VISUALIZATION TECHNIQUES IN CLASSIFICATION OF 3D URBAN LIDAR POINT CLOUD

Beena Kumari

Master of Science by Research Thesis

January 2016

International Institute of Information Technology, Bangalore

VISUALIZATION TECHNIQUES IN CLASSIFICATION OF 3D URBAN LIDAR POINT CLOUD

Submitted to International Institute of Information Technology,
Bangalore
in Partial Fulfillment of
the Requirements for the Award of
Master of Science by Research

by

Beena Kumari
MS2013003

International Institute of Information Technology, Bangalore
January 2016

Thesis Certificate

This is to certify that the thesis titled **Visualization Techniques in Classification of 3D Urban LiDAR Point Cloud** submitted to the International Institute of Information Technology, Bangalore, for the award of the degree of **Master of Science by Research** is a bona fide record of the research work done by **Beena Kumari**, **MS2013003**, under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

_____

Prof. Jaya Sreevalsan Nair

Bangalore,

The 11th of January, 2016.

VISUALIZATION TECHNIQUES IN CLASSIFICATION OF 3D URBAN LIDAR POINT CLOUD

Abstract

Several supervised and unsupervised techniques exist in literature and in practice, for semantic labeling of objects in 3D urban LiDAR point cloud. However, the problem with these approaches is that they predominantly require a domain expert to either generate training data or evaluate the output. **We** approach this problem by using an interactive unsupervised method for semantic classification. We propose to preserve structural class labels with semantic class labels to get geometric- and contextual- aware semantic labeling. Such an enhanced classification will enable extraction of building footprints and road boundaries. We also propose a visual analytic framework to aid both structural and semantic classifications. The motivation for our framework is to reduce the dependency of users on domain experts for semantic classification, and provide flexibility to a user for exploration and analysis of data.

For structural classification, we have used structure tensor-based feature detection method. Across tensor-based methods, different types of tensors such as structure, voting and anisotropic diffusion tensor can be used to encode the local geometry of a point. We have extended anisotropic diffusion tensor-based method used for triangle meshes to unstructured point cloud. While determining the substitutability of the new tensors to the structure tensor, we have found that no well-defined metric exists for their comparison. We propose to use the shape of local neighborhood and saliency value as metrics for comparison of feature detection methods. We have used visualization techniques to study these two metrics, and found that the anisotropic diffusion tensor, generated using our method, encodes local geometry of point better than existing geometrical tensors.

Preserving the two classifications in the labeling of the points gives a geometry-aware contextual semantic labeling. We demonstrate results of our semantic classification on benchmark airborne LiDAR data-set of Vaihingen site provided by ISPRS, which are unstructured, and initially unlabeled. The overall accuracy of our algorithm is 78.2%. Our tool can be useful for finding building foot-prints and road boundaries. Our tool can also be useful for generating training data for supervised classifier.

Acknowledgements

My sincere gratitude to my advisor Prof. Jaya Sreevalsan Nair for her invaluable advice, continuous encouragement and support. It was my pleasure to work under her guidance. I am grateful to Prof. Ingrid Hotz, Professor, Linköping University, for her helpful comments on my research work. I would also like to thank Prof. G. Srinivasaraghavan for sharing his expertise on machine learning and providing helpful comments on my research work. I would like to thank Prof. S. Rajagopalan for sharing his expertise on GIS (geographic information system) and providing helpful comments for evaluation of my algorithm. I would like to thank Prof. T. K. Srikanth for sharing his expertise on visualization and providing feedbacks on my research work.

My sincere gratitude to Annu Singh and Sumant Kulkarni for their insightful comments and suggestions on my thesis. I am grateful to Raghavendra GS, Raksha PS, Shilpi Banerjee and Vishnu Priya for helping me in editing my thesis. I am thankful to Dr. Kiruba Bhagirathi for her kind support and encouragement throughout of my work.

I would like to thank my family and friends for their invaluable moral support and encouragement which helped me to stay focused on my study. Finally, I thank the Department of Science and Technology, Government of India and EMC^2-RSA, for the financial support throughout my Masters program.

List of Publications

[1] Kumari, B., and Sreevalsan-Nair, J.,"An Interactive Visual Analytic Tool for Semantic Classification of 3D Urban LiDAR Point Cloud," (to appear) in Proceedings of the 23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, Seattle, November 2015.

[2] Kumari, B., Ashe, A., and Sreevalsan-Nair, J., "Remote Interactive Visualization of Parallel Implementation of Structural Feature Extraction of Three-dimensional Lidar Point Cloud," In Big Data Analytics (pp. 129-132). Springer International Publishing, December 2014.

Contents

List of Figures

List of Tables

List of Abbreviations

CRFs . . . . . Conditional Random Fields

DoNs . . . . . Difference of normals

DT . . . . . Diffusion Tensor

EM . . . . . Expectation Maximization

GIS . . . . . Geographic Information System

GNSS . . . . . Global Navigation Satellite System

HEM . . . . . Hierarchical Expectation-Maximization

IMU . . . . . Inertial Measurement Unit

ISPRS . . . . . International Society for Photogrammetry and Remote Sensing

LiDAR . . . . . Light Detection and Ranging

NDT . . . . . Normal Voting Diffusion Tensor

NT . . . . . Normal Voting Tensor

PCA . . . . . Principal Component Analysis

RADAR . . . . . Radio Detection and Ranging

RF . . . . . Reference Frame

SDT . . . . . Structure Diffusion Tensor

SF . . . . . Surface fitting

ST . . . . . Structure Tensor

CHAPTER 1

INTRODUCTION

Over the last few decades, LiDAR (Light Detection and Ranging) technology has gained popularity for accurately capturing topographic information of various earth surface features. LiDAR technology captures topographic information with high resolution in less time as compared to other remote sensing technologies like satellite imaging. Therefore, it is used for various applications that include three dimensional city modeling, planning, disaster management and virtual tourism, etc. For the analysis of urban LiDAR point cloud, semantic labeling is an important step that precedes segmentation, object extraction, and further analysis of objects.

Although extensive work has been done on semantic classification using satellite images, semantic classification in 3D airborne LiDAR data is increasingly in demand because LiDAR technology captures data with high resolution and takes less time for data collection. Typically, airborne LiDAR data is very large in size, non-uniformly sampled, not spatially indexed, and contains diverse and complex objects which are sometimes occluded or partially visible. Hence, semantic classification in airborne LiDAR data becomes a more challenging task due to the nature of LiDAR data.

We focus on two different kinds of classification of 3D urban LiDAR point clouds: semantic and structural classifications. Labeling points to pre-defined object classes in urban LiDAR point cloud is known as semantic classification or semantic labeling

of objects. Extraction of geometrical (structural) features in 3D point cloud gives a geometric or structural classification.

Semantic Classification Several automatic methods using supervised and unsupervised techniques have been studied in literature for semantic labeling or semantic classification of objects in urban LiDAR point cloud. Supervised methods require class label as training data to train classifier. These methods use training data to find suitable feature descriptor for clustering. However, training data is not readily available in case of LiDAR data-sets. The user needs to generate training data manually which is tedious and compute-intensive. Also in case of data-set, which contains objects which are not present in training data, a user has to generate the training data for them and trained the classifier. This scenario may occurs very frequently because of change in structure of objects with time, and different areas contain different kind of objects. For example, the shape and structure of buildings will not be same in rural area and modern city. Therefore, a user may not be able to get correct results using same training data-set where objects present in training data and test data belongs to same class but their shape or structure are drastically different from each other. Hence, training data is data-specific and it has to be regenerated for each new kind of objects.

On the other hand, unsupervised methods do not require any training data. Unsupervised methods compute feature descriptor from the data using prior information about the data. Thus, in both supervised and unsupervised methods, feature descriptors are either available with dataset or are derived from dataset. The associated problem with these approaches is the lack of guarantee in accessibility and availability of domain expert for generating training sets and analyzing feature descriptors. For example, a user who may need to do the classification, may not have sufficient domain expertise. In these cases, they need a help from an external domain expert and as follows, the availability of such experts are not always guaranteed.

A domain expert, who analyses data and generates the training set or evaluates the output, becomes indispensable for semantic classification of objects in urban LiDAR data. However, the functional need of users like urban planners, spatial data users, and engineers is to explore and analyze various features of these datasets, especially new ones, for an initial evaluation, where a domain expert may not be required. In order to reduce the dependency of users on domain experts for semantic classification while providing the flexibility of exploration and analysis of data, **we propose to use an interactive unsupervised method for semantic classification, and a visual analytic framework** for the same.

