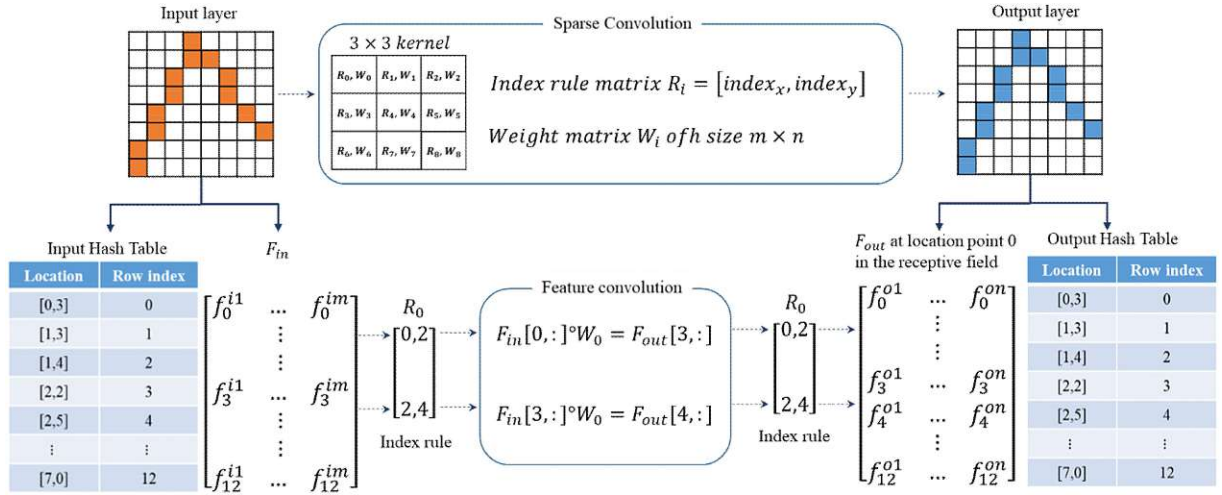


Fig. 6. Implementation of the convolution operation in SparseCNN.

TABLE II
PARAMETERS USED IN THE ENCODING OF POINTNET++

SA layers	Number of sampled points	Neighbors searching radius [m]	Number of Neighbors	MLP layer	Feature aggregation
Layer 1	8192	1	16	[64,64,128]	Max pooling
Layer 2	4096	5	64	[128,128,256]	Max pooling
Layer 3	2048	10	64	[128,128,256]	Max pooling

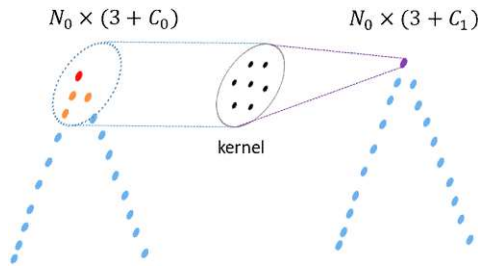


Fig. 7. Implementation of the convolution operation in KPConv.

as a number of layers. By default and if not otherwise specified, we apply a cosine decay learning rate schedule with the default setting for the three networks. The mini-batch size is set as 2 and the number of epochs is set as 18 for the training of SparseCNN, whereas we used the mini-batch size of 1 and 10 training epochs for the training of PointNet++ and mini-batch size of 1 and 12 epochs for KPConv. Thus the weights will be learned through more iterations in PointNet++ and KPConv than in SparseCNN. The encoder stage of PointNet++ contains three levels with different neighborhood sizes, and the neighboring points are randomly selected within the given radius. The parameters used in the encoder stage of PointNet++ are summarized in Table II. For SparseCNN the voxel size is defined as 8 cm and the encoder stage is composed of 7 levels with additional residual connections. For KPConv, the rigid kernel with 15 kernel points and a

grid-subsampling with a grid size of 0.08 m is applied in each level except for the first level. This means that the convolution is performed on raw input point clouds. In total 6 levels are used in the feature encoding of KPConv, and the convolution radius in each level is 0.2, 0.4, 0.8, 1.6, 3.2, and 6.4 m.

For the purpose of comparison, we selected the popular RF classifier as a baseline. To have as fair a comparison as possible, handcrafted features at multiple scales are computed for the RF classifier, so that context information can be included at multiple scales. For the classification of the Vienna ALS data, the geometrical features proposed by Weinmann [23] are employed. The features are computed using multiple KNN (k Nearest Neighbors) within a 3-D search radius of a maximum of 1 m, where the value of k is defined as 10, 20, and 30.

As for the power line point clouds, 2-D features derived from a vertically infinite cylindrical neighborhood are excluded, as those features cannot separate facility objects right above the surrounding ground or vegetation from those ground and vegetation points. Instead, three 2-D features are computed from a finite cylinder neighborhood, namely point count, height range, and height variation. The remaining 3-D features are the same as Weinmann's features used for Vienna. For the 3-D features, 4 different KNN neighborhood sizes are used: k is 10, 50, 100 within a radius of 0.5 m, as well as k is 100 within a maximum radius of 1 m. For 2-D features, six different finite cylinder neighborhood sizes are used: radius of 0.1 m with height of 0.2 and 0.5 m; radius of 0.5 m with height of 0.2, 0.5, and 1.5 m; and radius of 1 m with a height of 1.5 m.

TABLE III
AVERAGE AND STANDARD DEVIATION OF RECALL ON THE TEST TILES FROM THE VIENNA DATASET

Method	Ground (%)	Vegetation (%)	Buildings (%)	Others (%)	OA (%)
Random Forest	95.27±2.10	81.25±11.85	86.75±4.66	82.93±6.87	90.69±2.36
PointNet++	98.86±0.62	94.62±2.94	95.85±2.20	65.31±6.54	97.13±0.55
SparseCNN	99.07±0.39	96.47±1.60	97.49±1.41	78.65±5.12	98.12±0.41
KPConv	99.52±0.30	95.81±2.85	87.61±3.59	77.04±7.53	94.90±3.45

For both datasets, the same training tiles are used in PointNet++, SparseCNN, and KPConv. An alternative selection of training samples had to be used for RF because of limits on computation time. Here, an equal number of points per class are selected from each training tile and used for the training, as spatial continuous samples are not required by RF. For the Vienna point cloud, 50 000 samples per class are selected from each training tile for training, thus there is a total of 800 000 samples used for training ($4 \times 4 \times 50\,000$). For the power lines point clouds, 50 000 samples per class are selected from each area for training, and there is a total of 740 648 samples used for training, since the number of insulator points are fewer than 50 000 in some areas.

Since the deep learning methods take in a larger amount of training samples and cover larger context than RF, they very likely have better classification performance compared to RF. However, the main goal in this paper is not to compare different machine learning methods, but rather the two selected deep learning models.

D. Performance Metrics

For the evaluation of each test tile, the metrics recall and precision are computed for each class separately. Recall is the proportion of actual positives that is correctly classified. It expresses the ability of the classifier to identify all relevant instances. Precision is the proportion of predicted positives that is truly positive and measures the ability of the classifier not to predict as positive a sample that is actually negative.

To evaluate the methods on the entire dataset, we compute the average of recall and precision over all test tiles for each class. Additionally, the OA is measured as the micro-averaged recall of all test tiles, where the micro-averaged recall is computed by aggregating contributions of all classes. Besides, the standard deviation over these metrics is used to examine performance variations among the test tiles.

IV. EXPERIMENTS AND DISCUSSIONS

In this section, several classification experiments are presented based on PointNet++, SparseCNN, and KPConv as well as on RF, which is used as a baseline for the comparison. The classification results are presented and analyzed in Section IV-A. In addition, the computation time required for each classifier are given in Section IV-B and the classification uncertainty is discussed in Section IV-C. Furthermore, we explore the ability of the selected deep learning networks to generalize to new, unseen data in Section IV-D. Finally, the impacts of hyper-parameters of the three deep learning networks are investigated in Section IV-E.

TABLE IV
AVERAGE AND STANDARD DEVIATION OF PRECISION ON THE TEST TILES FROM THE VIENNA DATASET

Method	Ground (%)	Vegetation (%)	Buildings (%)	Others (%)
Random Forest	98.94±0.40	89.39±11.62	87.18±14.91	27.34±17.95
PointNet++	96.21±1.40	98.10±0.88	96.24±2.86	70.90±11.81
SparseCNN	98.46±0.50	98.53±0.66	96.81±1.70	71.61±12.16
KPConv	92.73±6.90	95.85±4.15	98.64±0.86	54.54±7.53

A. Classification Results and Discussion

1) *Vienna ALS Point Cloud*: For the Vienna ALS dataset, Tables III and IV provide the averages and standard deviations of the precision and recall metrics for all test tiles. Comparing to the results of RF, PointNet++, SparseCNN, and KPConv can all achieve a better classification performance with an OA up to 97.13%, 98.12%, and 94.90% respectively. As shown in Fig. 7, there is much salt and pepper noise in the classification result by RF, since it only relies on the handcrafted features which could become unstable at the boundaries of different objects or areas with lower point densities. The deep learning networks, especially PointNet++ and SparseCNN, can lead to a more homogenous classification result. Additionally, all the selected networks show great advantages in the prediction of the class *vegetation* and the accuracy of *buildings* is also significantly increased by PointNet++ as well as SparseCNN. The superior classification results by the selected networks suggest that deep learning can benefit from the gradually increasing receptive fields in the feature encoding, so that these high-level features with surrounding context can be effectively learned and lead to more accurate predictions, especially for those objects that are hard to characterize by a set of local geometric features.

Despite the improvement brought by the deep learning methods, we also found some defects in each deep learning network. Compared to SparseCNN and KPConv, PointNet++ has a difficulty in correctly detecting small objects like the meadow in the rectangle in Fig. 8(b), and this also leads to a low accuracy of *others*. Table V additionally presents the recall of *others* as well as the misclassification percentages of each test tile. This evaluation suggests that both SparseCNN and KPConv can differentiate the classes *others* and *ground* better than PointNet++, and by visual analysis, we found that the wrong labeling of *others* in PointNet++ mostly occurs at the junction areas between *others* objects and *ground*. One example is shown in Fig. 9, which presents a part of a playground area from Test tile 2. Both SparseCNN and KPConv can successfully identify the small infrastructures [black rectangle in Fig. 9(b) and (c)] with a height of around 1 m above ground, whereas PointNet++ misclassifies them as ground. This is probably caused by the

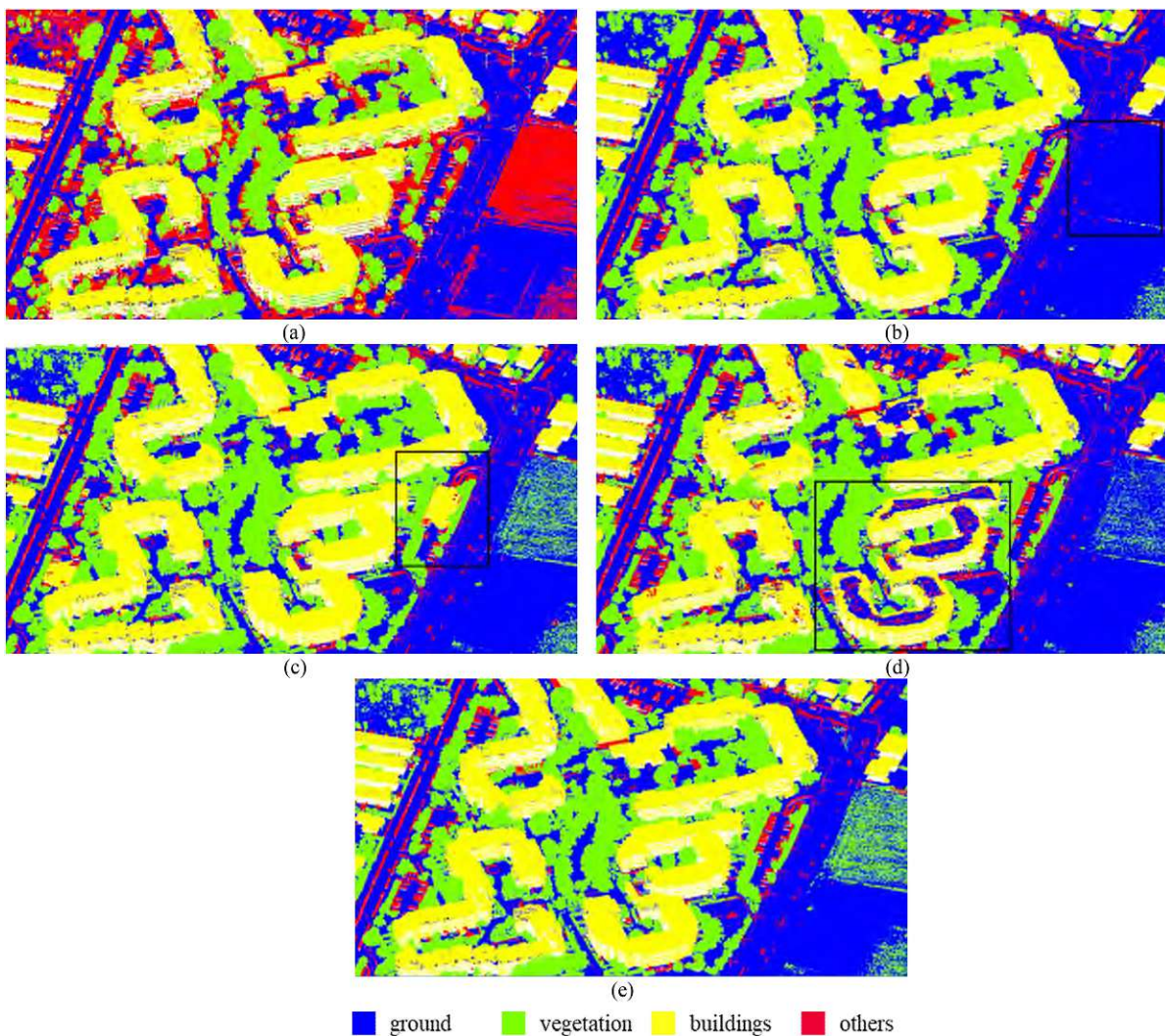


Fig. 8. Classification results of an area in Test tile 3. (a) Random Forest. (b) PointNet++. (c) SparseCNN. (d) KPConv. (e) Reference.

TABLE V
CORRECT CLASSIFICATION AND MISCLASSIFICATION PERCENTAGES OF OTHERS IN EACH TEST TILE, IN TERMS OF RECALL

Classification	models	Tile 1	Tile 2	Tile 3	Tile 4	Tile 5
Classified as Others	PointNet++	53.81%	68.24%	67.46%	63.63%	73.41%
	SpareCNN	68.86%	80.05%	81.12%	79.47%	83.77%
	KPConv	62.47%	77.76%	83.21%	79.61%	82.13%
Misclassified as ground	PointNet++	20.44%	12.68%	15.01%	16.53%	11.62%
	SpareCNN	14.24%	3.97%	2.90%	6.41%	4.59%
	KPConv	17.45%	5.83%	6.63%	9.55%	6.12%
Misclassified as vegetation	PointNet++	21.75%	13.82%	10.29%	3.81%	3.96%
	SpareCNN	12.71%	10.93%	7.55%	2.30%	3.16%
	KPConv	18.92%	13.87%	7.98%	4.82%	5.15%
Misclassified as buildings	PointNet++	4.00%	5.26%	7.25%	16.03%	11.01%
	SpareCNN	4.20%	5.05%	8.43%	11.83%	8.48%
	KPConv	1.16%	2.54%	2.19%	6.02%	6.60%

large initial neighborhood size used in PointNet++. A radius of 5 m is used by PointNet++ in the first layer, while SparseCNN and KPConv start with a relatively smaller neighborhood size with 0.08 and 0.2 m, respectively. Furthermore, in PointNet++ an interpolation using 3 nearest points is added for feature

backpropagation, which could result in an unstable classification at the boundaries of objects.

As for the performance of KPConv, a large number of building points, especially the inner parts that are far away from edges, are misclassified as *ground*, which can be seen in Fig. 8(d). The

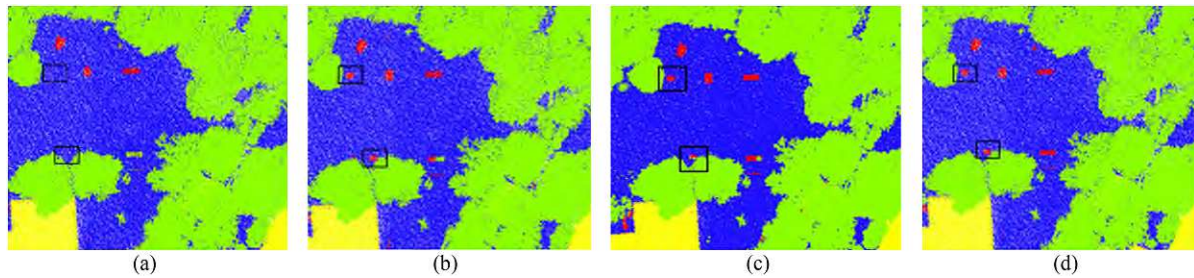


Fig. 9. Classification results of a playground in Test tile 2. (a) PointNet++. (b) SparseCNN. (c) KPCConv. (d) Reference.

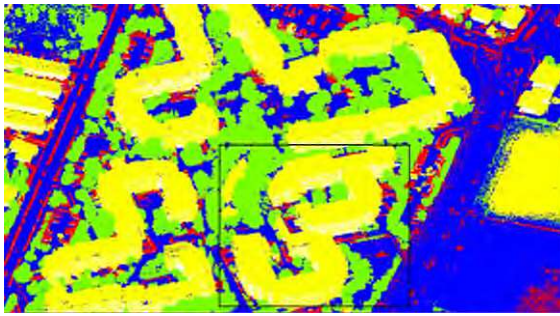


Fig. 10. Classification result by KPCConv on the subsampled point cloud.

main reason is that KPCConv in this case can only include a very local context to learn features. The largest radius of convolution of KPCConv at the deepest layer is 6.4 m, whereas the largest receptive field size in PointNet++ and SparseCNN can reach 10 and 51 m, respectively. Moreover, a much smaller area is covered by each patch used for training in KPCConv compared to PointNet++ and SparseCNN. The radius and the number of points in each patch are setup in this way to avoid OOM in KPCConv. Therefore, the roof points of which the neighborhood lacks the contextual information about other objects such as facade and ground, are easily misclassified as *ground*. Because of this, the infrastructures on the mislabeled roofs are wrongly predicted as *others*.

To further verify the above-mentioned argument regarding KPCConv, we performed a classification by KPCConv on a subset of the Vienna dataset and the point clouds in this subset has been subsampled with a grid size of 0.32 m. In this way, the patch that contains the same number of points can include a larger area. Thanks to the more sparse points, the radius of convolution can be doubled in each layer in KPCConv without causing any OOM problems, to include more global context. The classification result of the same area with Fig. 8 by KPCConv is presented in Fig. 10. By increasing the ranges of context that the convolution can access, the prediction of *buildings* has been greatly improved. Note that the classification result may be affected by the insufficient training samples, since only a subset of the Vienna dataset is used for training.

The overall best result is delivered by SparseCNN, however, we observed that it tends to misclassify some ground points as

buildings when there is a sudden height change, like the parking lot shown in the rectangle in Fig. 8(c). Moreover, since the receptive field size in SparseCNN can go up to 54 m in the last layer, the inclusion of large scale context may have a tendency of error expansions. For example, the object (rectangle in Fig. 11) is an underground café with a grass roof, which is labeled as *ground* in the reference because its roof is seen as a part of the ground. This object is classified as *buildings* by all investigated networks, as suggested by the context that these points rise up sharply without any transitions (jump edges between different levels of ground). Based on the prediction of the surroundings of this object, SparseCNN tends to expand the building points (possible erroneous points) to a larger area compared to PointNet++ and KPCConv.

2) *Power Line ALS Point Clouds*: For the power line dataset, the average classification accuracy over all test tiles is provided in Tables VI and VII in terms of precision and recall. The three deep learning networks outperform RF once again, concerning the average recall and precision on each class, as well as the OA. RF performs relatively competitive with the three deep learning networks in the classification of *conductors*, with the average difference in the recall at 3.31%, 3.36%, and 3.23% compared to PointNet++, SparseCNN, and KPCConv, respectively. This is because the linear characteristic of conductors facilitates the classification. Nevertheless, for the parts of conductors that are situated below the insulators, RF is inferior to the deep learning networks as illustrated in Fig. 12.

In accordance with our earlier observation of the Vienna dataset, we found the same type of erroneous labeling in this dataset for each classifier. Like the performance on the Vienna point cloud, RF tends to lead to heterogeneous classification results when compared to the three deep learning methods. As for the classification of insulators, the small objects in this case, PointNet++ performs not as good as the other two deep learning networks. As shown in Fig. 12, some points at the boundaries of insulators have been misclassified as conductors by PointNet++. Due to the error expansion of SparseCNN mentioned previously, some misclassification as the *environment* is also expanded to larger areas with this classifier, whereas we do not observe this for PointNet++, which achieves a recall of 97.15% in this example in Fig. 12. KPCConv faces a difficulty in the classification of pylons in this example which has a very high point density. Because of the limited number of input points, the test patches of this example tile can only cover a relatively small

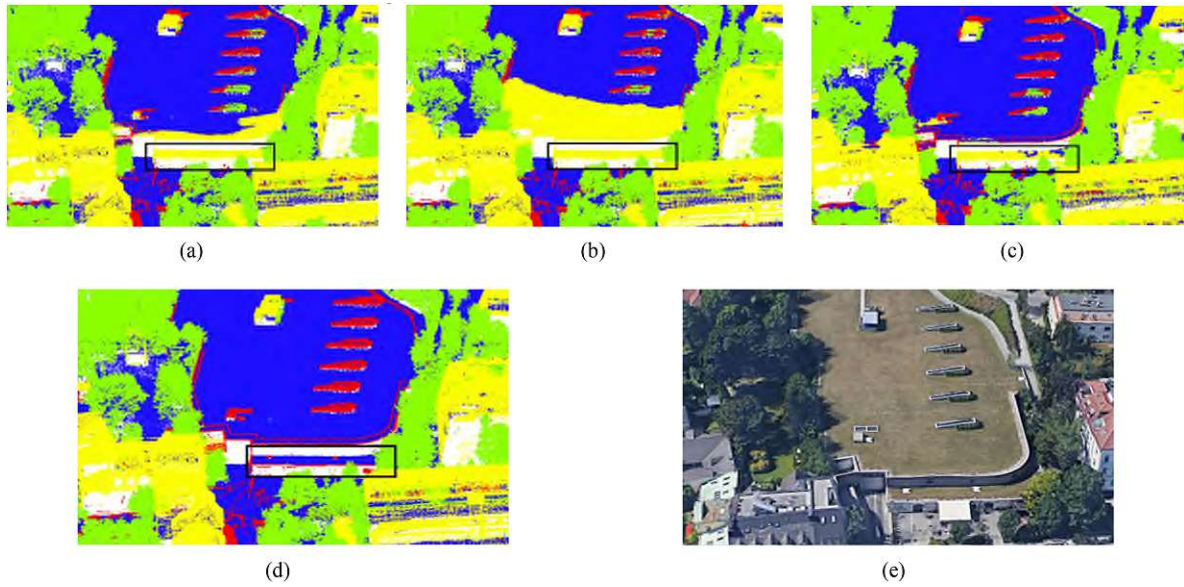


Fig. 11. Classification results of an area of an underground café in Test tile 2.

TABLE VI
AVERAGE AND STANDARD DEVIATION OF RECALL OF ALL TEST TILES

Method	Environment (%)	Conductors (%)	Pylons (%)	Insulators (%)	OA (%)
Random Forest	93.05±4.97	96.54±4.79	89.15±4.64	87.14±10.24	92.61±3.6
PointNet++	99.92±0.11	99.85±0.27	94.05±5.76	91.75±9.38	98.95±0.96
SparseCNN	99.94±0.05	99.90±0.07	97.41±2.6	99.76±0.16	99.55±0.48
KPConv	99.81±0.61	99.77±0.23	95.91±4.93	96.47±7.61	99.12±1.20

TABLE VII
AVERAGE AND STANDARD DEVIATION OF PRECISION OF ALL TEST TILES

Method	Environment (%)	Conductors (%)	Pylons (%)	Insulators (%)
Random Forest	97.51±2.13	98.56±1.36	75.44±22.36	48.09±25.41
PointNet++	98.65±2.07	99.34±0.75	99.33±0.85	94.41±5.26
SparseCNN	99.37±1.00	99.92±0.30	99.71±0.30	97.18±4.00
KPConv	98.75±2.38	99.90±0.19	99.46±0.54	95.09±5.90

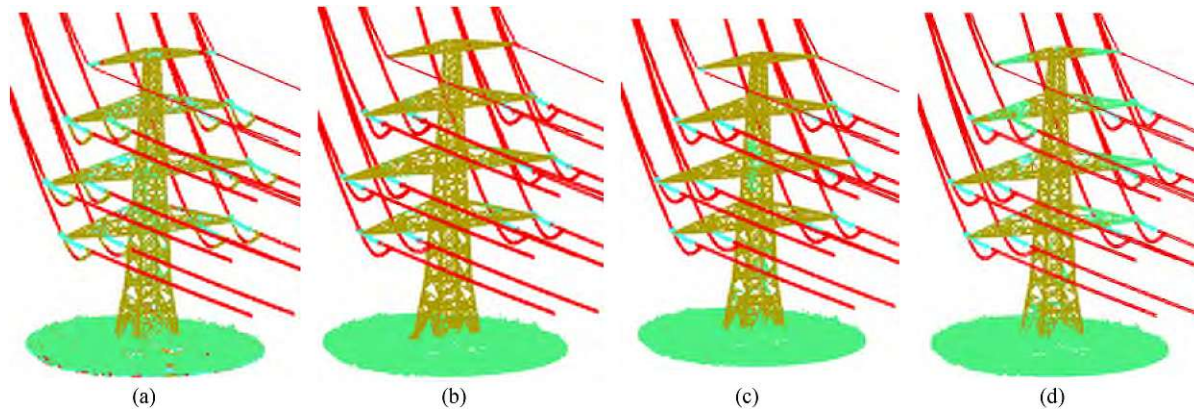


Fig. 12. Classification results of a test tile in Area_1 with the recall of pylons and insulators shown.

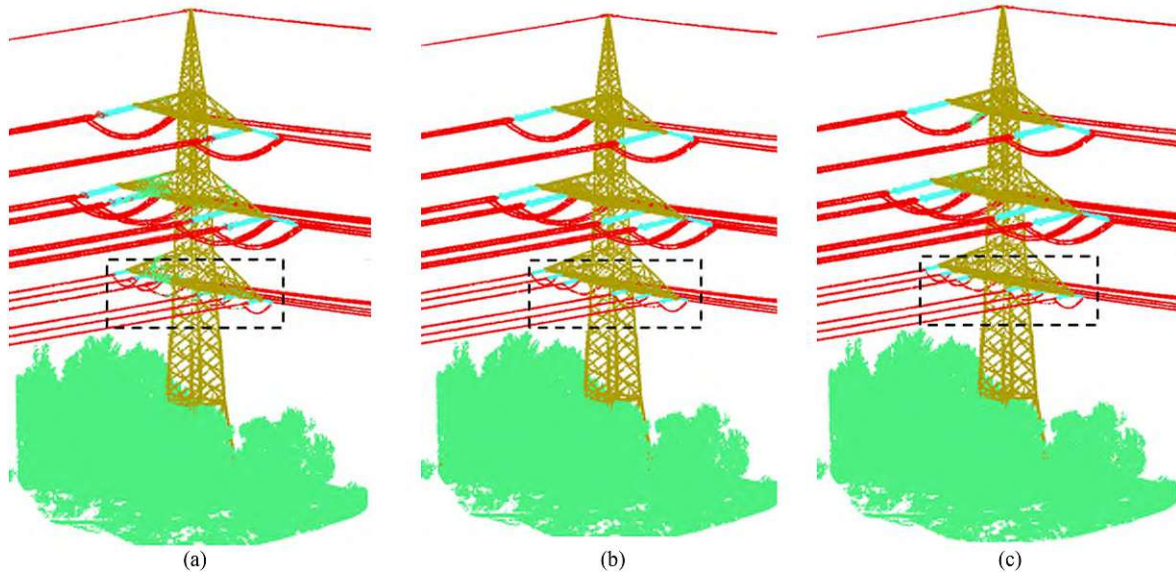


Fig. 13. Classification results of a test tile in Area_4.

TABLE VIII
PROCESSING TIME USED ON THE VIENNA POINT CLOUD

Processing steps		Random Forest	PointNet++	SparseCNN	KPConv
Preprocessing	Patch generation for training	0		4.9 h	1.3 h
	Patch generation for inference	0	3.6 h	0	1.5 h
	Handcrafted feature extraction	261.4 h	0	0	0
Training		5.3 h	158.4 h	82.8 h	88.1 h
Inference		15.1 h	34.1 h	54 min	6.5 h
Post-processing		0	15.4 h	0	12.8 h
Total		281.8 h	213.5 h	83.7 h	110.2 h

TABLE IX
PROCESSING TIME USED ON THE POWER LINE FACILITIES POINT CLOUDS

Processing steps		Random Forest	PointNet++	SparseCNN	KPConv
Preprocessing	Patch generation for training	0		2.1 h	1.8 h
	Patch generation for inference	0	35 min	0	20 min
	Handcrafted feature extraction	51.1 h	0	0	0
Training		2.9 h	110.8 h	12.9 h	28.5 h
Inference		58 min	2.5 h	2 min	40 min
Post-processing		0	1.3 h	0	1 h
Total		54.96 h	117.3 h	15.03 h	32.3 h

area, which thus cannot provide sufficient context for KPConv to learning effective features.

The accurate classification of insulators is the main challenge in this dataset, therefore another example is presented in Fig. 13 to examine the abilities of the three selected deep learning methods to detect insulators. In this example, PointNet++ assigns incorrect labels to the insulators with relatively small size that are located on the lowest crossarm, while both SparseCNN and KPConv can deliver accurate inference for those insulators regardless of the difference in size.

B. Computation Time

Tables VIII and IX present the computation time that the aforementioned four classifiers need for each processing step

on the two datasets. The training tiles are divided into patches that only cover partial areas of the tiles, so that they can serve as the input for the deep learning networks. The inference for PointNet++ and KPConv is implemented on the patches to avoid OOM. Besides the consideration of memory demand, PointNet++ requires the similar scales of training and inference patches in order to keep the point density of training samples same with that of inference samples in each set abstraction after FPS. Hence, patches are additionally generated for inference and a postprocessing step that merges patches into complete tiles is added for PointNet++ and KPConv. Note that there are 10 training epochs in PointNet++, 12 training epochs in KPConv, but 18 training epochs in SparseCNN.

The implementations of the deep learning networks and RF, which we use in our comparison, require, and exploit different

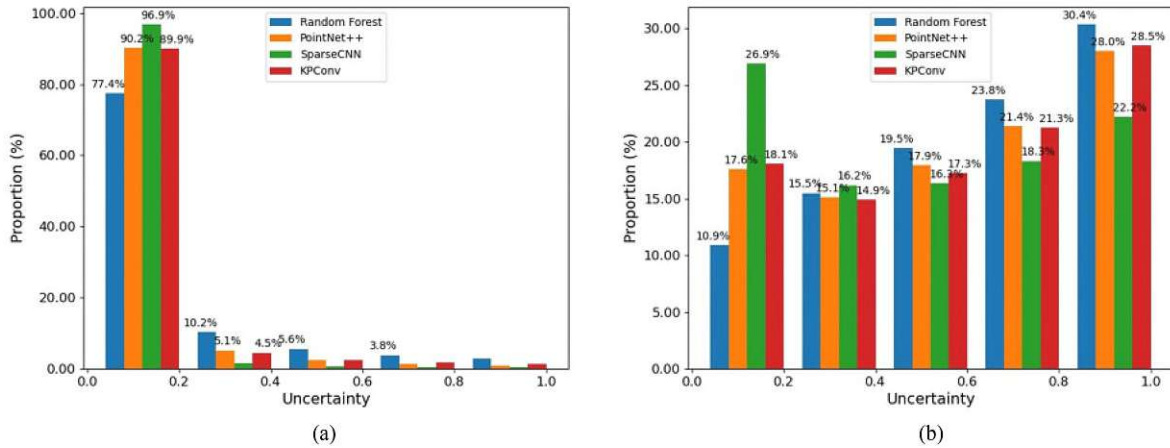


Fig. 14. Uncertainty distribution. (a) Uncertainty histogram of correctly labeled points. Only bars of values more than 3% are labeled with the numbers. (b) Uncertainty histogram of incorrectly labeled points.

hardware. The RF experiments were run on a computer equipped with an AMD Ryzen 7 2700 \times (3.70 GHz) processor, 32 GB RAM, and no GPU was involved for the processing. In contrast, we use another machine for the deep learning experiments with AMD Ryzen Threadripper 1900 \times (3.5 GHz) processor, 512 GB RAM, and an Nvidia RTX 2080 Ti with 11 GB RAM.