Our visual analytic framework identifies clustering parameters using visualization techniques, and uses interactive unsupervised hierarchical clustering method for semantic classification. Therefore, our method is independent of training data, and enables user to perform semantic classification in the absence of training data and domain experts. Although urban LiDAR point cloud contains diverse and highly complex objects, our method is independent of the complexity and structure of the objects in it. For example, some data may contain villages where houses are constructed in a historical way with lots of vegetation while other data may have modern town with taller buildings and narrow streets with less vegetation. Our tool can be used to explore and analyze new datasets and perform initial classification. It can also be used to generate training data for supervised methods.

The advantage of our visual analytic framework is that a user need not have expertise on LiDAR technology or visualization. With the minimum knowledge on user-defined parameters, feature descriptor and object classes, a user from diverse background can use our tool to perform structural classification and semantic classification of objects in urban LiDAR data. Therefore, our tool can be useful for such cases where a user does not have domain knowledge, training data is not available or domain expert is not available to perform classification.

Structural Classification Geometric (structural) features such as line- and surface- type features are used as feature descriptors for semantic classification, which are not usually preserved in the semantic classification. Line-type features are essential in determining important topological structures in terrain and non-terrain, such as breaklines, ridges, valleys, etc. This information can be useful for geologists or urban planners to gain more insights about data such as crack in buildings, etc. Hence, line-type features need to be preserved in semantic classification for further analysis of data. Therefore, semantic classification should be geometric- as well as contextual- aware. **We propose an augmented semantic classification** of point cloud, which is geometric- and contextual- aware, by combining structural and semantic classification.

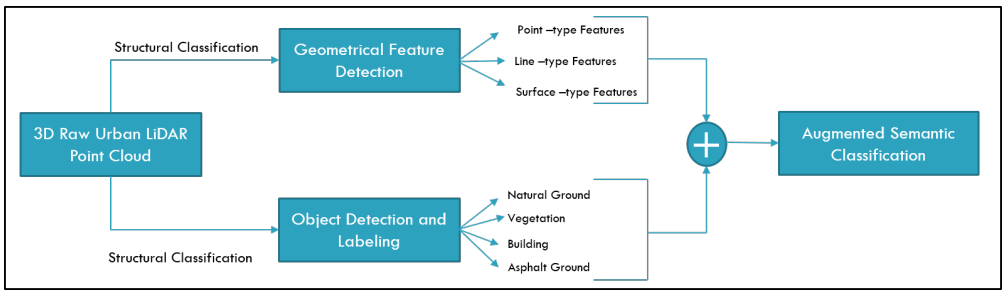


Figure FC1.1: Summary of problem statement. We perform structural and semantic classification of objects in urban LiDAR data, and combine both classification to get augmented semantic classification.

Several methods have been studied in the literature for extraction of geometric features in 3D point cloud. Some of these methods are local tensor-based methods, which in comparison to global methods, detect features accurately in a local frame of reference, and are relatively computationally less expensive. We have explored local second order symmetric semi-definite tensor-based methods for structural classification. Across these tensor-based methods, different types of second-order symmetric tensors have been used to encode the local geometry of a point, such as structure tensor, normal voting tensor, and anisotropic diffusion tensor.

In literature, anisotropic diffusion tensor has been used for extraction of structural

features in triangle meshes of 3D point cloud. We have determined that the anisotropic diffusion tensor can be used to extract structural features in unstructured point cloud data. Therefore, we propose extending the local tensor-based feature detection method using anisotropic diffusion tensor from triangle meshes to unstructured point cloud. Since LiDAR is an unstructured point cloud, we can use anisotropic diffusion tensor to extract structural features in LiDAR point cloud. We introduce two different strategies for generating anisotropic diffusion tensor to encode underlying geometry of a point in an unstructured point cloud. We compare the output of feature classification for local second order semi-definite tensor-based structural feature detection methods for unstructured point cloud.

While analysing the qualitative performance of different local tensor-based feature detection methods, we have found that no well-defined metric exists in the literature for their comparison. However, two predominant parameters namely, shape of local neighborhood and likelihood of points falling into different feature classes show geometrical nature of the point. Hence, these two parameters can be utilized as metrics for comparison of different local tensor-based feature detection methods. This leads to the prospect of using visualization techniques to compare the shape of local neighborhood of a point and likelihood of a point falling into different feature classes. Shape of local neighborhood of a point can be studied by visualizing the shape and orientation of tensor encoding the local geometry of that point. Saliency maps can be used to visualize the likelihood of points falling into different structural feature classes, and glyph visualization for shape and orientation of tensor.

1.1 Problem Statement

Our primary goal is to interactively explore and label the objects in 3D urban LiDAR point cloud in the absence of training data and with the least intervention of domain

expert. Our objective is to address the following two important challenges in urban LiDAR point cloud:

- Semantically label the objects in 3D urban LiDAR point cloud such that geometric as well as contextual information are preserved in the semantic labels as shown in the Figure FC1.2.
  - Additionally, the semantic classification algorithm will have to work with large-scale datasets, and provide real-time interactive visualizations. Hence, we further discuss scalability of our method.
- Accurately detect structural features in unstructured point cloud using second order semi-definite tensor.
  - Currently, there is no well-defined metric in the literature for comparing the efficiency of different local tensor-based structural feature detection methods. Hence, we identify metrics for assessment of local tensor-based structural feature detection methods.

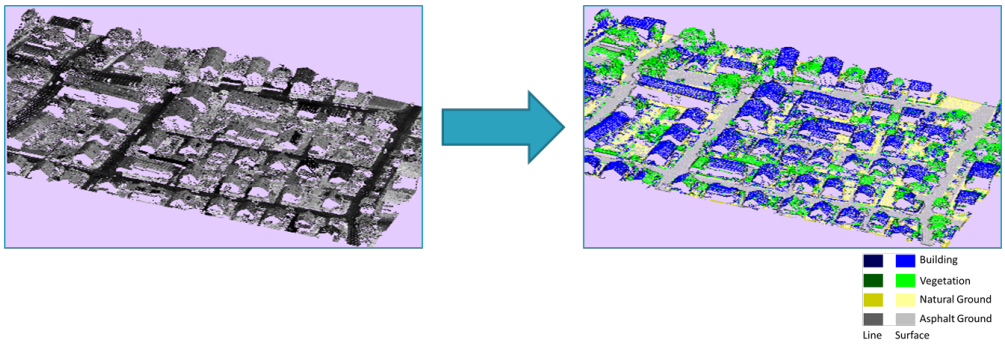


Figure FC1.2: Augmented semantic classification of objects into four different semantic classes: building, vegetation, natural ground and asphalt ground; and two structural classes: line and surface; for area 3 of Vaihingen dataset [1].

The purpose of this thesis is to explore and analyze new urban LiDAR datasets and perform initial classification in real time. We have worked on data derived from images

captured using airborne LiDAR technology for urban area and tested our algorithm on benchmark datasets provided by ISPRS.

1.2 Contribution

We have developed a visual analytic tool for exploration of data and for providing augmented semantic labels to objects in 3D urban LiDAR point cloud. Using our tool, user can explore and analyze data as well as semantically label them. It gives the flexibility to user to explore new dataset and perform initial coarse classification for quick processing for instance, for data quality control. The followings are our contributions:

- We propose a visual analytic framework which enables user to compute structural classification, interactively determine the hierarchy for the semantic classification, and visualize the augmented semantic classification in real-time.
  - We propose a novel interactive divisive (top-down hierarchical) clustering method, which is based on unsupervised machine learning methodology and is data-driven, for semantic classification. We propose to use a tree visualizer to locally choose clustering parameter, and perform clustering for semantic classification.
  - We propose an augmented semantic classification of point cloud, which is geometric- and contextual- aware, by combining multi-scale structural and semantic classifications. Each point in the cloud has a label as a tuple, one from structural classification and other from semantic classification.
- We propose to use second-order symmetric positive semi-definite diffusion tensor for structural feature extraction in unstructured 3D point cloud.
  - We introduce two different strategies for generating anisotropic diffusion tensors to detect structural features in unstructured point cloud.

– We propose to use the following two metrics for comparing different local second order symmetric positive semi-definite tensor-based structural feature detection methods: (1) shape of local neighborhood of a point, and (2) likelihood of a point falling into different feature classes.

1.3 Thesis Structure

The structure of thesis as follows: Chapter 2 provides a background study on LiDAR technology and literature survey is discussed in Chapter 3. Chapter 4 discusses our algorithm for extraction of structural features, semantic classification and, augmented semantic classification in 3D urban LiDAR point cloud. Chapter 5 provides details about designing criteria and implementation of our visual analytic framework. Chapter 6 discusses the local tensor-based structural feature detection methods. Chapter 7 discusses the experiments and results. Chapter 8 concludes the thesis with the future work.