As shown in Tables VIII and IX, the processing of SparseCNN is much faster than RF and the other two models because the neighborhood lookup is much more efficient using voxelization and hash tables. Additionally, this method also demands less memory than PointNet++ and KPConv. In the case of the power lines dataset, the memory consumption is 1985 MB, 8809 MB, 8575MB for SparseCNN, PointNet++, and KPConv, respectively. KPConv is more memory-demanding than PointNet++, and thus KPConv fails when a large number of points is taken as input.

As for RF, the computation of the handcrafted features accounts for the majority of the processing. This step is especially time-consuming when large volume point clouds are given. For the Vienna dataset, the feature computation took around 10 days, which is even longer than the training time of PointNet++ and makes RF the most time-demanding model among the four selected classifiers. However, the actual training of the RF classifier with precomputed features is faster than the training of the deep learning models. Regarding inference, the fastest results are achieved by SparseCNN, whereas PointNet++ and KPConv need the postprocessing for merging, thus they take patches also takes extra time to achieve the final classification results.

C. Classification Uncertainty Analysis

Another aspect of machine learning models is their ability of handling uncertainty, which we assess by analyzing the difference between the highest and the second-highest posterior class probabilities [74]. The uncertainty should be capable to reflect the classification confidence: ideally, a high uncertainty indicates a high possibility of misclassification, whereas a low

uncertainty implies a correct classification. Accordingly, a high uncertainty is expected for incorrectly labeled points and a low uncertainty is expected for correctly labeled points.

Fig. 14 provides the uncertainty histograms of correctly and incorrectly labeled points for RF and the three deep learning networks. The histograms are normalized, i.e., for each classifier, the values of correctly labeled points add up to 100%, and also the values for the incorrectly labeled points add up to 100%.

The uncertainty histogram of correctly labeled points shows a clear preference that most of the correctly labeled points also have a low uncertainty score, especially for the deep learning models. More than (or close to) 90% of correctly labeled points' uncertainty is in the range of 0 to 0.2. However, it is hard to find a dominant range of the uncertainty for the incorrectly labeled points. In contrast to the two deep learning models, the misclassified points of RF show a weak but obvious trend that lower uncertainty correlates with higher misclassification rates. However, there are no significant differences between the proportion of high uncertainty (in the range of 0.8 and 1.0) and the proportions in the other uncertainty ranges. For SparseCNN a large number of misclassified points (26.9%) have a low uncertainty (in the range of 0 and 0.2). The statistic of the uncertainty implies that the three deep learning models, especially SparseCNN, tend to give slightly over-confident classification results on the misclassified points compared to RF.

The uncertainty derived from the posterior class probabilities should enable to identify misclassified or informative points. To verify this for the aforementioned classifiers, points are selected by different uncertainty ranges, and the proportion of incorrectly labeled points is presented in Fig. 15. The plot indicates that the uncertainty is not informative for identifying misclassified points from the point-wise classification results. Even for the points that are selected by a high uncertainty (e.g., 0.9), it is only a fifty-fifty chance of filtering misclassified points for the three classifiers. However, from the perspective view of active learning, those point with high uncertainty should be informative and helpful for the training refinement. The usage of the uncertainty needs to be studied further.

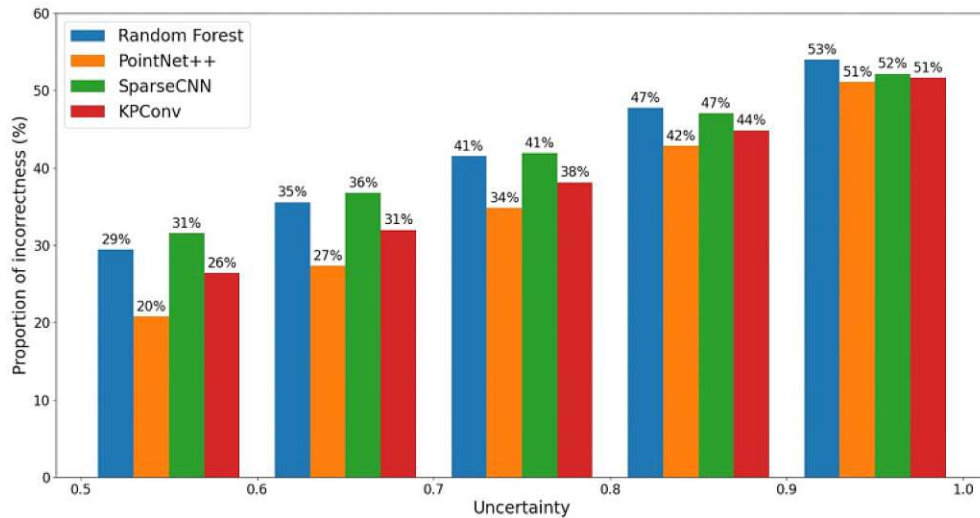


Fig. 15. Proportion of incorreced labeled points.

D. Classification Generalization Ability

The ability of a classifier to generalize describes how well a model can produce reasonable predictions for samples that were not encountered in the training dataset. We therefore assess the generalization ability of PointNet++, SparseCNN, and KPConv using the power lines facilities dataset, which contains point clouds collected in different areas during four different flight campaigns. Due to different flight trajectories relative to the power line, there is a major difference in point density between these tiles. To examine the generalization ability, only the tiles of Area_1 are used for the training, then the trained network is evaluated on tiles from Area_2, Area_3, and Area_4, respectively.

Fig. 16 shows the OA and the recall per class achieved by the three deep learning methods. In most novel areas, PointNet++ achieves a comparable classification accuracy with the other two networks in terms of the three dominant classes (environment, conductors, and pylons), as well as the OA. But in all test areas, PointNet++ performs much worse than SparseCNN and KPConv on the prediction of the class *insulators*. Compared to the dominant classes, this class has a smaller object size and the point clouds are therefore affected more significantly by variations in point density. There are also more tension insulators in Area_2 and Area_4 than in Area_3, which are harder to detect compared to the suspension insulators that are vertically hanging off the pylons. Thus the accuracy of insulators of those two areas are worse than for Area_3, despite the point density of Area_3 being much lower than in training Area_1. Additionally, in the classification results of PointNet++ the prediction of the class *environment* in Area_2 is not as good as for the other test areas, and about 9% below the accuracy achieved by the other two networks. The reason is probably that a large amount of dense vegetation appears in Area_2.

In KPConv, a grid-subsampling is implemented in each layer (except for the first layer), which allows input point clouds to have a consistent point density in the network. This enables

KPConv to be less impacted by varying density to some extent. However, KPConv leads to a less accurate classification of conductors in Area_4, which may be caused by the high density of the point clouds in Area_4. This also corroborates the findings of the classification in the Vienna dataset.

Overall, the evaluation on the power line facilities point clouds indicates that SparseCNN generalizes better than the other two networks, since the representation of voxels is intrinsically robust to varying density of points.

Note that the difference of point density has to be in a reasonable range when the training and test point clouds are collected from different campaigns. We also attempted to apply the model trained by the Vienna dataset to the Vaihingen dataset (ISPRS benchmark), but all models, PointNet++, SparseCNN, and KPConv, lead to poor classification results on the Vaihingen dataset, because its point density is much lower than in the used training dataset. For PointNet++ and KPConv, the points in the neighborhood used for feature learning changed notably. As for SparseCNN, a larger voxel size needs to be used to avoid too many empty voxels on the Vaihingen dataset.

E. Impact of Hyper-Parameters

We next evaluate the hyper-parameters of the methods that are external to the model. Their values cannot be estimated by training samples but are set by operators before training. Thus, apart from the learning rate schedule, mini-batch size, and epochs, the selections of initial input features or class balance strategies are also included in the hyper-parameters.

Our experiments to study the impact of hyper-parameters are focused on a subset of the power line dataset. Since the pylons in Area_2 are similar to the pylons in Area_1, our subset excludes the training tiles of Area_2 in order to reduce processing time. For the evaluation, the subset contains four test tiles of Area_1, 3 test tiles of Area_3, and 4 test tiles of Area_4. Based on a large number of experiments, we selected four important

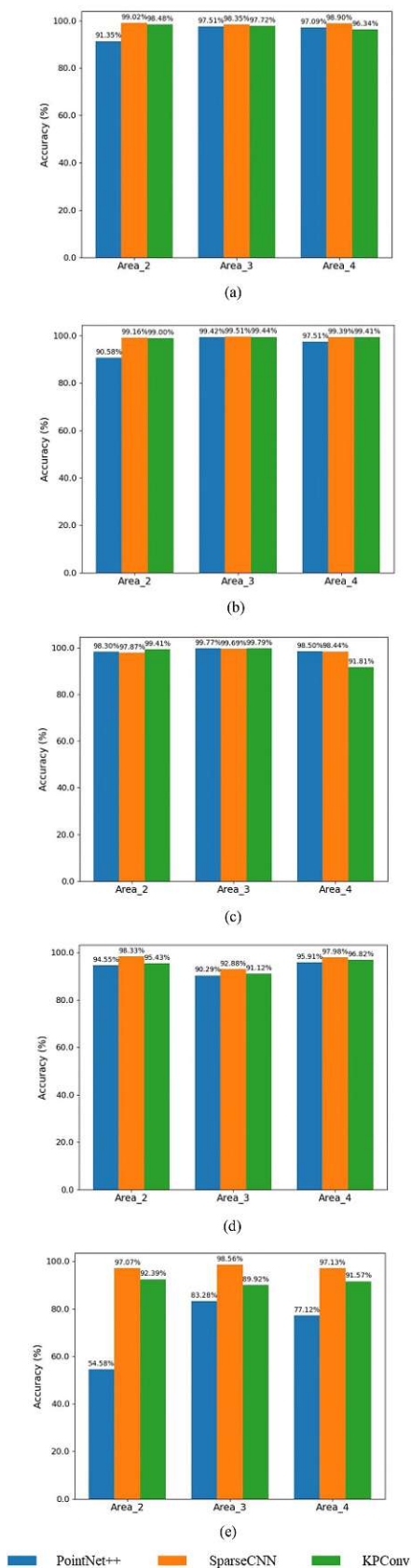


Fig. 16. Accuracy of classification results of three areas.

hyper-parameters to discuss, namely learning rate schedule, input features, class balance strategies, and architecture.

The baseline configuration is described in Section III and Table X presents the different configurations of the selected hyper-parameters, as well as the baseline configuration. In each experiment, only one type of hyper-parameters changes, the others are fixed to the baseline configuration.

In the class balance strategies, the filtering of training patches only selects patches that contain points of all four target classes as input, the weighted loss function is the same as mentioned in Section III. For PointNet++, the structure of the architecture is mainly defined by the selection of search neighborhoods, other parameters like the MLP structures and pooling are kept constant. Likewise, for SparseCNN, only different numbers of levels are explored in terms of the architecture, whereas the operators on each level remain fixed. As for KPCConv, the different radius of convolution and number of levels are investigated.

The accuracy difference is given as the difference in OA between test configurations and baseline configuration in each tile. Fig. 17 displays the distributions of accuracy differences over 11 test tiles by a boxplot based on a five number summary (minimum, first quartile, median, third quartile, and maximum), and the average values are depicted by red squares. Besides the OA differences, Fig. 17 also provides the accuracy differences of insulators to inspect the impact of the class balancing strategies, another reason is that the prediction of insulators is more challenging than the other classes and is easier influenced by the hyper-parameters configurations.

Overall, the accuracy achieved by SparseCNN varies much less than the accuracy achieved by PointNet++ and KPCConv.

Different learning rate decay methods have quite a small impact on PointNet++ regarding to the OA, but they tend to cause unstable classification performances of insulators. We observed that the accuracy of insulators drops largely on a few individual tiles when using constant learning rate or step-based learning rate decay. For both SparseCNN and KPCConv, the constant learning rate performs worse than the two learning rate decay methods on the classification of insulators.

Regarding the input features, intensity shows no significant impact on the performances of the selected deep learning models. However, the classification accuracy of PointNet++ decreases notably when only x, y, z coordinates are used as input. The largest accuracy reduction for SparseCNN is also found for insulators when using only x, y, z coordinates. By using only coordinates as input, KPCConv has a similar performance in the change of insulator accuracy like SparseCNN, where the maximum insulator accuracy decrease is 3.88% and 2.98% for KPCConv and SparseCNN, respectively.

The class balance strategies have no significant impact on PointNet++ and SparseCNN based on the average accuracy differences, however, they have a great influence on KPCConv. Without the filtering of training tiles, the performance by KPCConv, especially on the rarest class of insulators drops dramatically on some individual tiles. Using the weighted loss function only provides very limited help to improve the classification

TABLE X
DESCRIPTION OF THE HYPER-PARAMETERS

Hyper-parameters	Short names	Description		
Learning rate decay	LR[constant]	Initial LR=0.01, and the LR is fixed during training		
	LR[step-based]	Initial LR=0.01 The minimal LR is 0.00001. The LR decreases based on a step decay function, where the decay step is 500 and decay rate is 0.7.		
	LR[baseline]	Initial LR=0.01 The minimal LR is 0.00001. The LR decreases based on a cosine decay function, where the decay step is the total number of iterations. This means the LR gradually drops and reaches minimal LR until the end of training.		
Input features	Feat[xyz]	Only x,y,z are used as initial input features		
	Feat[intst]	Besides x,y,z coordinates, echo number, number of echoes and intensity are used as initial input features. Note that the intensity is normalized per patch to keep the intensity scaling consistent over tiles from different areas		
	Feat[baseline]	Besides x,y,z coordinates, echo number and number of echoes are used as initial input features.		
Class balance	CB[None]	All training tiles are used and an equal weight of each class is used in the loss function.		
	CB[weight]	All training tiles are used and different weights of classes are used in the loss function.		
	CB[filter]	The training tiles are filtered and an equal weight of each class is used in the loss function.		
	CB[baseline]	The training tiles are filtered and different weights of classes are used in the loss function.		
Architecture structures		PointNet++	SparseCNN	KPConv
	Arch[0]	Three layers, Neighbors searching radius=[1,3,15] Number of neighbors=[16,32,64]	Number of levels=4	Number of levels=6 radius=[0.4,0.8,1.6,3.2,6.4,12.8]
	Arch[1]	Three layers, Neighbors searching radius=[1,5,15] Number of neighbors=[16,64,64]	Number of levels=5	Number of levels=5 radius=[0.2,0.4,0.8,1.6,3.2,6.4]
	Arch[2]	Four layers, Neighbors searching radius=[1,5,15,15] Number of neighbors=[16,64,64,64]	Number of levels=6	Number of levels=7 radius=[0.2,0.4,0.8,1.6,3.2,6.4]
	Arch[baseline]	Three layers, Neighbors searching radius=[1,5,10] Number of neighbors=[16,64,64]	Number of levels=7	Number of levels=6 radius=[0.2,0.4,0.8,1.6,3.2,6.4]

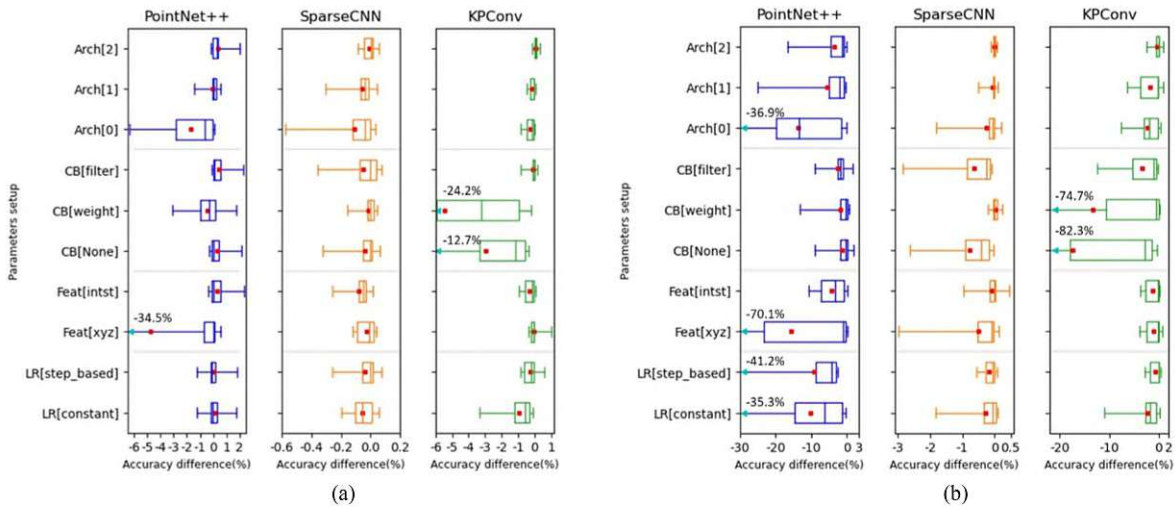


Fig. 17. Accuracy difference of the test tiles to the baseline, presented as box plots (minimum, maximum, first and third quartile, median) and mean value (red square). Cyan triangles indicate values that are beyond the limits of the x-axis and those values are also labeled. Note the different scales. (a) OA difference. (b) Accuracy differences of insulators.

of *insulators*. As for SparseCNN, the weighted loss function is particularly helpful in improving the prediction of insulators. For PointNet++, we observed an accuracy decrease at pylons in some individual tiles when using the weighted loss function, where the largest accuracy difference of pylons reaches -13.9% .

For the architectural structures, the results of PointNet++ are obviously impacted by the neighborhood selection, with a trend to better and more consistent performance with larger neighborhood areas. SparseCNN also performs better with more levels in the network, which likewise increases the neighborhood used for classifying each voxel. These results are consistent with common knowledge of deep learning in general. Moreover, since the accuracy difference between the structure using 6-layers and that of 7-layers is already small, we did not experiment with any more additional layers. For this dataset, KPConv favors a smaller convolution radius in order to achieve an accurate detection of insulators. The improved classification by adding more levels in the KPConv also accords with an earlier observation regarding SparseCNN.

In a nutshell, SparseCNN is less impacted by the selection of hyper-parameters than the other two networks. For PointNet++, the classification performance is significantly influenced by input features and architectural structures, and the learning rate decay methods are important for the feature learning of small objects, such as insulators. For KPConv, an appropriate class balance strategy needs to be adapted so that less dominant classes can be correctly classified in various scenes. For all the three networks, a more capable model can be trained by gradually decreasing learning rates and cosine-based decay is a safe choice, as it has fewer parameters to tune than a step-based decay. Moreover, echo attributes are helpful to improve the classification performance for all the mentioned networks.

V. CONCLUSION

This study presented the classification performance of PointNet++, SparseCNN, and KPConv on two very different ALS point clouds and conducted a comparison among PointNet++, SparseCNN, and KPConv in several aspects.

- 1) Overall SparseCNN has achieved a better classification accuracy than PointNet++ and KPConv. PointNet++ has a less accurate prediction on the objects with small sizes, like the *others* in the Vienna point cloud and the *insulators* in the power line facilities point clouds. KPConv has a difficulty in the classification of large-size objects in a dense point cloud, like the buildings in the Vienna dataset. Due to the demanding memory usage, only limited number of points can be fed into KPConv, which leads to a lack of sufficient contextual information for the features learning in large-scale datasets. Although SparseCNN may have a risk of error expansion, it has resulted in the overall best classification performance. Moreover, SparseCNN can preserve object boundaries and detail information very well by using a reasonable small voxel size, although SparseCNN performs classification on voxels instead of directly on points.

- 2) SparseCNN requires much less time and memory than PointNet++ and KPConv on training and inference. Compared to RF, SparseCNN improves the efficiency particularly in the classification processing of large amounts of point clouds, because handcrafted feature computation is quite time-consuming for the large-scale point clouds. The demanding memory usage of KPConv limits the number of input points, which could further affect its classification performance, especially on large-scale dense point clouds. The uncertainty estimated by the three investigated classifiers cannot fully reflect how accurate classification results are.
- 3) Both SparseCNN and KPConv show better generalization ability than PointNet++. Specifically, the experiments in the paper suggested that SparseCNN and KPConv can achieve a more robust model than PointNet++ regardless of the point density differences between training and test point clouds.
- 4) SparseCNN is less impacted by the different selections of hyper-parameters than PointNet++ and KPConv judged by the accuracy differences. For PointNet++, the neighborhood size in each layer needs to be chosen well. For KPConv, a certain class balance strategy on the generation of training patches is important, and the weighted loss function works not very well in improving the accuracy of less dominant classes. The training of all networks can both benefit from the learning rate decays and input features with additional echo attributes.

In summary, the classification results show the great potential of the deep learning models for classifying ALS point clouds. For classical machine learning, the input handcrafted features need to be well designed for the specific classification scenes, whereas more representative features can directly be learned by the deep learning models and lead to better classification performance. The experiments proved that SparseCNN can achieve better performance at a very efficient runtime in the classification of ALS point clouds, compared to PointNet++ and KPConv. KPConv has a slightly better performance than PointNet++ in the classification of the power line facilities, but it suffers from the demanding memory usage.

The results are also in line with the statements in Section II that CNNs are more capable than MLPs in semantic classification tasks, as the convolution naturally takes spatial connectivity into account, which is ignored by MLPs. Furthermore, the high classification accuracy that SparseCNN achieves for small objects such as cars and insulators, indicate that for ALS point clouds any losses in accuracy caused by voxelization are negligible. In fact, the voxelization leads to an improved efficiency, hence enables the use of more complex architectures and a larger context radius, and such factors lead to an overall increase in classification accuracy.

However, for any practical application, a lot of effort has to be put into investigating the hyper-parameter in order to choose optimal settings for the deep learning models. The training of deep learning models heavily relies on sufficient training samples, thus the Vienna tiles used in this article are published to provide the benchmark data for the deep

learning or machine learning classification on ALS point clouds.

ACKNOWLEDGMENT

The authors acknowledge TU Wien Bibliothek for financial support through its Open Access Funding Programme. The authors thank MA41 for kindly providing the ALS point clouds and agreeing to publish tiles that are used in this article. They also would like to thank Siemens for kindly providing the power line facilities point clouds in this study.

REFERENCES

- [1] G. Tran, D. Nguyen, M. Milenkovic, and N. Pfeifer, "Potential of full wave-form airborne laser scanning data for urban area classification—transfer of classification approaches between missions," *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. XL-7/W3, pp. 1682–1750, 2015.
- [2] J. Secord and A. Zakhor, "Tree detection in urban regions using aerial lidar and image data," *IEEE Geosci. Remote Sens. Lett.*, vol. 4, no. 2, pp. 196–200, Apr. 2007.
- [3] C. Mallet, F. Bretar, M. Roux, U. Soergel, and C. Heipke, "Relevance assessment of full-waveform lidar data for urban area classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 66, no. 6, pp. 71–84, 2011.
- [4] L. Guo, N. Chehata, C. Mallet, and S. Boukir, "Relevance of airborne lidar and multispectral image data for urban scene classification using random forests," *ISPRS J. Photogrammetry Remote Sens.*, vol. 66, no. 1, pp. 56–66, 2011.
- [5] J. Niemeyer, F. Rottensteiner, and U. Soergel, "Contextual classification of lidar data and building object detection in urban areas," *ISPRS J. Photogramm. Remote Sens.*, vol. 87, pp. 152–165, 2014.
- [6] G. Vosselman, M. Coenen, and F. Rottensteiner, "Contextual segment-based classification of airborne laser scanner data," *ISPRS J. Photogramm. Remote Sens.*, vol. 128, pp. 354–371, 2017.
- [7] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review," *IEEE Access*, vol. 7, pp. 19143–19165, 2019.
- [8] A. Garcia-Garcia, S. Orts-Escobedo, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A review on deep learning techniques applied to semantic segmentation," 2017, *arXiv:1704.06857*.
- [9] H. Liang, X. Sun, Y. Sun, and Y. Gao, "Text feature extraction based on deep learning: A review," *EURASIP J. Wireless Commun. Netw.*, vol. 2017, no. 1, pp. 1–12, 2017.
- [10] R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.
- [11] B. Graham, M. Engelcke, and L. van der Maaten, "3D semantic segmentation with submanifold sparse convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9224–9232.
- [12] S. Schmohl and U. Soergel, "Submanifold sparse convolutional networks for semantic segmentation of large-scale als point clouds," *ISPRS Ann. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 4, pp. 77–84, 2019.
- [13] L. Winiwarter, G. Mandlbürger, S. Schmohl, and N. Pfeifer, "Classification of ALS point clouds using end-to-end deep learning," *PFG–J. Photogramm., Remote Sens. Geoinf. Sci.*, vol. 87, no. 3, pp. 75–90, 2019.
- [14] W. Li, F.-D. Wang, and G.-S. Xia, "A geometry-attentional network for ALS point cloud classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 164, pp. 26–40, 2020.
- [15] C. Wen, L. Yang, X. Li, L. Peng, and T. Chi, "Directionally constrained fully convolutional neural network for airborne LiDAR point cloud classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 162, pp. 50–62, 2020.
- [16] M. Baltruschat, H. Nickisch, M. Grass, T. Knopp, and A. Saalbach, "Comparison of deep learning approaches for multi-label chest X-ray classification," *Sci. Rep.*, vol. 9, no. 1, pp. 1–10, 2019.
- [17] H. A. Arief, U. G. Indahl, G.-H. Strand, and H. Tveite, "Addressing overfitting on point cloud classification using atrous XCRF," *ISPRS J. Photogramm. Remote Sens.*, vol. 155, pp. 90–101, 2019.
- [18] M. Yousefhusien, D. J. Kelbe, E. J. Ientilucci, and C. Salvaggio, "A multi-scale fully convolutional network for semantic labeling of 3D point clouds," *ISPRS J. Photogramm. Remote Sens.*, vol. 143, pp. 191–204, 2018.
- [19] N. Li, N. Pfeifer, and C. Ressel, "Tiles of airborne laser scanning point clouds of Vienna, Austria (2016)" [Data set]. Zenodo, 2021. [Online]. Available: <http://doi.org/10.5281/zenodo.4777087>
- [20] C.-H. Lin, J.-Y. Chen, P.-L. Su, and C.-H. Chen, "Eigen-feature analysis of weighted covariance matrices for LiDAR point cloud classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 94, pp. 70–79, 2014.
- [21] R. Blomley, B. Jutzi, and M. Weinmann, "3D semantic labeling of ALS point clouds by exploiting multi-scale, multi-type neighborhoods for feature extraction," in *Proc. Int. Conf. Geogr. Object-Based Image Anal.*, 2016, pp. 1–8.
- [22] B. Höfle, W. Mücke, M. Dutter, M. Rutzinger, and P. Doringner, "Detection of building regions using airborne LiDAR—A new combination of raster and point cloud based GIS methods," in *Proc. Int. Conf. Appl. Geoinformatics*, 2009, pp. 66–75.
- [23] M. Weinmann, S. Urban, S. Hinz, B. Jutzi, and C. Mallet, "Distinctive 2D and 3D features for automated large-scale scene analysis in urban areas," *Comput. Graph.*, vol. 49, pp. 47–57, 2015.
- [24] K. Khoshelham, S. O. Elberink, and S. Xu, "Segment-based classification of damaged building roofs in aerial laser scanning data," *IEEE Geosci. Remote Sens. Lett.*, vol. 10, no. 5, pp. 1258–1262, Sep. 2013.
- [25] S. K. Lodha, D. M. Fitzpatrick, and D. P. Helmbold, "Aerial lidar data classification using adaboost," in *Proc. 6th Int. Conf. 3-D Digit. Imag. Model.*, 2007, pp. 435–442.
- [26] S. Xu, G. Vosselman, and S. O. Elberink, "Multiple-entity based classification of airborne laser scanning data in urban areas," *ISPRS J. Photogramm. Remote Sens.*, vol. 88, pp. 1–15, 2014.
- [27] D. Anguelov *et al.*, "Discriminative learning of Markov random fields for segmentation of 3d scan data," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 169–176.
- [28] J. Niemeyer, F. Rottensteiner, and U. Soergel, "Conditional random fields for lidar point cloud classification in complex urban areas," *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. 3, pp. 263–268, 2012.
- [29] S. Laible, Y. N. Khan, and A. Zell, "Terrain classification with conditional random fields on fused 3D LIDAR and camera data," in *Proc. Eur. Conf. Mobile Robots*, 2013, pp. 172–177.
- [30] C. Luo and G. Sohn, "Scene-layout compatible conditional random field for classifying terrestrial laser point clouds," *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. 2, no. 3, 2014, Art. no. 79.
- [31] J. Jung, L. Chen, G. Sohn, C. Luo, and J.-U. Won, "Multi-range conditional random field for classifying railway electrification system objects using mobile laser scanning data," *Remote Sens.*, vol. 8, no. 12, 2016, Art. no. 1008.
- [32] J. Niemeyer, F. Rottensteiner, U. Soergel, and C. Heipke, "Hierarchical higher order CRF for the classification of airborne LIDAR point clouds in URBAN areas," *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.-ISPRS Arch.*, vol. 41, pp. 655–662, 2016.
- [33] D. Maturana and S. Scherer, "Voxnet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 922–928.
- [34] G. Riegler, A. Osman Ulusoy, and A. Geiger, "Octnet: Learning deep 3D representations at high resolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3577–3586.
- [35] R. Klokov and V. Lempitsky, "Escape from cells: Deep kd-networks for the recognition of 3D point cloud models," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 863–872.
- [36] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou, "Tangent convolutions for dense prediction in 3D," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3887–3896.
- [37] H. Su *et al.*, "Splatnet: Sparse lattice networks for point cloud processing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2530–2539.
- [38] M. Kiefel, V. Jampani, and P. V. Gehler, "Permutohedral lattice CNNs," 2014, *arXiv:1412.6618*.
- [39] V. Jampani, M. Kiefel, and P. V. Gehler, "Learning sparse high dimensional filters: Image filtering, dense CRFs and bilateral neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4452–4461.
- [40] Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-voxel CNN for efficient 3D deep learning," 2019, *arXiv:1907.03739*.
- [41] H. Tang *et al.*, "Searching efficient 3D architectures with sparse point-voxel convolution," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 685–702.
- [42] F. Zhang, J. Fang, B. Wah, and P. Torr, "Deep fusionnet for point cloud semantic segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, Art. no. 6.
- [43] R. Cheng, R. Razani, E. Taghavi, E. Li, and B. Liu, "2-S3Net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network," 2021, *arXiv:2102.04530*.

- [44] J. Xu, R. Zhang, J. Dou, Y. Zhu, J. Sun, and S. Pu, "RPVNet: A deep and efficient range-point-voxel fusion network for LIDAR point cloud segmentation," 2021, *arXiv:2103.12978*.
- [45] V. E. Liong, T. N. T. Nguyen, S. Widjaja, D. Sharma, and Z. J. Chong, "AMVNet: Assertion-based multi-view fusion network for LIDAR semantic segmentation," 2020, *arXiv:2012.04934*.
- [46] I. Alonso, L. Riazuelo, L. Montesano, and A. C. Murillo, "3D-mininet: Learning a 2D representation from point clouds for fast and efficient 3D LIDAR semantic segmentation," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5432–5439, Oct. 2020.
- [47] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.
- [48] M. Jiang, Y. Wu, T. Zhao, Z. Zhao, and C. Lu, "Pointsift: A sift-like network module for 3D point cloud semantic segmentation," 2018, *arXiv:1807.00652*.
- [49] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "PointWeb: Enhancing local neighborhood features for point cloud processing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5565–5573.
- [50] J. Li, B. M. Chen, and G. H. Lee, "So-net: Self-organizing network for point cloud analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9397–9406.
- [51] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, "PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 5589–5598.
- [52] L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4558–4567.
- [53] H. Liu, Y. Guo, Y. Ma, Y. Lei, and G. Wen, "Semantic context encoding for accurate 3D point cloud segmentation," *IEEE Trans. Multimedia*, vol. 23, pp. 2054–2055, 2021.
- [54] M. Feng, L. Zhang, X. Lin, S. Z. Gilani, and A. Mian, "Point attention network for semantic segmentation of 3D point clouds," *Pattern Recognit.*, vol. 107, 2020, Art. no. 107446.
- [55] Y. Ma, Y. Guo, H. Liu, Y. Lei, and G. Wen, "Global context reasoning for semantic segmentation of 3D point clouds," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2020, pp. 2931–2940.
- [56] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.
- [57] Q. Hu *et al.*, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11108–11117.
- [58] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8895–8904.
- [59] A. Komarichev, Z. Zhong, and J. Hua, "A-CNN: Annularly convolutional neural networks on point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7421–7430.
- [60] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "POINTCNN: Convolution on x-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 820–830.
- [61] W. Wu, Z. Qi, and L. Fuxin, "POINTConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9621–9630.
- [62] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 87–102.
- [63] F. Groh, P. Wieschollek, and H. Lensch, "Flex-convolution (million-scale point-cloud learning beyond grid-worlds)," 2018, *arXiv:1803.07289*.
- [64] M. Atzmon, H. Maron, and Y. Lipman, "Point convolutional neural networks by extension operators," 2018, *arXiv:1803.10091*.
- [65] A. Boulch, "ConvPoint: Continuous convolutions for point cloud processing," *Comput. Graph.*, vol. 88, pp. 24–34, 2020.
- [66] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 6411–6420.
- [67] A. Boulch, G. Puy, and R. Marlet, "FKAConv: Feature-kernel alignment for point cloud convolution," in *Proc. Asian Conf. Comput. Vis.*, 2020, pp. 381–399.
- [68] X. Chang *et al.*, "Shapenet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.
- [69] A. X. C. Dai, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5828–5839.
- [70] X. Li, L. Wang, M. Wang, C. Wen, and Y. Fang, "DANCE-NET: Density-aware convolution networks with context encoding for airborne LiDAR point cloud classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 166, pp. 128–139, 2020.
- [71] P.-H. Hsu and Z.-Y. Zhuang, "Incorporating handcrafted features into deep learning for point cloud classification," *Remote Sens.*, vol. 12, no. 22, 2020, Art. no. 3713.
- [72] F. Rottensteiner *et al.*, "The ISPRS benchmark on urban object classification and 3D building reconstruction," *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. 1, no. 1, pp. 293–298, 2012.
- [73] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervention*, 2015, pp. 234–241.
- [74] D. Tuia, E. Pasolli, and W. J. Emery, "Using active learning to adapt remote sensing image classifiers," *Remote Sens. Environ.*, vol. 115, no. 9, pp. 2232–2224, 2011.