CHAPTER 2

LIDAR TECHNOLOGY

LiDAR, Light Detection and Ranging, is an active remote sensing technology that collects topographic information of objects on earth's surface. It is similar to RADAR (Radio Detection and Ranging) except that it uses laser pulses instead of radio waves. Although this technology has been in existence for such a long time, it attained public awareness only after Apollo 15 mission in 1971, when the first space-based LiDAR instrument was used for measurement of lunar surface around equator [2]. Eventually as it gained popularity, it received more wider acceptance as a technology that uses lights of different wavelengths for detecting distant objects. In this chapter, we have referred to [3–5] for background study on LiDAR technology.

Broadly, there are two types of LiDAR system: 1) Ground or Terrestrial, and 2) Airborne. In the terrestrial LiDAR system, laser scanner is mounted either on a moving vehicle or a device like tripod or stationary vehicle and collects topographic information of surrounding objects lying within the laser scanner's range. Terrestrial system is used to conduct highway and rail surveys, create 3D city models of interior and exterior space, locate or analyze light poles and wire, etc. In the airborne LiDAR system, laser scanner is mounted on an aeroplane or a helicopter and shoots laser pulses towards target on earth's surface and collects reflected laser pulses. It captures aerial view of the target and computes geo-referenced information of objects in the target area. There

are two types of airborne sensors: topographic and bathymetric. Topographic sensors typically use a near-infrared laser to compute geo-referenced information of objects on landforms above sea level. Bathymetric sensors use water-penetrating green light to compute geo-referenced information of objects on landforms below sea level. Topographic maps show elevation of objects while bathymetric maps show depth information of objects. We have used airborne LiDAR data of urban area for our research work.

2.1 Airborne LiDAR Technology

An airborne LiDAR system consists of: 1) An airborne vehicle like small charter, helicopter, plane, etc., with laser scanner and, 2) Inertial moment unit (IMU) device and, 3) **Global Navigation Satellite System (GNSS)** as shown in the Figure FC2.1. GNSS provides precise positioning information of the aircraft. An accurate IMU monitors the orientation (roll, yaw and pitch) of the aircraft. Laser scanner mounted on aircraft shoots laser pulses towards ground on the target area at the rate of hundreds of pulses per second and collects the reflected pulses and records the time between transmitted and reflected pulses. Since laser travels at the speed of light, the range to the target is determined using the equation:

Range = ct/2

where c is the speed of light and t is total time of flight of a laser pulse. Target'coordinates are determined using laser's position in three dimensional space, range to target and scan angle. GNSS records the location of aircraft at fixed interval, and another GNSS on the ground provides differential correction for more accurate position estimate. Scan angle is recorded by LiDAR instruments, and IMU tracks the airplane's motion. Topographic information is computed using this information and results in 3D unstructured point cloud with X, Y and Z or latitude, longitude, and elevation information of objects in the target area. Generally, LiDAR data provides **target horizontal**

(X-Y) spacing capability up to 30 centimeters and a vertical (Z) accuracy from 10-20 centimeter.

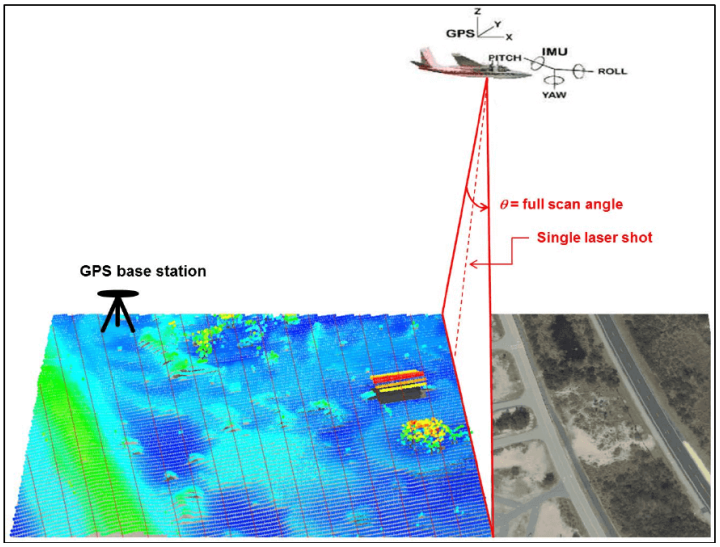


Figure FC2.1: Airborne LiDAR Technology. Image Courtesy [3].

There is also a multiple return LiDAR system as shown in the Figure FC2.2, which can capture up to five returns per pulse. It enables us to capture the information under forest canopy and understory by collecting multiple returns per pulse. Multiple return is a very useful feature descriptor to distinguish vegetation from buildings. It helps us to get better ground reference and has other benefits like collecting same point density using lower-flying sensor with a lower PRF (Pulse Repetition Frequency) or sensor flying at higher altitudes with higher PRF.

2.2 Active and Passive System

Based on the source of energy, remote sensing technologies can be broadly classified into two categories: Active and Passive System. Active system can generate as well as detect light beams. On the other hand, passive system can only detect the waves or light coming from natural source of energy, for example, sun, radiation from the earth,

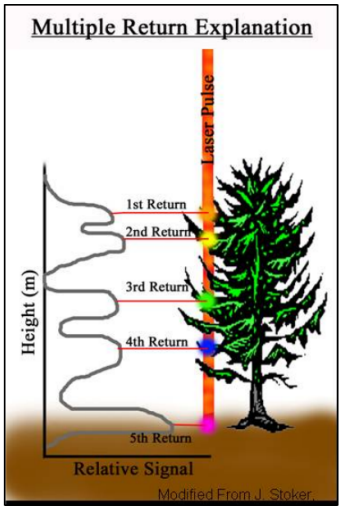


Figure FC2.2: Multiple Return System. Image Courtesy [3].

etc. LiDAR uses active sensor which emits energy in the electromagnetic spectrum. Therefore, it can be used to collect data at day as well as at night time. It emits discrete pulses of radiation at a particular frequency within the range of infrared band with wavelength ranges of 500-1000 nm.

2.3 Advantages and Disadvantages

LiDAR is used heavily for collection of topographic data. As stated in [4]:

> "This technology offers several advantages over the conventional methods of topographic data collection viz. higher density, higher accuracy, less time for data collection and processing, mostly automatic system, weather and light independence, minimum ground control required, and data being available in digital format right at beginning".

Owing to these advantages, this technique is popularly used for flood-modeling and similar applications such as, bathymetry, geomorphology, glacier modeling, etc. LiDAR technology has some limitations such as difficulty in surveying during rainy or

misty conditions, inability to accurately delineate stream channels and shorelines often visible on photographic images, etc.

2.4 LAS Format

LiDAR data are stored in a binary format, called LAS file format which is developed by the American Society for Photogrammetry and Remote Sensing (ASPRS) LiDAR Committee. LAS format enables us to share data among different vendors, users, software, etc., by providing a standard format for efficient storage of LiDAR datasets. The current version of LAS approved by ASPRS is 1.4 [1] which was released in July, 2013. Prior to this, there were 1.0, 1.1, 1.2 and 1.3 versions. LAS format consists of a public header block, one or more variable length records, and point data records. All LAS versions are backward compatible.

[1] http://www.asprs.org/a/society/committees/standards/LAS_1_4_r13.pdf

CHAPTER 3

LITERATURE SURVEY

Semantic classification of objects in urban LiDAR point cloud is an active research area in geographic information system (GIS) community. Our goal is to perform classification without training data with least intervention from a domain expert. We propose an interactive unsupervised clustering method for semantic classification and a visual analytic framework for the same. We propose an augmented semantic classification, which is geometric- and contextual- aware, by combining structural classes (line, surface) with four semantic classes namely building, vegetation, natural ground and asphalt ground. There is a large body of work in the area of semantic classification of objects in LiDAR data. In this chapter, we discuss some of the work which is relevant to various modules of our proposed framework. Our visual analytic framework for the augmented semantic classification and the interactive divisive clustering for semantic classification are novel ideas, and we have not found any prior relevant work on those lines.

Multi-scale Structural Point Classification Structural features can be extracted from data either in their raw state or in their processed state, such as, meshes. While mesh generation from point data is a well-researched method for analyzing point cloud data, it is generally computationally intensive. LiDAR data does not have explicit connectivity information, which implies that computing meshes is an overhead. Meshes are more heavy-weight data structure as compared to point cloud, and LiDAR data are large scale

datasets. Therefore, we have worked with point cloud extracted from LiDAR datasets, and used point-based techniques for structural feature classification.

Point based techniques for structural classification can be broadly divided into following categories, based on their key methods [6]: principal component analysis, graph theory, surface reconstruction, Gauss map, Voronoi-based and local tensor based techniques. Weber et al. [7] allude to some of these methods as the ones dedicated solely to point-sampled geometry. The reader may find a good categorization of several such methods in [6, 7].

Summarizing, even though the graph theory-based and Gauss map-based techniques provide a wide range of feature points in the presence of noise, they do not detect the nature of the point very accurately. **On the other hand, though surface reconstruction- and Voronoi- based techniques are computationally very expensive, they detect and classify points accurately as compared to other methods** [6]. Local tensor-based techniques for structural classification are known to be a trade off, which detect features sufficiently, accurately, and computationally less expensive, when compared to the surface reconstruction- and Voronoi- based techniques. Hence, in our approach, we have used local tensor-based technique to extract structural features from point cloud.

Keller et al. [8] have used a multi-scale approach using Principal Component Analysis (PCA) and curvature estimation methods to classify the points into different feature classes. Keller et al. have computed feature graph using PCA to obtain the feature lines. However, as with all graph theory-based methods, the algorithm suffers due to sensitivity to noise and dependency of accuracy on sampling rate of data. Blomley et al. [9] have used multi-scale approach using shape distribution features for point classification, as opposed to covariance features, proposed by Keller et al. [8]. We have used the latter in our work, even though Blomley et al. have stated the challenges of using covariance features, which include finding the optimal scale. Keller et al. have

alleviated the problem by averaging across multiple scales.

Object-based and Contextual Point Classification Haala and Brenner [10] combine a normalised DSM from LiDAR data with the three spectral bands of a scanned color infrared (CIR) image and then uses ISODATA (Iterative Self-Organizing Data Analysis Technique algorithm) for clustering. For this purpose, the height data and the images have to be co-registered. Awrangjeb et al. [11] stated that the methods which superimpose imagery and point cloud data face various problems such inaccurately co-registered imagery and LiDAR data, some information is not clearly in imagery due to shadow and occlusion. To resolve these issues, we have used a single mode dataset, namely 3D airborne LiDAR point cloud for classification.

Song et al. [12] have given an analysis of the effectiveness of using LiDAR intensity data for land-cover classification using LiDAR intensity data instead of the multi-spectral data, where a uniform grid derived from point cloud is used. Chehata et al. [13] have used multiple classifiers using random forest for supervised classification based on several classes of LiDAR parameters. Further, they have elaborated on the importance of variables used in the classification. Niemeyer et al. [14] have used conditional random fields (CRFs) for classifying points into building, low vegetation, tree, natural ground, and asphalt ground, using geometrical features as well as an intensity value. They have further improved their results by using random forest in addition to the CRFs [15]. While the afore-mentioned work gives object-based classification, Niemeyer et al. [16] have proposed inclusion of context as an additional cue to the supervised classification. Context helps to exploit spatial locality. In similar lines, we refer to our object-based classification as a contextual one.

Object Extraction & Classification While our work pertains to point classification, which will potentially be used for segmentation and object extraction, there exists a

class of algorithms which extract objects to perform segmentation and then, classify objects. Golovinskiy et al. [17] have extracted objects from point cloud by using variants of several conventional methods such as normalized cuts, and then, labeled the objects using a classifier. Eich et al. [18] have extracted objects using region growing algorithm and then, using spatial axioms to semantically classify shapes of the objects in indoor environments. Rabbani et al. [19] have used smoothness constraints, derived from local normals and point connectivity in local neighborhood to compute segmentation.

Local Feature Descriptors The semantic classification of real world datasets is not an exact science, as it is subjective to the topography, data acquisition, the heterogeneity of materials (man-made, natural, vegetation, asphalt, etc.), LiDAR parameters, and other meta-data of the dataset. Hence, point-wise local descriptors are required at each point for classification. Tombari et al. [20] have categorized local descriptors into signature and histogram based feature descriptors, and we have used this categorization implicitly in our proposed method. Using both types of local descriptors has been proven to give better segmentation of the point cloud [14, 21]. In our proposed work, we use signature descriptors for detection of structural features for structural classification. We use a combination of signature and histogram descriptors in the case of semantic classification.

Other Similar Methodologies Ramiya et al. [21] have used geometrical analysis for the segmentation using both curvature and colorimetric distance for colored LiDAR data; and supervised learning for the classification. The steps in their segmentation process are very similar to those of our structural classification. Lari et al. [22] have derived cylindrical neighborhood-based analysis for semantic classification, especially of the Vaihingen dataset [1]. We have used **cylindrical as well as spherical neighborhood** for computation of feature descriptor.

Other applications of unsupervised classification techniques, such as [23, 24], exist. Ghosh and Lohani [23] have used density-based clustering method DBSCAN for extraction of natural as well as man-made clusters of building points exclusively, as the clustering was done using the positional attributes of the points alone. Lafarge and Mallet [24] have used a graph-cut based method for a graph formulated between the points as an energy minimization problem, which gives a very robust segmentation. In our semantic classification, we use both iterative clustering algorithm and region growing algorithm, whose context-aware classification can be compared to graph cut based method [24]. We have used Expectation Maximization (EM) algorithm for interactive hierarchical (Top-down) clustering.

Research in interactive hierarchical clustering is gaining visibility recently, and we have found several papers in interactive agglomerative clustering, which is based on bottom-up approach of merging clusters. Guo et al. [25] have proposed a method of selecting subspaces for clustering of spatial data. Packer et al. [26] have used visual analytic framework for performing interactive agglomerative clustering of spatial data. Jang et al. [27] have used visual analytic framework to visualize dendrogram of the agglomerative clustering and used gesture patterns to determine which clusters to merge. The afore-mentioned work use visualization to select the clusters to merge at every level, thus making it interactive. In similar vein, our work pertains to interactively deciding which cluster to split and what parameters to use for the splitting. While Preiner et al. [28] have used hierarchical EM in an agglomerative fashion for surface reconstruction from point cloud data, our work pertains to hierarchical EM and we use it in a divisive (top-down) fashion.

CHAPTER 4

AUGMENTED SEMANTIC CLASSIFICATION

Extraction of spatial features in LiDAR datasets of urban environment is known as semantic classification or semantic labeling. We classify LiDAR points into four different classes: building, vegetation, natural ground and asphalt ground. It is well known that semantic classification, obtained from learning on LiDAR parameters and geometric features (such as point-, line-, and surface- type feature points) preserves labels for semantic (similar to object-based) classification. Structural (or geometry-based) classification which is obtained from geometric features is usually not preserved in semantic classification. Line-type features are essential in determining important topological structures in terrain and non-terrain such as breaklines, ridges, valleys, etc. Therefore, we propose an augmented semantic classification of point cloud, which is geometry- and contextual- aware, by combining structural and semantic classifications. Thus, we have labels as tuples such as (line, building), (surface, building), etc., by combining (line, surface) structural classes with four semantic classes building, vegetation, natural ground and asphalt ground. Our augmented classification is useful for finding line-type features in the building and asphalt ground classes.

Conventionally, line-type features are computed from grid image, obtained from LiDAR elevation data, rather than from point clouds [29]. However, the methods which superimpose imagery and point cloud data face various problems [11] such as: (a) there

exists unresolved differences in LiDAR data and imagery due to non-availability of accurately co-registered imagery and LiDAR data as both are collected separately, and (b) some information is not clearly available in imagery due to shadow and occlusion. In other words, imagery and LiDAR data have different characteristics and it becomes difficult to combine the feature descriptors of imagery and LiDAR data. To resolve these conflicts and achieve efficient computation, we propose extracting structural and contextual information from a single mode dataset, namely the point cloud.

We use a multi-scale approach for extracting structural features from point clouds directly [8], which ensures persistence of features across multiple scales as discussed in Section 4.2. We have extended the multi-scale approach to semantic classification to extract spatial features. For semantic classification, we use our *tree visualizer tool* to interactively determine a divisive (top-down hierarchical) clustering using user-selected feature descriptors, to determine four specific classes, namely, building, vegetation, natural ground, and asphalt ground.

Our method comprises of two modules: (a) augmented semantic classification (Section 4.4), and (b) visual analytic framework which enables visualization and data analytics in an interleaved manner. Augmented semantic classification is discussed in detail in this chapter and visual analytic tool will be explained in Chapter 5. Our classification algorithm has three steps of classification: structural, semantic, and augmented semantic classifications. Structural and semantic classification are discussed in Sections 4.2 and 4.3 respectively. Augmented semantic classification is explained in Section 4.4. The proposed system is summarized in Figure FC4.1, where the GUI is built upon visual analytic framework and the data is preprocessed before the classification.