Nan Li received the B.E. degree in surveying and mapping and the M.E. degree in cartography and geographic information engineering from Tongji University, Shanghai, China, 2010 and 2013, respectively. She is currently working toward the Ph.D. degree at the Photogrammetry Research Group, the Department of Geodesy and Geoinformation, Technische Universität Wien (TU Wien), Vienna, Austria.

Her work devoted to the classification of LiDAR point clouds by using machine learning methods.

Olaf Kähler received the Diploma in computer science from the Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany, in 2004, and the Ph.D. degree in computer science from the Friedrich-Schiller-Universität Jena, Jena, Germany, in 2009. Until May 2011, he has been a Research Associate with the Department of Engineering, the University of Cambridge and then joined the Active Vision Group, the University of Oxford, in June 2011, as a Postdoctoral Research Assistant. As of July 2016, he joined the Research Group Active Vision Technologies, Siemens Technology in Graz where he is currently a Senior Key Expert on large scale visual analytics. His research interests include a wide range of computer vision topics, structure-from-motion, 3-D reconstruction and geometry, but also semantic interpretation of 2-D and 3-D data.

Norbert Pfeifer was born in Vienna, Austria, in 1971. He received the Dipl. Ing. and Ph.D. degrees in surveying engineering from Technische Universität Wien (TU Wien), Vienna, in 1997 and 2002, respectively.

From 2003 to 2006, he was Research Assistant and Assistant Professor with TU Delft, Netherlands. In 2006, he was Lecturer with the Department of Geography, the University of Innsbruck, Innsbruck, Austria, and Senior Researcher with the Centre for Natural Hazard Management, alp-S, in Innsbruck. Later in 2006, he took the position of Professor with TU Wien in Photogrammetry. He has coauthored more than 100 articles in journals and six books. His research interests are LiDAR signal processing, calibration of airborne and terrestrial Laser Scanning, classification and segmentation of Lidar point clouds, 3-D modeling, and application of point clouds in the environmental sciences.

Prof. Pfeifer is member of the International Society of Photogrammetry and Remote Sensing.

Publication V

- Title: Active learning to extend training data for large area airborne LiDAR classification
- Authors: Li, N. and Pfeifer, N
- Published in: ISPRS Geospatial Week 2019, International Archives of the Photogrammetry, Remote Sensing & Spatial Information Science
DOI: 10.5194/isprs-archives-XLII-2-W13-1033-2019



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

ACTIVE LEARNING TO EXTEND TRAINING DATA FOR LARGE AREA AIRBORNE LIDAR CLASSIFICATION

Nan Li^{1,2,*}, Norbert Pfeifer¹

¹ Department of Geodesy and Geoinformation, Technische Universität Wien, 1040 Vienna, Austria - (nan.li, norbert.pfeifer)@geo.tuwien.ac.at

² College of Survey and Geoinformation, Tongji University, 200092 Shanghai, China - linan123@tongji.edu.cn

KEY WORDS: active learning, semi-supervised classification, training data selection

ABSTRACT:

Training dataset generation is a difficult and expensive task for LiDAR point classification, especially in the case of large area classification. We present a method to automatically extend a small set of training data by label propagation processing. The class labels could be correctly extended to their optimal neighbourhood, and the most informative points are selected and added into the training set. With the final extended training dataset, the overall (OA) classification could be increased by about 2%. We also show that this approach is stable regardless of the number of initial training points, and achieve better improvements especially stating with an extremely small initial training set.

1. INTRODUCTION

LiDAR (Light Detection And Ranging) automatic classification has been an important study topic over years. Supervised statistical approaches, such as Support Vector Machines (SVM) (Secord and Zakhor, 2007) or Random Forest (Guo et al., 2011) have been widely applied and achieved good performance. Additionally, to incorporate the spatial contextual information, Markov Random Field (MRF) and Conditional Random Field (CRF) are successfully used for contextual classification and achieve smoother results than the classifications based on individual independent features (Niemeyer et al., 2014; Shapovalov et al., 2010). This research mostly focuses on site-specific classification for 3D points at a small scale. Only few papers were published on large area LiDAR classification.

Extensive 3D point clouds over large area would result in handcrafted features inhomogeneity, making automated points cloud classification difficult. This would bring further challenges for class separability when only small training data is available. Especially, supervised classifiers rely on the quality of the labeled training data. The training samples should be fully representatives of the class-type statistics to allow the classifier to find the correct solution. In the case of large area classification, this constraint makes the generation of an appropriate training set a difficult and expensive task that requires extensive manual interaction. This is a common problem for classification of large amounts of data, and only a small amount of reference points can be manual labelled due to the limited economical and temporal resources. Therefore, the classification model constructed on the collected small training data could show poor generalization capabilities when applied to the rest of large amount of data. Additionally, manual training set definition is usually done by visual inspection of the scene and the successive labeling of each sample. This phase is highly redundant as well as time-consuming.

A solution to the problem of training data extraction is represented by semi-automatic active learning methods. Its key

idea is to select the samples whose inclusion in the training set would be beneficial to the classification performance. And the semi-automatic active learning already has shown to be effective for hyperspectral image classification. For instance, a combination of the SVM classifier is commonly used (Mitra et al., 2004; Tan et al., 2014), samples that are close to the hyper-plane are selected into the training dataset. In order to be adaptive with any generative classifier, the maximum information gain (Rajan et al., 2008) and breaking tie (Luo et al., 2005) can also be used to select uncertain samples. A co-training approach proposed by (Romaszewski et al., 2016) scored samples by combining spatial and spectral features, an optimal training set would be learned by iteratively adding new samples with high scores.

In this paper, we aim to extend a small set of initial labelled samples during a process of label propagation. By adapting an optimal neighborhood selection, the knowledge about class labels from the training set can be correctly extended to their neighborhood. And one most informative point is selected by BT (breaking tie) and added into the training set. In this way, we extend the training dataset, and automatically label the newly added samples. Compared with original small training set, the new extended training set could be more representative for features and capable to improve the classification results.

The rest of the paper is organized as follows: Section 2 explains our method. Section 3 presents the experiment on real data and its results, while Section 4 describes the performance along iteration and the impact of the number of initial training points. Summaries are provided in Section 5.

2. METHODOLOGY

Normally, the active learning approach consists of two components. The first is the selection of the most useful unlabelled samples to the classifier, and the second is how to determine the class labels of these new selected samples. In this paper, we start with a small set of suboptimal training points. The

* Corresponding author

breaking ties (BT) method (Luo et al., 2005) is applied to sample the informative unlabelled points. And the label of selected unlabelled point is determined by the spatial similarity. Since the class label is highly correlated with spatial similarity of points, we could assume that points located in the same neighbourhood are likely to have the same label with the center point. After adding those informative samples to the training dataset, the classification model is forced to focus on conflicting areas and to improve its generalization capabilities. The processing sequence is as follows:

- Step 1: Based on the initial small training dataset, an initial classifier is built;
- Step 2: For each training point, finding its' optimal neighbouring points;
- Step 3: The classifier is applied to those neighbouring points, and one most informative point is selected by the minimal BT value and labelled by the current training point.
- Step 4: Extending training dataset by adding new samples, and updating the classifier;
- Step 5: Repeating step 2,3,4, until a maximum iteration number is met. Then, the final training set is used to refine the classifier;
- Step 6: Finally, the classifier is used to predict labels for all unlabelled points.

The following section 2.1 describe the estimation of the optimal neighborhood, and section 2.2 induces the breaking ties

2.1 Label propagation by the optimal neighbourhood

By taking the advantage of spatial correlation of point cloud, the knowledge about class labels of training points can be extended to their neighborhood. To guarantee the accuracy of label propagation, an optimal neighborhood estimation method is applied (Li et al., 2019). The neighboring points are adaptively selected by weighted geometric similarity, so that all neighboring points that potentially belong to the same object with the concerned points could be included.

Here, the geometric similarity is measured by the angle between the normal vectors and point-to-plane orthogonal distances, while the weights are determined by the local surface variations (σ). To avoid lacking enough neighboring points for non-planar points, like vegetation, we assign larger weights to those non-planar points to increase the geometric similarity with neighbors. The weight function is defined in Eq. (1), and neighboring points that satisfy Eq. (2) are collected as the optimal neighbors of the concerned point:

$$Weight(p_0, p_i) = \begin{cases} 1 & \text{if } \sigma(p_0) \leq THR_\sigma \\ e^{\sigma(p_0)} \cdot e^{\sigma(p_i)} & \text{else } \sigma(p_0) > THR_\sigma \end{cases} \quad (1)$$

$$\begin{aligned} & Weight(p_0, p_i) \cdot nv_{p_0} \cdot nv_{p_i} \geq \cos(THR_\alpha) \\ & \text{and } Weight(p_0, p_i) \cdot |(p_0 - p_i) \cdot nv_{p_0}| \leq THR_d \end{aligned} \quad (2)$$

Where $\sigma(p_0)$ is the local surface variation in the point p_0 . THR_σ is a threshold to determine whether p_0 may belong to a planar object. nv_p denotes the normal vector of point p and $p = [x_p, y_p, z_p]$ denotes the 3D coordinates of point p . THR_α is the threshold of the normal vector-angle change and THR_d is the threshold of the local point-to-plane orthogonal distance.

2.2 Sample selection by BT

The BT technique is focused on the diversity of the unlabeled samples, which is obtained by the minimum difference between the two highest posterior class probabilities. The more a point

shows a similar posterior probability between the two most probable classes, the more it is uncertain and thus capable of providing useful information if added to the training dataset (Tuia et al., 2011). Thus the BT value of point p_i is formed by Eq.(3):

$$BT(p_i) = \max_{c \in C} (P(l_i = c | p_i)) - \max_{c \in C \setminus c^+} (P(l_i = c | p_i)) \quad (3)$$

Where $P(l_i = c | p_i)$ probability for class prediction l_i of a point p_i , $c \in C$ corresponds to one class c among the C possible classes, and $c^+ = \max_{c \in C} (P(l_i = c | p_i))$ is the most probable class for point p_i .

After finding all optimal neighboring points for one training point, the point minimizing Eq.(3) is then taken and labeled by the current, certain training point. The procedure is implemented for all training points and repeated for several times, the final selected labeled training points are used to refine the classifier.

3. RESULTS

3.1 Datasets

The point cloud we used was a fully labelled airborne LiDAR dataset of Vienna, Austria. The selected area is 1270×200 m², and the average density is about 50 points/m². This area represents a complex urban scene, including a mixture of high and low vegetation, high-rise and small detached houses, and flat and sloped ground. Five domain classes were categorized for the Vienna dataset: *ground, vegetation, roofs, façades and others* that include fences, cars, street lights, power lines and so on.

To get an impression of the dataset, the percentage distribution of each class in the dataset are shown in Table. 1.

Class	Percentage
Ground	53.70%
Vegetation	26.72%
Roofs	14.04%
Facades	1.54%
Others	4.00%

Table 1. Percentage distribution of each class in the dataset

3.2 Experiment setup and results

We used the random forest (RF) as the probabilistic classifier. For the optimal neighbourhood estimation, the spherical neighbourhood with radius of 2m was used for initial neighbouring points searching. Then the optimal neighbouring samples are selected by weighted geometric similarity and labelled by its neighbouring labelled training point. The initial number of training points is 100 per class, and the iteration was empirically set as 3 to extend the training data.

Figure. 1 shows the classification results using initial training dataset, extended training dataset and the reference dataset. From visual inspection, a more smooth classification result is achieved after training dataset extension. For instance, as shown in the marked area A in Figure. 2, more points are correctly classified as ground after the active learning, whereas those points are wrongly labelled as others in the initial training dataset. Another notable change appears in the marked area B in Figure. 2. There is a large amount of points misclassified as vegetation by the initial classification. After the active learning, most of those points' labels are changed into façades, this situation could be explained by a small error in the reference data (seen in the Figure. 2(c)).

From the Table. 2, the OA (overall) accuracy was increased by 1.7% after the active learning of training points. A relatively significant improvement was achieved for the class of vegetation and others, 4.00% and 3.37%, respectively. However, the accuracy of façade has dropped to 74.14% using the extended training dataset. 13.36% façade points are misclassified as vegetation which is 6.31% higher than the initial classification, and a few of façade points (2.41%) are misclassified as roofs since they are easily mixed up over the conjunctions of roofs and façades.

Class	Initial accuracy	Accuracy after active learning
OA	84.24%	85.98%
Ground	88.60%	89.35%
Vegetation	79.74%	83.73%
Roofs	81.82%	84.85%
Façades	84.59%	74.14%
Others	64.06%	67.37%

Table 2. Accuracy comparison of classification using initial training set and extended training set

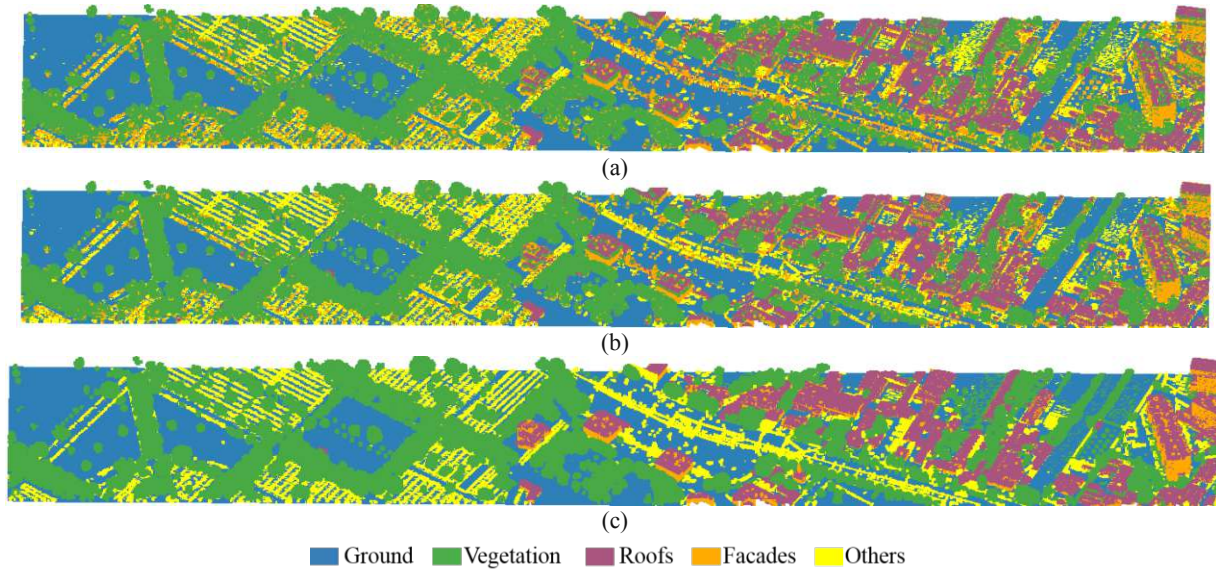


Figure 1. The classification results. (a) using the initial training dataset; (b) using the extended training dataset; (c) the reference labelled data.