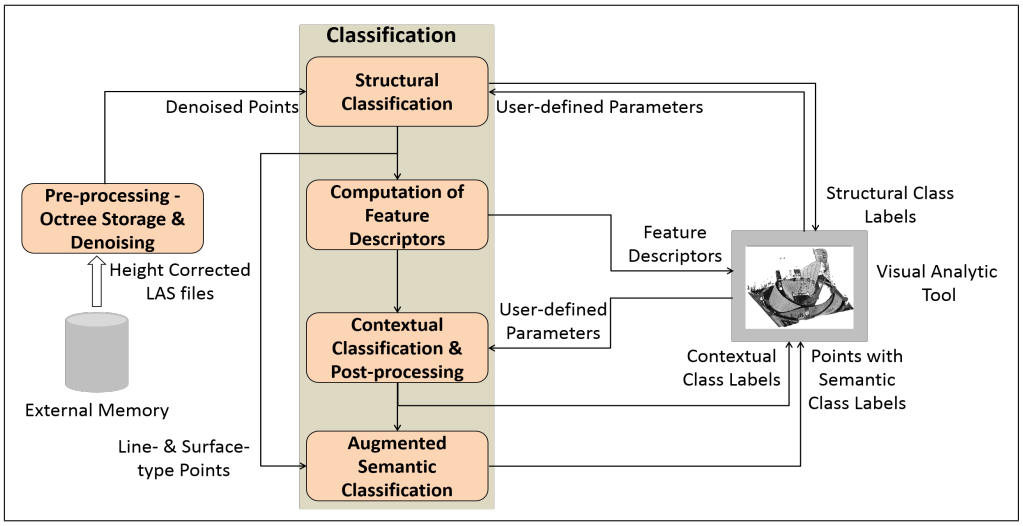


Figure FC4.1: Our proposed visual analytic framework to achieve augmented semantic classification.

4.1 Multi-scale vs Adaptive Scale Approach:

In literature, there are two approaches namely, multi-scale and adaptive scale based approach. The multi-scale approach used is different from the adaptive scale one. The adaptive scale based approaches use optimal neighborhood size per point to reveal the most optimal shape of the local neighborhood [30–32]. In case of multi-scale based approaches, lower and upper bound of scale are defined. Feature values are computed at different values of scale within the upper and lower bound of scale and then, feature values computed at different scale are summed up. Mean of final feature value has been used to find the shape of local neighborhood. However, adaptive scale based approaches also need bound of scales like multi-scale based approaches[1].

Demantké [30] et al. have determined the bounds of scales, i.e., local neighborhood radius, and described how to find the optimal scale. On the other hand, taking the average of feature values (isotropy, linearity, and principal curvatures) across multiple scales encode the persistence of features [8]. Brodu and Lague [33] have considered all

[1] In [30–32], their methods are referred to as multi-scale approach, even though they are adaptive scale, using optimal scale at each point; we refer to them as "adaptive scale" to make a distinction with our multi-scale approach.

scales at once by building a higher-dimensional feature space, and applying a classifier on it using supervised learning method. This approach proves to be computationally intensive for unsupervised method such as ours while it gives promising results for geometric classification by taking average of multiple scales as in [8].

We have used multi-scale based approach in our algorithm instead of using optimal scale. The reason is that LiDAR data contains complex and diverse objects of different sizes. Therefore, it might be possible that a single scale will not be able to capture the true geometrical nature of all objects. For example, some objects in the data show its true geometrical shape at m1 scale while other objects show their true geometrical shape at m2 scale. If we use m1 scale to detect the true geometrical shape for all objects, we may wrongly detects shape of some objects. It might not be possible to capture such cases with single/adaptive scale approach as they give hard classification. Therefore, we have used multi-scale based approach. In our multi-scale approach, encoding persistence of features across multiple scales gives the likelihood of a point having neighborhood of a specific shape; thus, averaging the feature values gives a larger geometrical information.

In our multi-scale approach, encoding persistence of features across multiple scales gives the likelihood of a point having neighborhood of a specific shape; thus, averaging the feature values gives a larger geometrical information. In certain cases, where the likelihood of a point belonging to a specific primitive is not dominant using a single choice of scale, albeit optimal, the error for assigning a geometric class label will be higher than the case when the decision has been made by using information across several scales. For example, points lying close to an edge or crease will show roughly the same likelihood of belonging to a line primitive or a surface primitive, in such cases, finding an optimal scale will be not beneficial. The multi-scale approach gives more separable feature space, increases the spatial resolution and gives soft clustering as compared to the single scale approach. Hence, we have used multi-scale based approach

for our algorithm.

4.2 Structural Classification

The structural features of interest in 3D point cloud include sharp feature lines, such as ridges, ravines, crest lines, edges of buildings, etc and surface features like roof tops. We are mainly interested in surface- and line- type feature points. Several methods for structural feature detection exist in literature as discussed in Chapter 3.

LiDAR dataset in raw form contains noise due to instrumental error and the environment in which data are collected. Therefore, denoising of data has been done using conditional outliers removal filter. For each point, $r-$ nearest neighbors are searched. If number of neighbors is less than a threshold, then the point is considered as outlier and filtered out.

We have used the algorithm proposed by Keller et al. [8] for structural classification. Keller et al. have proposed an user-assisted system which uses linear dimensionality reduction technique namely, the Principal Component Analysis (PCA) in a multi-scale fashion to determine the stochastic classification of local neighborhood of each point. It classifies the 3D points in the cloud as disc, curve, critical curve, and critical disc points. The disc and critical disc points refer to points on surfaces, and hence are surface- type features while curve and critical curve refer to points on lines (such as creases, borders, and ridges), and hence are line-type feature points. This algorithm has been referred to as structural feature-based, geometry-based, or topology-based, owing to the eigen-analysis (i.e. spectral analysis) of the covariance matrix or the structural tensor of the local neighborhood, which gives the local reference frame (RF) [20]. The local RF computed using eigenvalue decomposition of the covariance matrix has been proven to be one of the most robust local RF [20]. The likelihood of the points falling into any of the three classes (i.e., point, line, surface) is determined by saliency values, C_p ,C_l

and C_s, which are computed using a multi-scale approach. Since the saliency values are computed from local RF, they are signature descriptors.

However, the algorithm proposed by Keller et al. has the following limitations: (a) it is computationally expensive, and (b) it is heavily dependent on user-defined parameters. We have identified that the algorithm proposed by Keller et al. is embarrassingly parallel and hence, we propose parallel implementation for the algorithm proposed by Keller et al. Parallel implementation for the algorithm proposed by Keller et al. is discussed in Chapter 5. While Keller et al. state **(b)** as a specific advantage of their application [8] to be user-controllable, we have observed that adjusting the parameters for different datasets is a difficult task. Hence, we have explored alternative approaches using local tensors as feature descriptors which are discussed in Chapter 6.

4.3 Semantic Classification

We perform hierarchical divisive clustering in a top-down fashion using EM clustering for semantic classification to obtain four object or semantic classes. In LAStools, which is an application for exploring and visualizing files in .las format, Hug et al. [34] have used an object hierarchy constructed from contours for classification. In contrast, our hierarchical classification is point cluster-based. We enumerate the feature descriptors which we have considered as clustering parameters in Section 4.3.1. We explain the idea and applicability of interactive divisive (top-down hierarchical) clustering for semantic classification in Section 4.3.2. Clusters belonging to a class are given the same label. Furthermore, to ensure spatial locality of the classes and preserve the context, region growing within a class has been performed which is explained in Section 4.3.3.

4.3.1 Feature Descriptors

In computer vision, feature descriptors represent specific characteristics of point cloud, and multiple features give the feature vector of the point known as feature descriptor. Feature descriptors can be used to distinguish different classes of objects in point cloud as each class has its own range of values for a specific feature descriptor. Feature descriptors can be categorized into two classes - signature and histogram based [20]. Tombari et al. state that a proper combination of signature and histogram feature descriptors is required for shape inference. We have used the following combination of feature descriptors for our semantic classification:

1. Values at the point:
   (a) **Height (h)** is normalized height i.e. height of objects above the ground, and is good for separating ground and non-ground points. Details on computation of normalized height are given in Section 5.2.1.
   (b) **Intensity (I)** is sensitive to the incidence angle of the laser beam [16] and we have used normalized I to distinguish the asphalt ground from natural ground. We normalize intensity between minimum and maximum value of range. We have not considered normalization of intensity for incidence angle as it is outside the scope of our current work.
2. Statistical features of the local neighborhood:
   (a) **Height variance (σ_h^2)** is the variance of the normalized height computed from $r-$nearest neighbors, in cylindrical neighborhood, and is known to be high for vegetation and gabled roofs.
   (b) **Height range (δ_h)** is the difference between maximum and minimum normalized height of $r-$nearest neighbors, in cylindrical neighborhood, and is known to be high for vegetation and boundary of buildings.

(c) **Ratio of point densities ($\mathscr{R}_\rho$)** in spherical neighborhood to that in cylindrical neighborhood of radius. We have observed that it is low for building facades and vegetation.

(d) **Difference of normals (DoNs)** $\mu(\delta_n)$ is the mean of orientation angles of vectors between the point and neighbors in the local spherical neighborhood $\mathscr{N}_p$. Difference of normals at the point (p) and a neighbor p_i, is given by $\delta_n(p,p_i) = \frac{p \cdot p_i}{\|p\| . \|p_i\|}$ and $\mu(\delta_n) = \frac{\sum_{p_i \in \mathscr{N}_p} \cos^{-1} \delta_n(p,p_i)}{cardinality(\mathscr{N}_p)}$.

3. Structural features of the local neighborhood:

(a) Eigen-based LiDAR features are computed using the eigen values, $\lambda_0, \lambda_1, \lambda_2$, of structure tensor which is computed using $r-$nearest neighbors in spherical neighborhood (different from other descriptors). Given $\lambda_2 \geq \lambda_1 \geq \lambda_0$, **anisotropy ($\mathscr{A}$)** and **sphericity ($\mathscr{S}$)** are the features computed as: $a = \frac{\lambda_2 - \lambda_0}{\lambda_2}$, and $s = \frac{\lambda_0}{\lambda_2}$. $\mathscr{A}$ is known to be high for planar regions and low for non-planar regions and conversely, $\mathscr{S}$ is known to be high for non-planar regions and low for planar regions; and hence can be used to distinguish vegetation from planar roofs.

(b) **Surface fitting (SF)** tells how best the local surface fits at any point p of point cloud using $r-$nearest neighbors in spherical neighborhood. It is known to be high for planar regions and low for non-planar regions, and can be used for distinguishing planar roof from ground.

(c) **Root mean squared (RMS) curvature** RMS_κ is given as $\sqrt{\frac{{\kappa_1}^2 + {\kappa_2}^2}{2}}$, given κ_1 and κ_2 are the eigenvalues of principal directions at any point p. It is known to be low for surface-type of feature points and high for point- and line-type feature points.

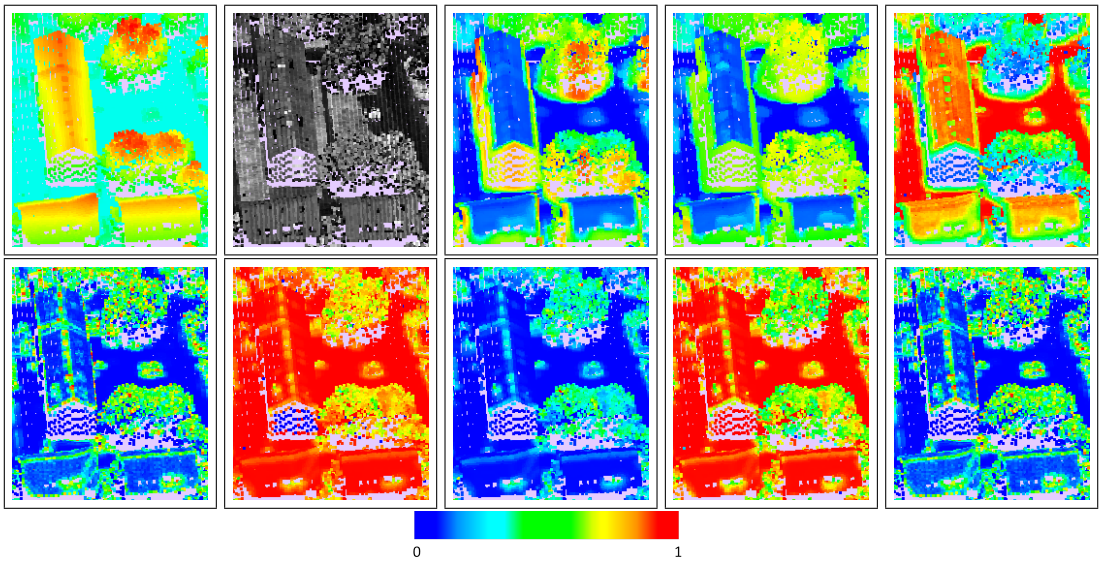


Figure FC4.2: Heatmaps of feature descriptors (left-to-right): point-based ones (height, intensity) and statistical ones (height variance, height range, ratio of point densities in the top and difference of normals (DoNs) in the bottom (first left image)); and structural ones (anisotropy, sphericity, surface fitting and curvature in the bottom) for Area 1 of Vaihingen dataset. Except for the intensity heatmap derived using grayscale spectrum, the others use a rainbow spectrum.

The descriptors in items 2 and 3 can be considered to be histogram and signature based descriptors respectively and together, they are local feature descriptors. Items 1 are point-based feature descriptor. Conventionally, multiple returns in LiDAR point cloud are known to be very important point-based feature to distinguish vegetation from building. However, in the benchmark datasets which we have used for testing, several points do not have multiple returns due to the leave-on-conditions at the time of flight [35]. Therefore, we have not used the multiple returns for our experiments.'

4.3.2 Interactive Divisive Clustering

By interactive hierarchical clustering, we imply that the user can customize the parameters used for clustering at each level or subtree in the hierarchy. Our algorithm is divisive clustering as the user clusters the subset of points described by the node into two clusters at each active node of the k-ary tree. User sub-selects the clustering param-

eter from the set of feature descriptors as described in Section 4.3.1. The user makes an informed decision of the clustering parameters after observing clustering tendency of the feature descriptors using supporting visualizations. Any efficient partitioning clustering algorithm such as EM, k-means, graph cuts, etc., can be used to split the clusters, once the user has confirmed that there are clear clusters for the chosen set of points in the chosen parameter space.

In our clustering hierarchy, the entire point cloud is the root node and at each subsequent level, we split points described by root node into k clusters. While the number of clusters for splitting at each node in the tree can be a variable, we have used k = 2 in our application throughout the tree as it was optimal for efficient computations mainly for three reasons: (a) clustering results can be updated interactively, (b) large-scale datasets can be handled and (c) finding two clear clusters in the parameter space at each split is easier than larger number of clusters. Independently $k = 2$ meets the design requirements of the tree visualizer, as given in Section 5.1.2.

We have used the feature descriptor discussed in Section 4.3.1 as clustering parameter subsequently in the tree. At each active node of the tree, the user selects appropriate clustering parameter after looking at the colormaps of the feature descriptor corresponding to the subset of points in the active node. Then, the user classifies the subset of points in the active node into two classes using chosen clustering parameter. EM Gaussian mixture model has been used for classification.

We have tested our algorithm on the following airborne LiDAR system (ALS) datasets: Data 1 and Data 2 of Vaihingen dataset, from the ISPRS benchmark data [36]. Data 1 is an inner city of Vaihingen consisting of old historical buildings with a complex structure with vegetation. Data 2 is a residential area which contains high rising buildings of different sizes with less vegetation. The point density of the data varies between 4 and 7 $points/m^2$. First we perform structural classification and compute feature descriptors

for semantic classification. Then we use our tree-visualizer tool to perform semantic classification. For semantic classification, we start from root node of the tree as shown in the Figure FC4.3 which contains unlabeled point cloud and perform clustering until **termination condition** is arrived. We merge clusters in some of the leaf nodes which visually imply belonging to the same class. We finally get four specific classes, namely, building, vegetation, natural ground, and asphalt ground as shown in the Figure FC4.10.

Figures FC4.3- FC4.10 show working of one of the tree models which we have created for semantic classification by choosing different sets of parameters for the hierarchical cluster. First, we have classified points into ground and non-ground points, then non-ground points into building and vegetation classes, and ground points into natural ground and asphalt ground. User can play with our tree-visualizer tool and try different combinations of parameters to create different tree models to improve the output.

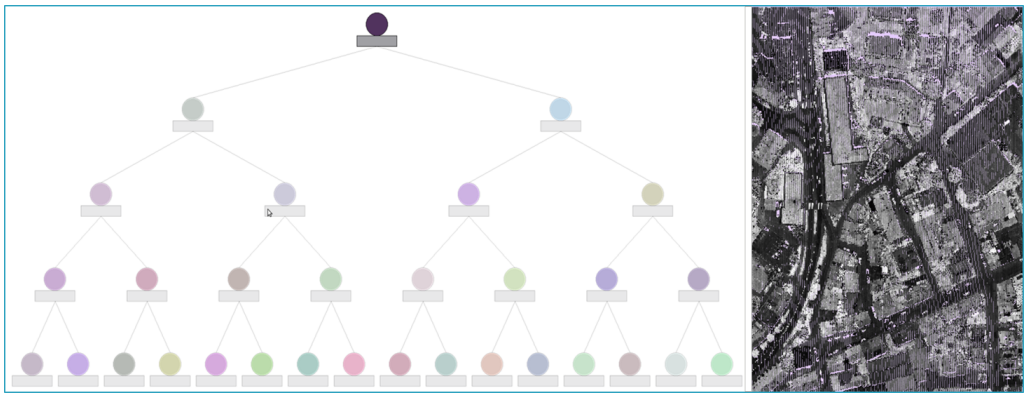


Figure FC4.3: Working of tree-visualizer tool for semantic classification of area 1 of Vaihingen dataset. Step 1: Root node contains unlabeled point cloud.