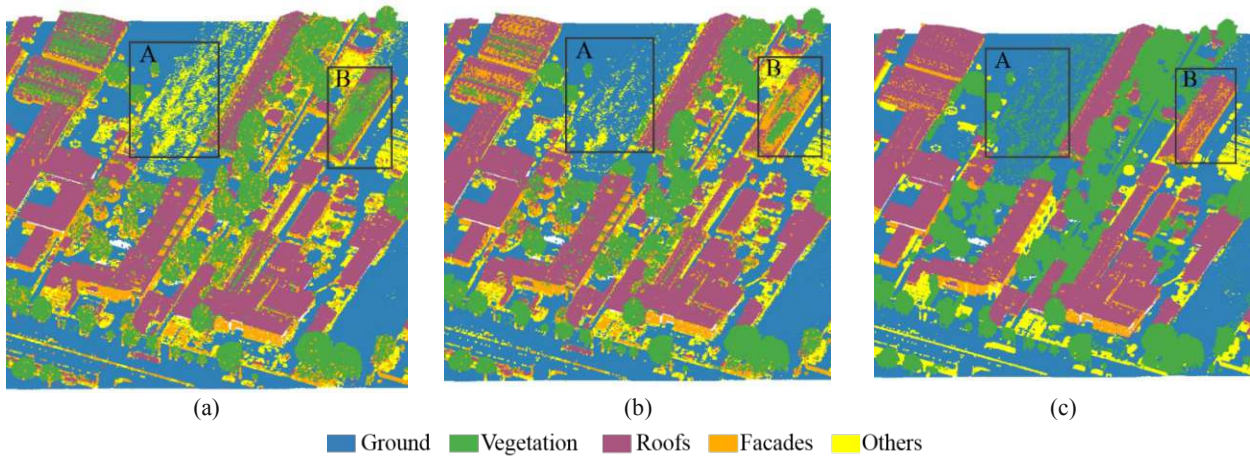


Figure 2. Detailed comparison of classification results. (a) using the initial training dataset; (b) using the extended training dataset; (c) the reference labelled data.

4. DISCUSSIONS

To access the stability of this active training data learning method, we started with different amounts of training dataset, which includes 10,100 and 1000 initial training points per class, respectively. Each experiment was repeated 3 times. The accuracy changes along the iterations are shown in Figure. 3, which are the average accuracy and its standard deviation over 3 experiments.

Compared to the initial classification results, the OA accuracies were all increased after the active learning (seen in Figure. 3(a)). Notably, the significant overall accuracy improvement was achieved by the smallest set of initial training data of 10 samples per class. It gained 5% higher OA accuracy than initial classification, while 2.4% and 1.5% OA increase for initial training points of 100 and 1000 per class, respectively. The representativeness of the extremely small training set is usually lacking strongly, thus the effect of adding new informative samples would be notable when it was started with a poor initial

classification result. While the models trained by 100 and 1000 samples per class are already decent, the improvement would become moderate when the amount of the initial training set is raised. Also due to the incompleteness of small initial training set of 10, the variation is relatively larger than the other two initial training sets.

We also observed that the accuracy would be immediately improved by extending the training data in the 1st iteration, and the accuracy only has slight changes over iterations besides the accuracy of façade. It means that the samples that are selected during the first extension are the most informative and could be effective to increase classification ability, whereas other samples from the rest of iterations may have very similar feature vectors with samples that already exist in the training set. Therefore, they could not provide more useful information to achieve better accuracy. This is caused by the local neighbourhood we used for label propagation. However, the trend of accuracy change of

façade is different from the others. Façade points tend to be misclassified into vegetation during this active learning procedure. Since generally the optimal neighbourhood favours points that are located in the same plane, vegetation points that lie in the same vertical plane would have similar feature vectors with the vertical façade points. Iteratively including those vegetation points into training dataset would lead to the confusion with façade.

Another interesting finding is that the performance of active learning would be fundamentally impacted by the initial amount of training samples. The accuracy with 100 initial training points per class reached 84.5% after 6 iterations, meanwhile the total number of training points per class is 1600. This result is still not as good as the initial classification accuracy using random 1000 training samples at first. But it is comparable to the classification by initial training samples of 300 per class (84.04%).

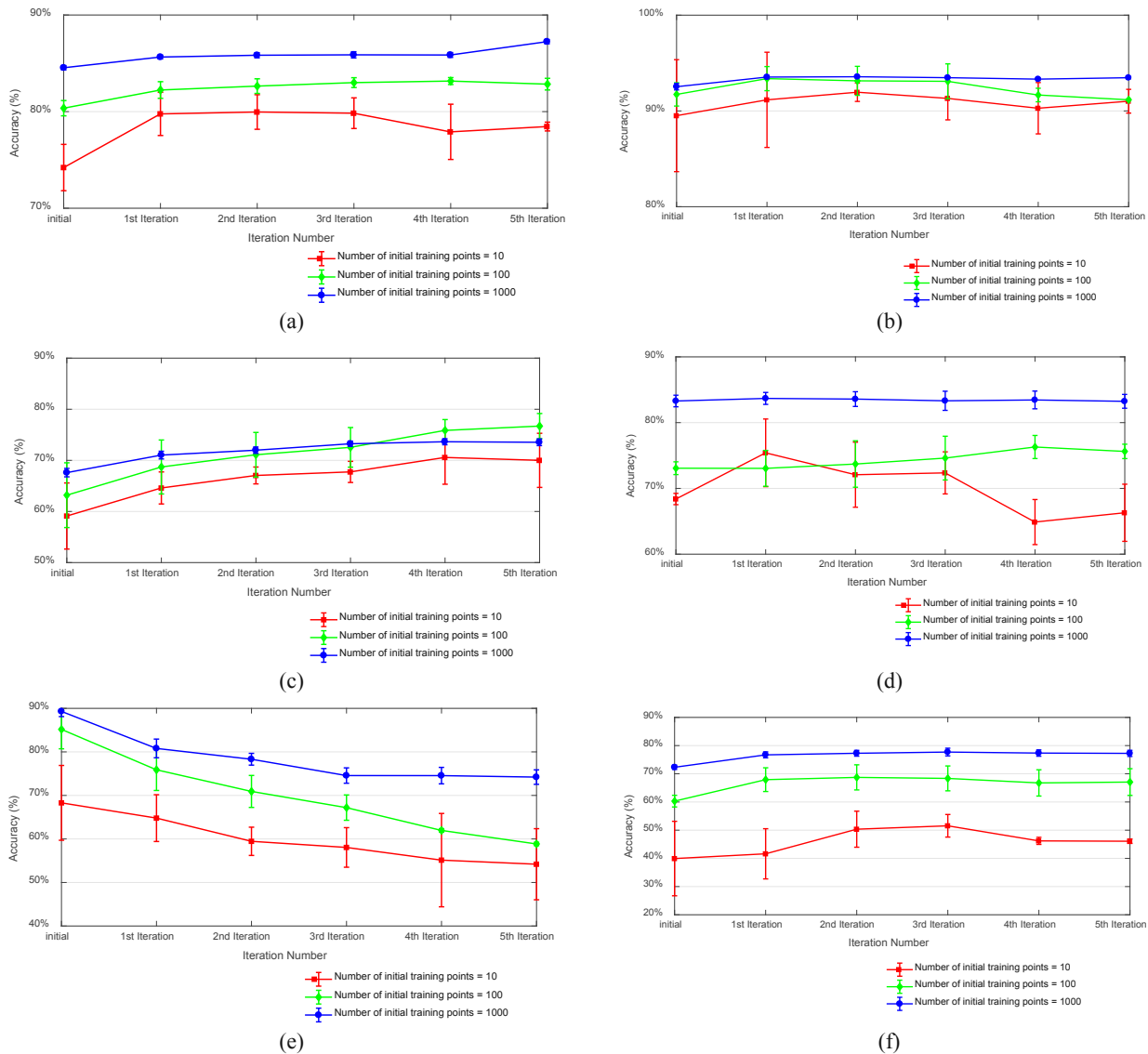


Figure 3. The trend of accuracy changes over iterations. (a) overall accuracy; (b) ground accuracy; (c) vegetation accuracy; (d) roofs accuracy; (e) façades accuracy; (f) others accuracy.

5. SUMMARY

We proposed an effective active learning method to automatically extend training points. Classification accuracy was increased by using the extended training dataset, which was significant especially starting with an extremely small set of 10 labelled points per class. An optimal training dataset would be achieved by a few of iterations. The reasonable amount of training samples also keep the classifier learning efficient. Due to the limitation of initial training sample, an exploration for initial samples selection will be considered in the further research.

ACKNOWLEDGEMENTS

The Vienna airborne LiDAR dataset was provided by the 'Stadtvermessung Wien', the Magistrate of the City of Vienna.

REFERENCES

- Guo, L., Chehata, N., Mallet, C., Boukir, S., 2011. Relevance of airborne lidar and multispectral image data for urban scene classification using Random Forests. *ISPRS J Photogramm Remote Sens* 66, 56-66.
- Li, N., Liu, C., Pfeifer, N., 2019. Improving LiDAR classification accuracy by contextual label smoothing in post-processing. *ISPRS J Photogramm Remote Sens* 148, 13-31.
- Luo, T., Kramer, K., Goldgof, D.B., Hall, L.O., Samson, S., Remsen, A., Hopkins, T., 2005. Active learning to recognize multiple types of plankton. *Journal of Machine Learning Research* 6, 589-613.
- Mitra, P., Shankar, B.U., Pal, S.K., 2004. Segmentation of multispectral remote sensing images using active support vector machines. *Pattern recognition letters* 25, 1067-1074.
- Niemeyer, J., Rottensteiner, F., Soergel, U., 2014. Contextual classification of lidar data and building object detection in urban areas. *ISPRS J Photogramm Remote Sens* 87, 152-165.
- Rajan, S., Ghosh, J., Crawford, M.M., 2008. An active learning approach to hyperspectral data classification. *IEEE Trans Geosci Remote Sens* 46, 1231-1242.
- Romaszewski, M., Głomb, P., Cholewa, M., 2016. Semi-supervised hyperspectral classification from a small number of training samples using a co-training approach. *ISPRS J Photogramm Remote Sens* 121, 60-76.
- Secord, J., Zakhor, A., 2007. Tree detection in urban regions using aerial lidar and image data. *IEEE Geosci. Remote Sens. Lett* 4, 196-200.
- Shapovalov, R., Velizhev, E., Barinova, O., 2010. Nonassociative markov networks for 3d point cloud classification. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXVIII, Part 3A*. Citeseer.
- Tan, K., Li, E., Du, Q., Du, P., 2014. An efficient semi-supervised classification approach for hyperspectral imagery. *ISPRS J Photogramm Remote Sens* 97, 36-45.
- Tuia, D., Pasolli, E., Emery, W.J., 2011. Using active learning to adapt remote sensing image classifiers. *Remote Sensing of Environment* 115, 2232-2242.



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

3, Classification performance comparison of the investigated methods

Different contextual point-wise classification methods have been developed or explored in the Publication I-IV. Therefore, the focus in this chapter is on a performance comparison of the classification methods investigated in the published papers, which include the tensor-based sparse representation classification (denoted by TSRC) proposed in the Publication I & II, the contextual label smoothing strategy (denoted by Label-Smoothing) proposed in the Publication III, and the deep learning networks (PointNet++, SparseCNN and KPConv) investigated in the Publication IV.

3.1 Test data

The comparison is conducted by using two datasets. The first one is the ALS dataset of Vienna, which has been organized into a number of 1270m×1020m tiles. The elaborate description about the ALS dataset of Vienna, such as the point density, can be found in the Publication IV. The selected tiles are the same as that used in the Publication IV, which contains 4 training tiles and 5 test tiles. The considered classes are *ground*, *vegetation*, *building* and *others* which includes common street objects such as street lights, cars and shrubs.

The second dataset is the ALS dataset of Vaihingen, which is a part of ISPRS 3D labeling benchmark. The used training and test point clouds are the same as described in the ISPRS 3D points benchmark. Six classes are considered in this task, namely *ground*, *cars*, *fences*, *roofs*, *façade* and *vegetation*. Compared to the ALS dataset of Vienna, the ALS dataset of Vaihingen has a smaller volume and lower point density.

3.2 Setup

Random Forest is used as a baseline here, which is also used as the initial classification input for Label Smoothing. The Random Forest classification results of the Vienna dataset and the Vaihingen dataset are collected from the Publication III and the Publication IV, respectively.

The experiment of TSRC is only conducted on the Vienna dataset. To build the classifier, 50 training samples per class are randomly selected from the 4 training tiles. The handcrafted features and geometric parameters employed by TSRC are described in the Publication II. However, considering the demanding inference time required by TSRC, only a small part (317.5 m × 255 m) of a tile of the Vienna dataset is used for evaluation.

For Label Smoothing, the same parameters setting is used on the two datasets (the Vienna and Vaihingen dataset), and the details can be found in the Publication III. As mentioned previously, its initial classification input is provided by the Random Forest classification result.

For the three deep learning methods, the classification results of the Vienna dataset are collected from the Publication IV, where the training and test setup is also presented. For the experiment of the Vaihingen dataset, PointNet++ and SparseCNN use the same setup, such as the data preparation and the configuration of hyper-parameters, as that used for the Vienna dataset, respectively. For KPConv, the same configuration of hyper-parameters is used, however, fewer points (25k) are used to generate the training and test patches to avoid Out of Memory (OOM) for the Vaihingen dataset, compared to that (50k) used for the Vienna dataset. Because the Vaihingen point clouds has a lower point density than the Vienna point clouds, the area of each patch generated from the Vaihingen dataset is accordingly larger than the Vienna dataset, when the patch contains the same amount of points. Due to the grid-subsampling procedure used in each layer of KPConv, the number of subsampled points is determined by the grid-size and the area of the input

patch. That is, with a fixed grid-size, the number of input instances becomes larger when the input patch covers a large area. Therefore, using the same amount of points to generate input patches for the Vaihingen dataset leads to OOM.

3.3 Results

3.3.1 Vienna dataset

Fig. 1 shows the classification results by the aforementioned methods. Tab. 1 and Tab. 2 present the classification accuracy in terms of recall and precision, respectively. TSRC leads to the least accurate classification result, but also is the most time-consuming method among all concerned classifiers. As a post-processing method, Label-Smoothing proves to be a very effective method for improving classification accuracy. It achieves a close classification performance to PointNet++ and SparseCNN, and even delivers a better result in the classification of *building*, compared to KPConv. SparseCNN achieves the best overall classification performance by a narrow margin.

Despite of the inferior performance of TSRC, the amount of training samples used by TSRC is much less than that by other approaches. The number of training samples used by each approach is listed in the following:

- TSRC: 50 training samples per class are selected, total 200 points are used for learning.
- Random Forest: 20000 training samples per class are selected, total 80000 points are used for learning.
- Label-Smoothing: requires no additional training samples and takes a probabilistic classification result as input.
- PointNet++, SparseCNN and KPConv: use four tiles for learning and the number of points is 447,208,684.

Tab. 1 Recall on the test region from the Vienna dataset (the best accuracy in each category is in red)

Methods	Ground (%)	Vegetation (%)	Buildings (%)	Others (%)	OA (%)
Random Forest	98.11	90.00	92.84	79.37	93.99
TSRC	97.36	90.13	83.31	64.50	90.20
Label Smoothing	98.96	91.20	98.15	80.53	96.43
PointNet++	99.17	94.78	95.78	72.69	96.17
SparseCNN	99.67	95.91	99.19	79.48	98.02
KPConv	99.79	96.13	80.10	79.06	91.61

Tab. 2 Precision on the test region from the Vienna dataset (the best accuracy in each category is in red)

Methods	Ground (%)	Vegetation (%)	Buildings (%)	Others (%)	OA (%)
Random Forest	98.89	91.29	97.50	48.62	93.99
TSRC	93.37	85.61	98.23	38.16	90.20
Label Smoothing	98.52	98.66	98.76	55.74	96.43
PointNet++	94.45	98.54	98.33	81.52	96.17
SparseCNN	98.05	98.50	98.85	85.98	98.02
KPConv	87.31	97.65	99.00	64.42	91.61

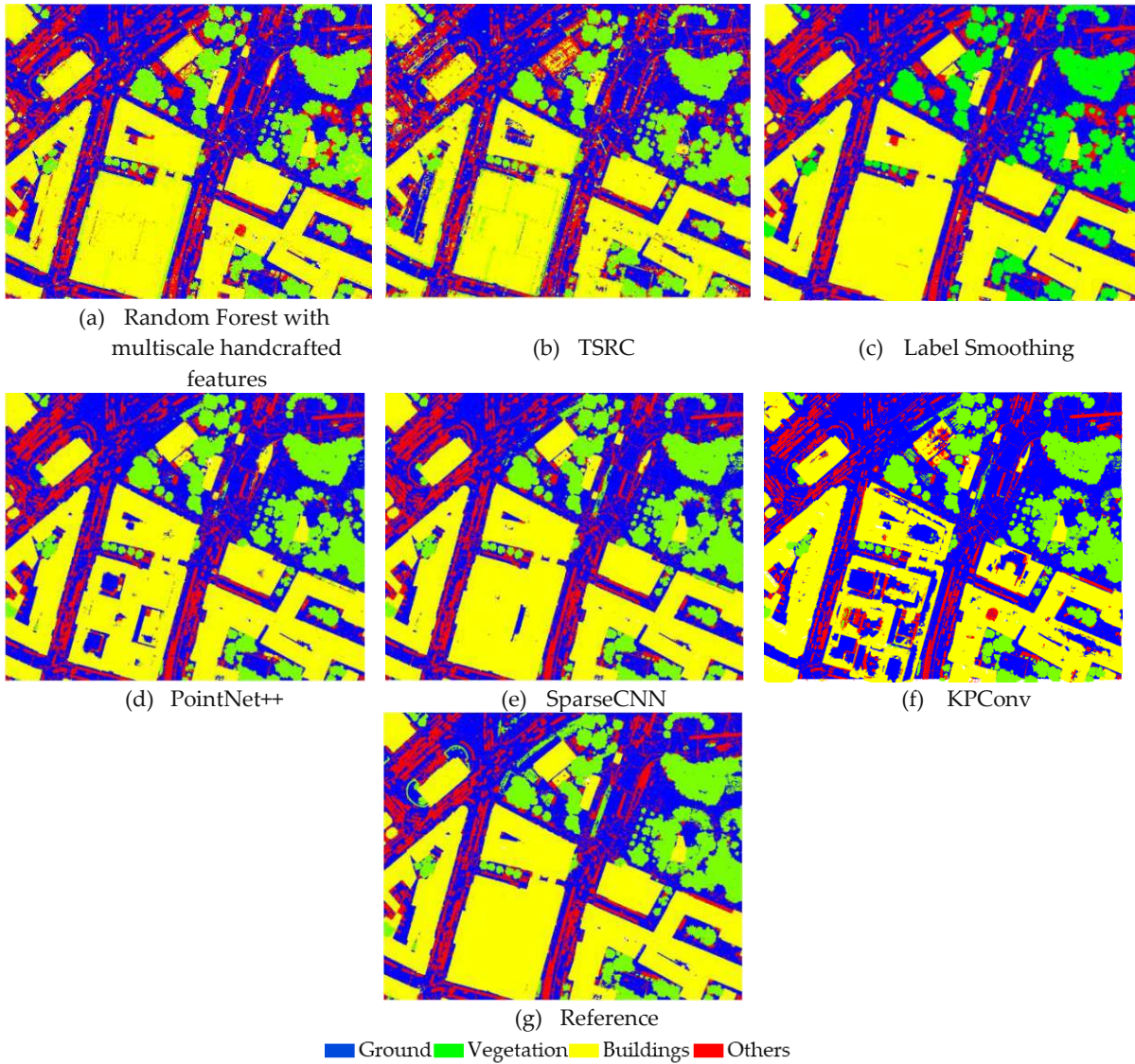


Fig. 1: Classification results of the test region.