4.3.3 Post-processing

After semantic classification, we have used region growing algorithm [19] on building and vegetation clusters to ensure that spatial locality is preserved. Subsequently, we have assigned the color to objects belonging to different semantic or object classes according to the color scheme used in [15] as shown in the Figure FC4.11.

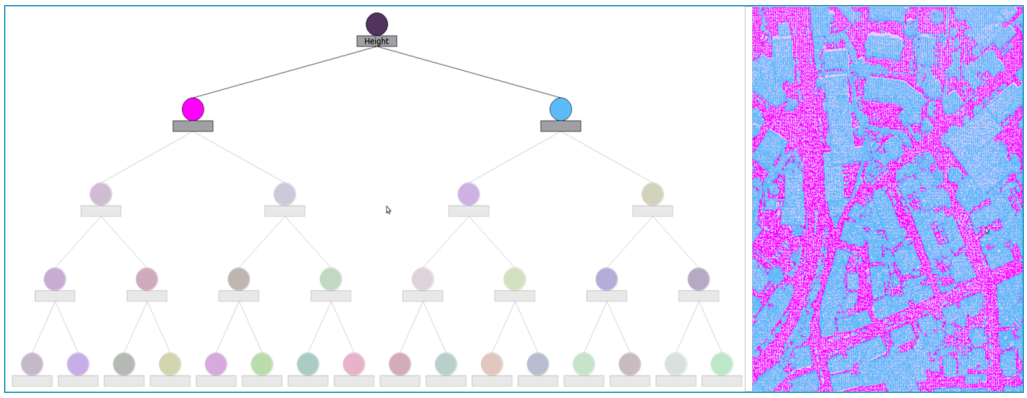


Figure FC4.4: Working of tree-visualizer tool for semantic classification of area 1 of Vaihingen dataset. Step 2: Points contained in the root node are classified into two clusters using height as a clustering parameter.

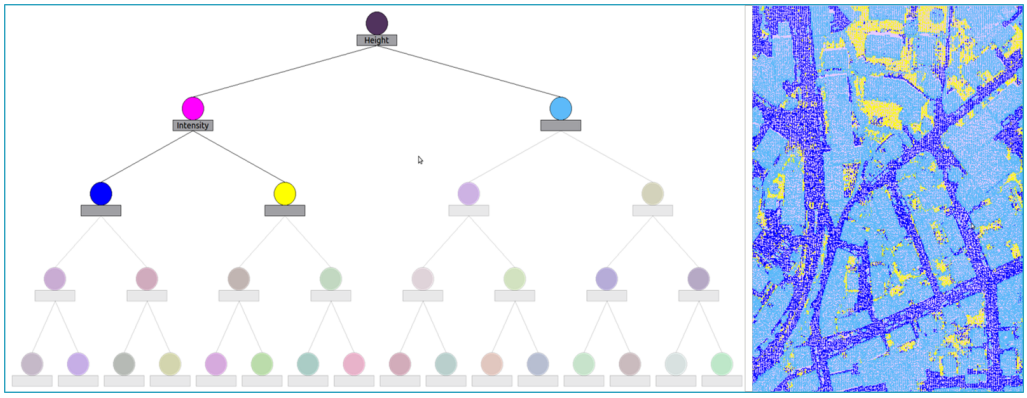


Figure FC4.5: Working of tree-visualizer tool for semantic classification of area 1 of Vaihingen dataset. Step 3: Points contained in the node highlighted as magenta color are classified into two classes, asphalt ground and natural ground using intensity as a clustering parameter. Yellow shows natural ground and dark blue shows asphalt ground.

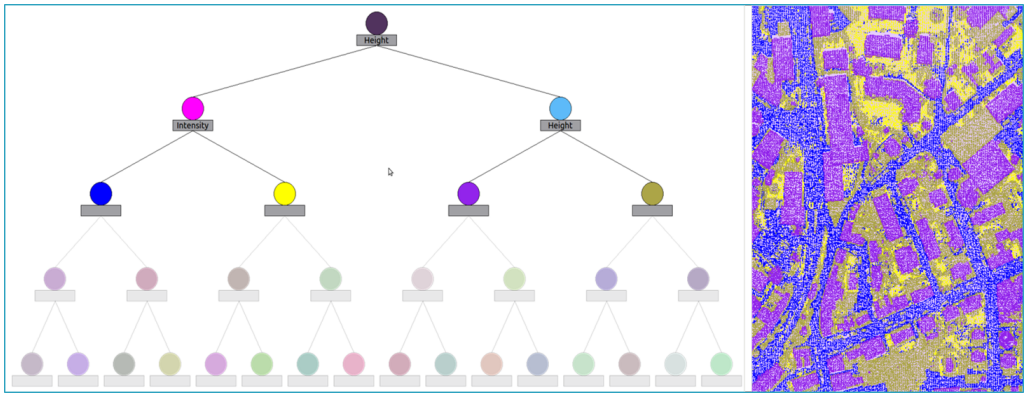


Figure FC4.6: Working of tree-visualizer tool for semantic classification of area 1 of Vaihingen dataset. Step 4: Points contained in the node highlighted as light blue color are classified into two clusters using height as a clustering parameter.

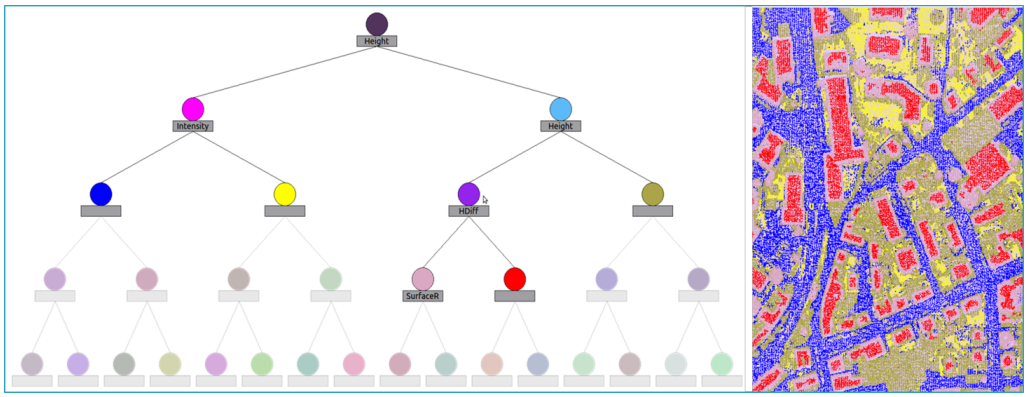


Figure FC4.7: Working of tree-visualizer tool for semantic classification of area 1 of Vaihingen dataset. Step 5: Points contained in the node highlighted as purple color are classified into two clusters using height-range as a clustering parameter.

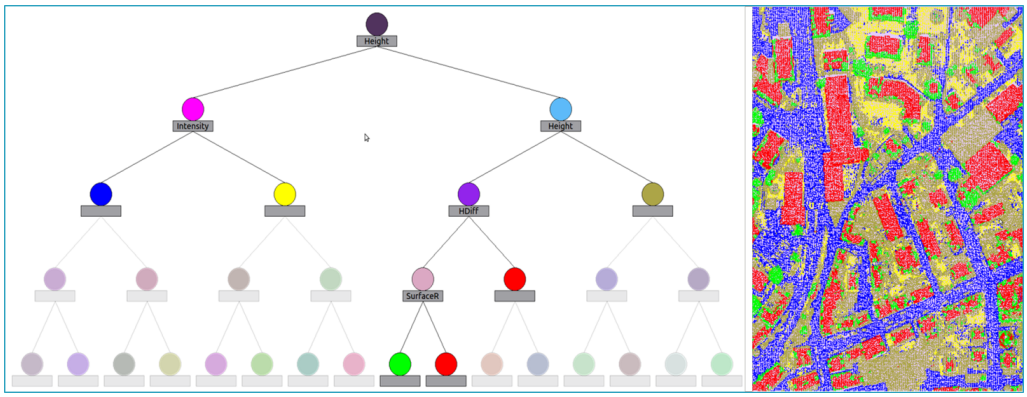


Figure FC4.8: Working of tree-visualizer tool for semantic classification of area 1 of Vaihingen dataset. Step 6: Points contained in the node highlighted as light pink color are classified into two clusters using surface-fitting as a clustering parameter.