Thanks to the small amount of training samples needed by TSRC, the learning of TSRC classifiers only costs a few minutes. However, it demands a long time for inference, because each point has to be represented as a four-dimensional tensor and be projected into the dictionary in order to compute the label-relevant sparse tensor. For the $317.5 \text{ m} \times 255 \text{ m}$ test region, TSRC has spent around 4 days to finish the classification. Thus, it's not very feasible to apply TSRC on a large-scale point cloud. Additionally, the inferior result by TSRC suggest that, the proposed high-dimensional tensor is not an optimal strategy to explore the local spatial distribution.

Thus, Label-Smoothing and the selected deep learning methods are further compared on a large-scale point cloud, which is an entire test tile that covers a $1270\text{m} \times 1020\text{m}$ area and locates in the center of Vienna. Tab. 3 and Tab. 4 show the recall and precision of the classification results, and Tab. 5 presents the computation time for each individual step of all considered methods. In accordance with the results shown in the previous small area, Label Smoothing shows the effectiveness in improving classification performance and achieves a comparable result with the three selected deep learning methods. The less accurate classification of *others* by PointNet++ indicates that, the class boundaries tend to be not well preserved for the small-size objects with a low point density, and another example is the misclassified meadow presented in the Publication IV. As for KPConv, it has a difficulty in delivering correct labels for large-size objects. As seen in Fig. 1(f), a large area of *building* is wrongly classified as *ground* by KPConv, due to the limited

convolution radius considered in KPConv. The best performance is again achieved by SparseCNN regarding to the OA and accuracy of individual class.

Regarding the running time, the inference by the deep learning models are faster than the Label Smoothing. Note that the implementations of the deep learning models and Label Smoothing require and use different hardware. Label Smoothing was implemented in Matlab and run on a computer equipped with an AMD Ryzen 7 2700X (3.7GHz) processor, 32 GB RAM and particularly no GPU was involved for the processing. While, another machine was used for deep learning experiments with AMD Ryzen Threadripper 1900X (3.5 GHz) processor, 512 GB RAM, and an Nvidia RTX 2080 Ti with 11 GB RAM, and most computation is done by GPU. However, one benefit of Label Smoothing is that it requires no additional reference data and training processing.

Tab. 3 Recall on the test tile from the Vienna dataset (the best accuracy in each category is in red)

Methods	Ground (%)	Vegetation (%)	Buildings (%)	Others (%)	OA (%)
Random Forest	98.07	88.69	91.75	82.35	93.19
Label Smoothing	99.27	89.75	96.66	81.86	95.99
PointNet++	98.86	95.92	97.39	73.41	96.88
SparseCNN	99.08	96.58	98.74	83.77	98.02
KPConv	99.68	96.74	86.58	82.13	92.70

Tab. 4 Precision on the test tile from the Vienna dataset (the best accuracy in each category is in red)

Methods	Ground (%)	Vegetation (%)	Buildings (%)	Others (%)	OA (%)
Random Forest	98.70	84.31	98.20	47.92	93.19
Label Smoothing	97.97	95.54	98.99	55.53	95.99
PointNet++	95.55	97.97	98.51	82.62	96.88
SparseCNN	98.39	98.30	98.62	84.32	98.02
KPConv	88.92	95.55	99.21	59.40	92.70

Tab. 5 Computation time on the test tile from the Vienna dataset

Methods	training	inference	merge
Label Smooth	0 h	18.0 h	0 h
PointNet++	158.4 h	5.5 h	1.9 h
SparseCNN	82.8 h	9 min	0 h
KPConv	88.1 h	1.2 h	2.5 h

3.3.2 Vaihingen dataset

Except TSRC, Label Smoothing and the selected deep learning networks are further applied on the second dataset, the ALS dataset of Vaihingen. The main aim of this experiment is to find out whether the selected deep learning methods can keep their advantageous when only limited training samples are available.

Tab. 6 and Tab. 7 shows the classification accuracy in terms of recall and precision, respectively. A part of the classification result is presented in Fig. 2. All the investigated methods achieve a better classification performance than Random Forest, regarding OA. Label Smoothing demonstrates its effectiveness in avoiding over-smoothing of minority classes, such as *cars* and *fences*, which, however, are largely misclassified by the three deep learning methods, especially by PointNet++ and SparseCNN.

Compared to their performance on the Vienna ALS point clouds, all the three deep learning networks deliver a less accurate classification result on the Vaihingen dataset. Whereas, the classification improvement by Label Smoothing is more significant than that obtained on the Vienna dataset. In addition, PointNet++ shows a more serious defect in correctly identifying small-size objects on the Vaihingen dataset, compared to the Vienna dataset. An example is illustrated in Fig. 2, the *cars* are barely detected by PointNet++. Like the result on the Vienna dataset, KPConv

tends to misclassify points of *roofs* as *ground*, because of the small convolution radius used in the architecture. In both datasets, SparseCNN outperforms the other method in general. However, the accuracy of the minority classes, such as *cars*, *fences* and *façade*, is unsatisfactory, which is even worse than that obtained by Random Forest.

For the three deep learning methods, even though a weighted loss function is used in the training to emphasis minority classes, the classification performance in the minority classes (*cars* and *fences*) of the Vaihingen data is much inferior to that (classification of *others*) in the Vienna dataset. This may suggest the importance of sufficient training data and that the impact of adding weights in loss functions is limited.

We published the part of the Vienna dataset used in the Publication IV in Zenodo (<http://doi.org/10.5281/zenodo.4777087>), which can serve as the training dataset for further studies.

Tab. 6: Recall of the Vaihingen dataset from ISPRS benchmark (the best accuracy in each category is in red)

Methods	Ground (%)	Cars (%)	Fences (%)	Roofs (%)	Façade (%)	Vegetation (%)	OA (%)
Random Forest	86.14	64.43	52.99	82.30	52.59	70.27	79.32
Label Smooth	92.57	63.51	46.01	93.23	56.80	82.42	86.52
PointNet++	93.49	13.46	2.24	86.98	48.46	89.71	87.44
SparseCNN	92.08	36.46	28.67	93.82	64.40	88.42	89.43
KPConv	92.40	44.42	18.70	85.96	61.82	87.95	87.24

Tab. 7 Precision of the Vaihingen dataset from ISPRS benchmark (the best accuracy in each category is in red)

Methods	Ground (%)	Cars (%)	Fences (%)	Buildings (%)	Façade (%)	Vegetation (%)	OA (%)
Random Forest	97.13	15.58	42.58	88.36	39.50	62.29	79.32
Labelsmooth	95.35	37.68	50.87	93.34	53.03	74.55	86.52
PointNet++	94.08	83.72	61.03	95.00	64.67	69.12	87.44
SparseCNN	96.85	51.60	31.68	95.10	58.86	76.69	89.43
KPConv	95.64	63.74	36.84	91.96	57.54	71.75	87.24

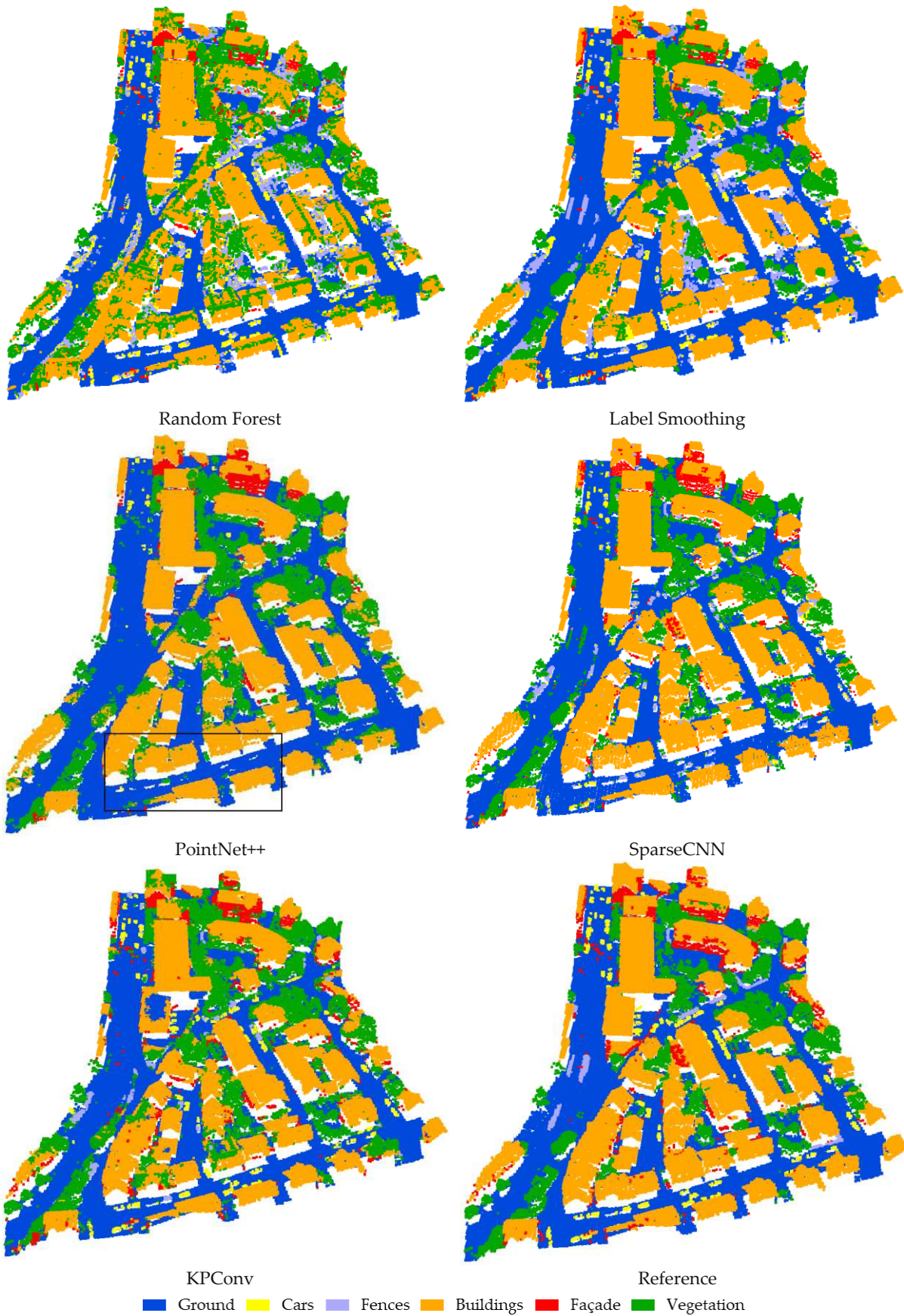


Fig. 2 Classification results of the Vaihingen dataset

3.4 Discussion

Based on the aforementioned experiments, a brief discussion of investigated methods is presented in the following.

3.4.1 TSRC

TSRC attempts to consider the spatial distribution and the handcrafted features of neighboring points simultaneously by the high-dimensional tensor representation. However, it turns out less effective than a Random Forest model that uses multiscale handcrafted features as inputs for the purpose of classification considering context.

Nevertheless, numerous studies in the field of remote sensing regard hyperspectral images as 3D tensors to jointly consider spatial and spectral features for dimensionality reduction, by which a classification improvement is achieved (Renard and Bourennane, 2009). The regular grids of images provide a natural structure for the 3D tensor representation, which also lies perfectly in the mathematic framework of 3D tensor data.

For irregular 3D points, it's especially implicit to define a high-dimensional tensor to represent their spatial arrangement in 3D space. The descriptor used in TSRC contains comprehensive information of spatial distribution and handcrafted features of points within a neighborhood, but also leads to high-degree complexity. High-degree of complexity usually increase the capability of models, a typical example is a deep learning architecture which usually comprises a large number of learnable parameters (could be millions). Correspondingly, a large amount of training data is required to learn such a great deal of parameters. Moreover, the achievements of deep learning methods are also attributed to the developed hardware, GPU, since it enables finishing the computation in a reasonable time.

As for TSRC, its less effective classification results may suggest that the sparse projection over the dictionary generated from only a few of training samples has limited capability for learning representative features from the highly complex tensors. Additionally, it is infeasible for TSRC to select a large amount of samples to reconstruct the spare representation, as the sparse coding is an iterative processing and thus is very time-consuming. Another fundamental argument for the unsatisfactory results could be whether the high-dimensional tensor is a convincing descriptor to include the local context.

3.4.2 Label Smoothing

Label Smoothing has proven to be an effective post-processing strategy to refine initial labels. Label Smoothing starts from a prior and probabilistic classification result, which can be both a strength and a weakness at the same time. The major advantage is that only little knowledge for constructing neighborhood is required, no further effort of reference data needs to be setup. This enables the Label Smoothing to be easily performed on large-scale point clouds and be independent of classes. Additional prior information about the spatial distribution pattern of classes can facilitate the processing of the Label Smoothing. However, the context information is computed by a well-defined neighborhood and initial classification probabilities. As a result, the classification improvement is limited by the initial classification results. If a poor initial classification results with high confidence is produced on a large region, wrong labels may not be corrected by the Label Smoothing.

3.4.3 PointNet++, KPConv and SparseCNN

The experiments have confirmed the outstanding capability of PointNet++, KPConv and SparseCNN for classifying point clouds. In comparison to MLP based networks, CNN based

networks are theoretically considered advantageous, as they can take the spatial arrangements of 3D points inherently into account, whereas the spatial information is ignored in MLP based methods. The results by the three networks are also in accordance with this statement. SparseCNN that employs a voxel-wise CNN achieves the best classification performance in both dataset, KPConv that conducts a point-wise convolution delivers a slightly better classification accuracy than PointNet++, especially in separating small-size objects. Additionally, some empirical knowledge is gained through the experiments:

For PointNet++, the geometric parameters such as neighborhood radius and number of neighbors need to be carefully selected and vary in different scenarios. For instance, the geometric parameters for indoor point clouds are set very differently from that for outdoor point clouds.

KPConv shows a great potential for implementing point-wise convolution. However, it suffers from the demanding memory consumption, which leads to OOM problem when a large convolution radius is used for dense point clouds. Thus, the number of points in each input patch needs to be adjusted according to the point density, in order to avoid OOM.

Regarding SparseCNN, the voxels provide a natural way to apply convolution on 3D point clouds. The most issues of PointNet++ and KPConv, such as the dependency of geometric parameters in PointNet++ and the memory usage in KPConv, have been solve by using SparseCNN. As a result, it provides the best classification accuracy, while requiring less running time and memory. The resolution loss caused by voxelization can be negligible compared to the gains. However, the satisfactory classification performance relies on a large amount of available training data.

4, Conclusions

Context is an important information to improve classification performance. SparseCNN is a particular effective model to incorporate context, meanwhile it is efficient in terms of running time and memory consumption. KPConv has a slightly better classification performance than PointNet++, especially on small-size objects. However, KPConv suffers from a demanding memory usage, so only limited neighboring points can be included in the convolution of KPConv. The lack of long-range context interactions leads to an unsatisfactory classification on large-size objects. The findings about the deep learning networks are also in line with the statement, that CNNs are superior to MLPs for semantic classification tasks, as CNNs can exploit the spatial arrangement inherently, which is ignored by MLPs.

An alternative strategy is the Label Smoothing. Although it is a two-step processing and less efficiency compared with an end-to-end classification scheme, it achieves satisfactory and even comparable classification results with PointNet++ and KPConv. Unlike the deep learning models, it requires no large volumes of training data and is independent of varying point densities in diverse scenarios. Label Smoothing proves to be an advantageous and feasible strategy to refine initial labels at large scales. Moreover, deep learning methods managed to learn effective task-specific context features directly from raw input data, but researchers often criticize their multilayer non-linear structures for being “black box” and being not traceable. In contrast, in Label Smoothing, the neighborhood and the compatibility coefficients that are used to derive context information have clear physical meanings, which are much easier to setup compared with the hyper-parameters in deep learning methods.

Context as discussed in this dissertation only refers to the surrounding characteristics in terms of data itself. This assists classification to a limited extent. Context cannot explain cases where the elements are classified by their functions rather than by their appearances, such as the land use mapping. The functions can be hardly deduced by their surrounding context. Additionally, the attributes that are collected by sensors determine what type of context information can be learned from point clouds, for examples, road surface markings, can not be identified without additional color or spectral attributes.

A future perspective relates to investigation of training data. It's no doubt that the quality of reference labels of training data is essential to achieve a well-behaved model. However, a few of erroneous labels are inevitably produced by manual or semi-supervised classification. We found that the deep learning methods as well as Random Forest are capable to handle the erroneous labels to some extent. In that sense, it is more important for evaluation data to have a highly accurate labelling, compared to the training data, in order to provide a reliable accuracy assessment. Correspondingly, a more robust methodology for accuracy assessment may become a further topic, for example, using the confidence of parameters estimation during the training rather than evaluating the accuracy after the classification. In addition, since the dependency of large amounts of training data, it is well worth spending efforts on actively learning training data.



Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Bibliography

Lee, I. and Schenk, T., Perceptual organization of 3d surface points. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 2002, 34(3/A) , pp. 193-198.

Filin, S. and Pfeifer, N., Neighborhood systems for airborne laser data. *Photogrammetric Engineering & Remote Sensing*, 2005, vol. 71(6) , pp. 743-755.

Linsen, L. and Prautzsch, H., Local versus global triangulations. *Proceedings of EUROGRAPHICS*. 2001.

Weinmann, M., Jutzi, B., Hinz, S. and Mallet, C., Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2015, vol. 105, pp. 286-304.

Demantke, J., Mallet, C., David, N. and Vallet, B., Dimensionality based scale selection in 3d lidar point clouds. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 2011, 38(Part 5): W12.

Guo, L., Chehata, N., Mallet, C. and Boukir, S., Relevance of airborne lidar and multispectral image data for urban scene classification using random forests. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2011, vol. 66(1) , pp. 56-66.

Mallet, C., Bretar, F., Roux, M., Soergel, U. and Heipke, C., Relevance assessment of full-waveform lidar data for urban area classification. *ISPRS journal of photogrammetry and remote sensing*, 2011, vol. 66(6) , pp. S71-S84.

Khoshelham, K., Elberink, S. O. and Xu, S., Segment-based classification of damaged building roofs in aerial laser scanning data. *IEEE geoscience and remote sensing letters*, 2013, vol. 10(5) , pp. 1258-1262.

Myint, S. W., Lam, N. S. N. and Tyler, J. M., Wavelets for urban spatial feature discrimination. *Photogrammetric Engineering & Remote Sensing*, 2004, vol. 70(7) , pp. 803-812.

Pacifici, F., Chini, M. and Emery, W. J., A neural network approach using multi-scale textural metrics from very high-resolution panchromatic imagery for urban land-use classification. *Remote Sensing of Environment*, 2009, vol. 113(6) , pp. 1276-1292.

Pesaresi, M. and Benediktsson, J. A., A new approach for the morphological segmentation of high-resolution satellite imagery. *IEEE transactions on Geoscience and Remote Sensing*, 2001, vol. 39(2) , pp. 309-320.

Zhou, W., Huang, G., Troy, A. and Cadenasso, M. L., Object-based land cover classification of shaded areas in high spatial resolution imagery of urban areas: A comparison study. *Remote Sensing of Environment*, 2009, vol. 113(8) , pp. 1769-1777.

Vargas, J. E., Falcão, A. X., dos Santos, J. A., Esquerdo, J. C. D. M., Coutinho, A. C. and Antunes, J. F. G., Contextual superpixel description for remote sensing image classification. *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2015, pp. 1132-1135.

Li, J., Bioucas-Dias, J. M. and Plaza, A., Spectral-spatial hyperspectral image segmentation using subspace multinomial logistic regression and markov random fields. *IEEE Transactions on Geoscience and Remote Sensing*, 2011, vol. 50(3) , pp. 809-823.

Zhao, W., Du, S., Wang, Q. and Emery, W. J., Contextually guided very-high-resolution imagery classification with semantic segments. *ISPRS journal of photogrammetry and remote sensing*, 2017, vol. 132, pp. 48-60.

Rodríguez-Cuenca, B., Malpica, J. A. and Alonso, M. C., A spatial contextual postclassification method for preserving linear objects in multispectral imagery. *IEEE transactions on geoscience and remote sensing*, 2012, vol. 51(1) , pp. 174-183.

Ma, L., Liu, Y., Zhang, X., Ye, Y., Yin, G. and Johnson, B. A., Deep learning in remote sensing applications: A meta-analysis and review. *ISPRS journal of photogrammetry and remote sensing*, 2019, vol. 152, pp. 166-177.

Parikh, H., Patel, S. and Patel, V., Classification of sar and polsar images using deep learning: A review. *International Journal of Image and Data Fusion*, 2020, vol. 11(1) , pp. 1-32.

Simonyan, K. and Zisserman, A., Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations*. 2015.

He, K., Zhang, X., Ren, S. and Sun, J., Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770-778.

Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K. Q., Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261-2269.

Sherrah, J., Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. *arXiv preprint arXiv:1606.02585*, 2016.

Liu, Y., Fan, B., Wang, L., Bai, J., Xiang, S. and Pan, C., Semantic labeling in very high resolution images via a self-cascaded convolutional neural network. *ISPRS journal of photogrammetry and remote sensing*, 2018, vol. 145, pp. 78-95.

Zhang, J., Lin, S., Ding, L. and Bruzzone, L., Multi-scale context aggregation for semantic segmentation of remote sensing images. *Remote Sensing*, 2020, vol. 12(4) , pp. 701.

Maggiori, E., Tarabalka, Y., Charpiat, G. and Alliez, P., High-resolution aerial image labeling with convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 2017, vol. 55(12), pp. 7092-7103.

Marmanis, D., Wegner, J. D., Galliani, S., Schindler, K., Datcu, M. and Stilla, U., Semantic segmentation of aerial images with an ensemble of cnns. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2016, vol. 3, pp. 473-480.

Wang, H., Wang, Y., Zhang, Q., Xiang, S. and Pan, C., Gated convolutional neural network for semantic segmentation in high-resolution images. *Remote Sensing*, 2017, vol. 9(5), pp. 446.

Han, W., Feng, R., Wang, L. and Cheng, Y., A semi-supervised generative framework with deep learning features for high-resolution remote sensing image scene classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2018, vol. 145, pp. 23-43.

Hong, D., Yokoya, N., Xia, G.-S., Chanussot, J. and Zhu, X. X., X-modalnet: A semi-supervised deep cross-modal network for classification of remote sensing data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2020, vol. 167, pp. 12-23.

Yang, B., Dong, Z., Liu, Y., Liang, F. and Wang, Y., Computing multiple aggregation levels and contextual features for road facilities recognition using mobile laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2017, vol. 126, pp. 180-194.

Vosselman, G., Coenen, M. and Rottensteiner, F., Contextual segment-based classification of airborne laser scanner data. *ISPRS journal of photogrammetry and remote sensing*, 2017, vol. 128, pp. 354-371.

Zhang, J., Lin, X. and Ning, X., Svm-based classification of segmented airborne lidar point clouds in urban areas. *Remote Sensing*, 2013, vol. 5(8), pp. 3749-3775.

Weinmann, M., Schmidt, A., Mallet, C., Hinz, S., Rottensteiner, F. and Jutzi, B., Contextual classification of point cloud data by exploiting individual 3d neighbourhoods. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences II-3 (2015)*, Nr. W4, 2015, 2(W4), pp. 271-278.

Wei, L., Yu, M., Liang, Y., Yuan, Z., Huang, C., Li, R. and Yu, Y., Precise crop classification using spectral-spatial-location fusion based on conditional random fields for uav-borne hyperspectral remote sensing imagery. *Remote Sensing*, 2019, vol. 11(17), pp. 2011.

Niemeyer, J., Rottensteiner, F. and Soergel, U., Contextual classification of lidar data and building object detection in urban areas. *ISPRS journal of photogrammetry and remote sensing*, 2014, 87: 152-165.

Landrieu, L., Raguet, H., Vallet, B., Mallet, C. and Weinmann, M., A structured regularization framework for spatially smoothing semantic labelings of 3d point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2017, vol. 132, pp. 102-118.

Luo, C. and Sohn, G., Scene-layout compatible conditional random field for classifying terrestrial laser point clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2014, vol. 2(3), pp. 79-86.

Qi, C. R., Su, H., Mo, K. and Guibas, L. J., Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652-660.

Qi, C. R., Yi, L., Su, H. and Guibas, L. J., Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 2017, pp. 5099-5108.

Jiang, M., Wu, Y., Zhao, T., Zhao, Z. and Lu, C., Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652*, 2018.

Zhao, H., Jiang, L., Fu, C.-W. and Jia, J., Pointweb: Enhancing local neighborhood features for point cloud processing. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5565-5573.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M. and Solomon, J. M., Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019, vol. 38(5), pp. 1-12.

Maturana, D. and Scherer, S., Voxnet: A 3d convolutional neural network for real-time object recognition. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 922-928.

Graham, B., Engelcke, M. and van der Maaten, L., 3d semantic segmentation with submanifold sparse convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9224-9232.

Riegler, G., Osman Ulusoy, A. and Geiger, A., Octnet: Learning deep 3d representations at high resolutions, 2017, pp. 3577-3586.

Klokov, R. and Lempitsky, V., Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 863-872.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X. and Chen, B., Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 2018, pp. 820-830.

Wu, W., Qi, Z. and Fuxin, L., Pointconv: Deep convolutional networks on 3d point clouds. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 9621-9630.

Xu, Y., Fan, T., Xu, M., Zeng, L. and Qiao, Y., Spidercnn: Deep learning on point sets with parameterized convolutional filters. Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 87-102.

Groh, F., Wieschollek, P. and Lensch, H., Flex-convolution (million-scale point-cloud learning beyond grid-worlds). arXiv preprint arXiv:1803.07289, 2018.

Atzmon, M., Maron, H. and Lipman, Y., Point convolutional neural networks by extension operators. arXiv preprint arXiv:1803.10091, 2018.

Boulch, A., Convpoint: Continuous convolutions for point cloud processing. Computers & Graphics, 2020, vol. 88, pp. 24-34.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F. and Guibas, L. J., Kpconv: Flexible and deformable convolution for point clouds. Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 6411-6420.

Schmohl, S. and Sörgel, U., Submanifold sparse convolutional networks for semantic segmentation of large-scale als point clouds. ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences, 2019, vol. 4, pp. 77-84.

Winiwarter, L., Mandlbürger, G., Schmohl, S. and Pfeifer, N., Classification of als point clouds using end-to-end deep learning. PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science, 2019, vol. 87(3), pp. 75-90.

Li, W., Wang, F.-D. and Xia, G.-S., A geometry-attentional network for als point cloud classification. ISPRS Journal of Photogrammetry and Remote Sensing, 2020, vol. 164, pp. 26-40.

Wen, C., Yang, L., Li, X., Peng, L. and Chi, T., Directionally constrained fully convolutional neural network for airborne lidar point cloud classification. ISPRS Journal of Photogrammetry and Remote Sensing, 2020, vol. 162, pp. 50-62.

Li, X., Wang, L., Wang, M., Wen, C. and Fang, Y., Dance-net: Density-aware convolution networks with context encoding for airborne lidar point cloud classification. ISPRS Journal of Photogrammetry and Remote Sensing, 2020, vol. 166, pp. 128-139.

Jochem, A., Höfle, B., Rutzinger, M. and Pfeifer, N., Automatic roof plane detection and analysis in airborne lidar point clouds for solar potential assessment. Sensors, 2009, vol. 9(7), pp. 5241-5262.

Tooke, T. R., van der Laan, M. and Coops, N. C., Mapping demand for residential building thermal energy services using airborne lidar. Applied Energy, 2014, vol. 127, pp. 125-134.

Yu, B., Liu, H., Wu, J., Hu, Y. and Zhang, L., Automated derivation of urban building density information using airborne lidar data and object-based method. Landscape and Urban Planning, 2010, vol. 98(3-4), pp. 210-219.

Hecht, R., Meinel, G. and Buchroithner, M. F., Estimation of urban green volume based on single-pulse lidar data. IEEE Transactions on Geoscience and Remote Sensing, 2008, vol. 46(11), pp. 3832-3840.

Alonzo, M., Bookhagen, B., McFadden, J. P., Sun, A. and Roberts, D. A., Mapping urban forest leaf area index with airborne lidar using penetration metrics and allometry. Remote Sensing of Environment, 2015, vol. 162, pp. 141-153.

Plowright, A. A., Coops, N. C., Eskelson, B. N. I., Sheppard, S. R. J. and Aven, N. W., Assessing urban tree condition using airborne light detection and ranging. *Urban Forestry & Urban Greening*, 2016, vol. 19, pp. 140-150.

McLaughlin, R. A., Extracting transmission lines from airborne lidar data. *IEEE Geoscience and Remote Sensing Letters*, 2006, vol. 3(2), pp. 222-226.

Sohn, G., Jwa, Y. and Kim, H. B., Automatic powerline scene classification and reconstruction using airborne lidar data. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci*, 2012, vol. 1-3, pp. 167-172.

Mills, S. J., Castro, M. P. G., Li, Z., Cai, J., Hayward, R., Mejias, L. and Walker, R. A., Evaluation of aerial remote sensing techniques for vegetation management in power-line corridors. *IEEE Transactions on Geoscience and Remote Sensing*, 2010, vol. 48(9), pp. 3379-3390

Prieto, I., Izkara, J. L. and Usobiaga, E., The application of lidar data for the solar potential analysis based on urban 3d model. *Remote Sensing*, 2019, vol. 11(20), pp. 2348.

Yamagata, Y., Murakami, D., Yoshida, T., Seya, H. and Kuroda, S., Value of urban views in a bay city: Hedonic analysis with the spatial multilevel additive regression (smar) model. *Landscape and Urban Planning*, 2016, vol. 151, pp. 89-102.

Bernat Gazibara, S., Krkač, M. and Mihalić Arbanas, S., Landslide inventory mapping using lidar data in the city of zagreb (croatia). *Journal of Maps*, 2019, vol. 15(2), pp. 773-779.

Hung, C.-L. J., James, L. A. and Hodgson, M. E., An automated algorithm for mapping building impervious areas from airborne lidar point-cloud data for flood hydrology. *GIScience & Remote Sensing*, 2018, vol. 55(6), pp. 793-816.

Peng, F., Wong, M. S., Wan, Y. and Nichol, J. E., Modeling of urban wind ventilation using high resolution airborne lidar data. *Computers, Environment and Urban Systems*, 2017, vol. 64, pp. 81-90.

Yao, W., Hinz, S. and Stilla, U., Extraction and motion estimation of vehicles in single-pass airborne lidar data towards urban traffic analysis. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2011, vol. 66(3), pp. 260-271.

Renard, N. and Bourennane, S., Dimensionality reduction based on tensor modeling for classification methods. *IEEE Transactions on Geoscience and Remote Sensing*, 2009, vol. 47(4), pp. 1123-1131.