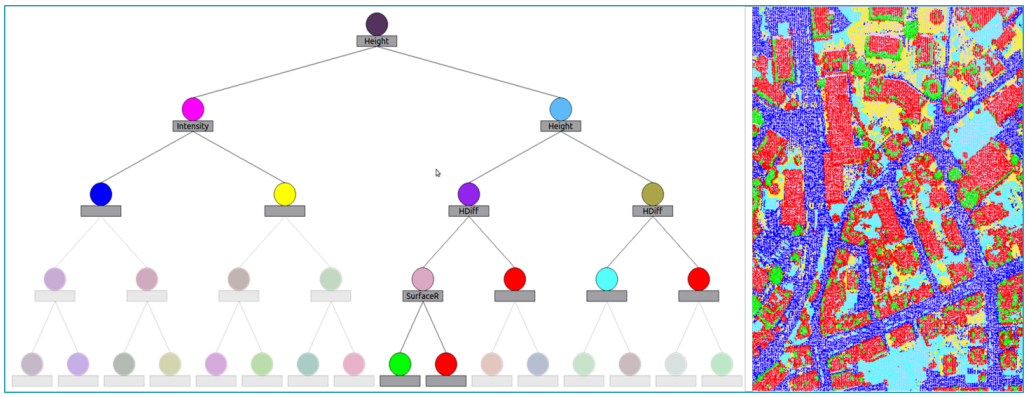


Figure FC4.9: Working of tree-visualizer tool for semantic classification of area 1 of Vaihingen dataset. Step 7: Points contained in the node highlighted as alive drab color are classified into two clusters using height-range as a clustering parameter.

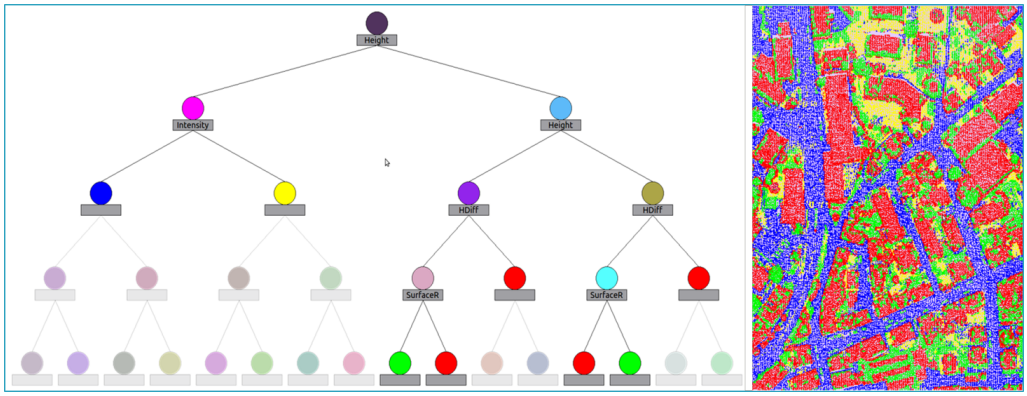


Figure FC4.10: Working of tree-visualizer tool for semantic classification of area 1 of Vaihingen dataset. Step 8: Points contained in the node highlighted as cyan color are classified into two clusters using surface-fitting as a clustering parameter. Points in the leaf nodes belonging to same classes are merged together by assigning same color to them. Red and green colored leaf nodes show building and vegetation respectively. Blue and yellow colored leaf nodes show asphalt ground and natural ground.

4.3.4 1-D Vs N-D Feature Descriptors

Currently, we are using a single parameter for clustering at each active node of tree visualizer for semantic classification. Instead of using a single clustering parameter, we can find optimum set of feature descriptors as clustering parameters. In our framework, a user has option to select multiple features together for clustering. But, it is difficult for a user to select multiple features as clustering parameter by looking into individual heatmap of feature descriptors. Therefore, multi-dimensional visualization techniques such as scatter plot or parallel co-ordinates can be explored to visualize all feature together in a single view space, and to select optimum set of feature descriptors as clustering parameter.

4.4 Augmented Semantic Classification

Conventionally, semantic classification is considered equivalent to object-based classification and additionally, Niemeyer et al. [16] have considered spatial locality in

object-based classification, thus making it contextual. However, in all the previous works, this classification loses the geometric information of the point which was computed and used as a feature descriptor for the object-based or semantic classification. Additionally, the supervised classification tends to reduce the contribution of the geometric dimensions, as given in [13]. While semantic classification use structure classification as feature descriptors, but not preserve labels for structural classification. Preserving the two classifications in the labeling of the points gives a geometry-aware semantic labeling. This can be useful for finding creases or ridges in different objects present in the data. Hence, we propose to preserve labels for both classification - semantic and structural, and we called this classification as augmented semantic classification.

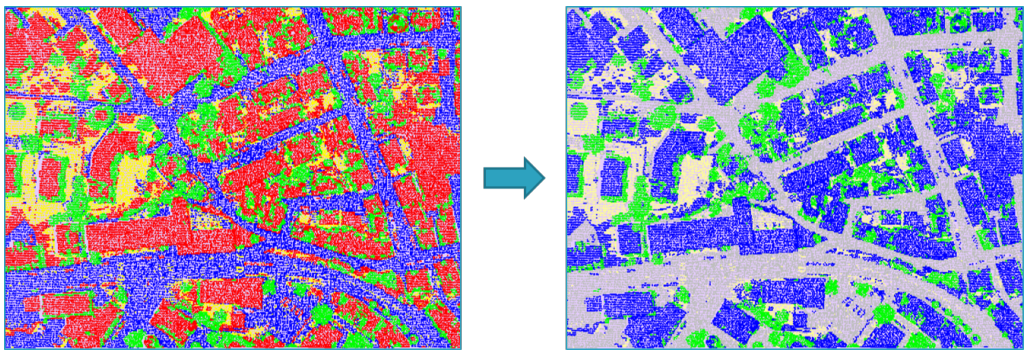
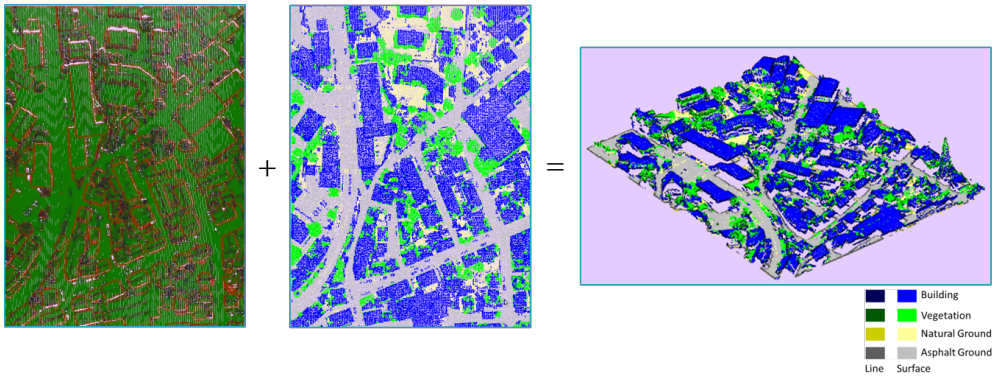


Figure FC4.11: Semantic classification of area 1 of Vaihingen dataset. Spatial locality is preserved using region growing algorithm and color scheme has been assigned to semantic classes according to the color scheme used in [15]. In left image, red shows building, green shows vegetation, yellow shows natural ground, and blue shows asphalt ground. In right image, blue shows building, green shows vegetation, yellow shows natural ground and grey shows asphalt ground.

Our proposed augmented semantic classification aims to preserve geometric as well as contextual information that each point in the point cloud embodies. Thus, each point in the point cloud has a label, which is an ordered pair of geometric and semantic class labels from their respective multi-class classifications. The idea behind preserving the geometric features is that identifying lines in the point cloud is beneficial for retrieving building footprints or road bank boundaries. The lines in the classification outcome

are generally computed using: (a) bounding polygon of clustered building-class and road-class points respectively [11], or (b) using LiDAR elevation data [29].



(a) (b) (c)

Figure FC4.12: Augmented semantic classification of area 1 of Vaihingen dataset.

However, the methods which use bounding polygons solely depend on accuracy of classification, and other methods which superimpose imagery and point cloud data face various problems [37]. It is challenging to combine the feature descriptors from data collected using two different modes. Hence, our idea is to preserve and use the line-type points computed from a single mode, i.e., the point cloud itself. Additionally, the geometric features, computed as described in Section 4.2, are relevant local feature descriptors computed from a robust local RF, which makes the case for preserving these specific features. Figure FC4.12 shows augmented semantic classification by combining structural and semantic classification. Figure FC4.13 shows the result of augmented semantic classification for area 1, area 2 and area 3 of Vaihingen dataset. We can clearly see in Figure FC4.13 that (line,building) shows the boundary of the roof of the building, and (line,asphalt ground) shows the road bank boundary.

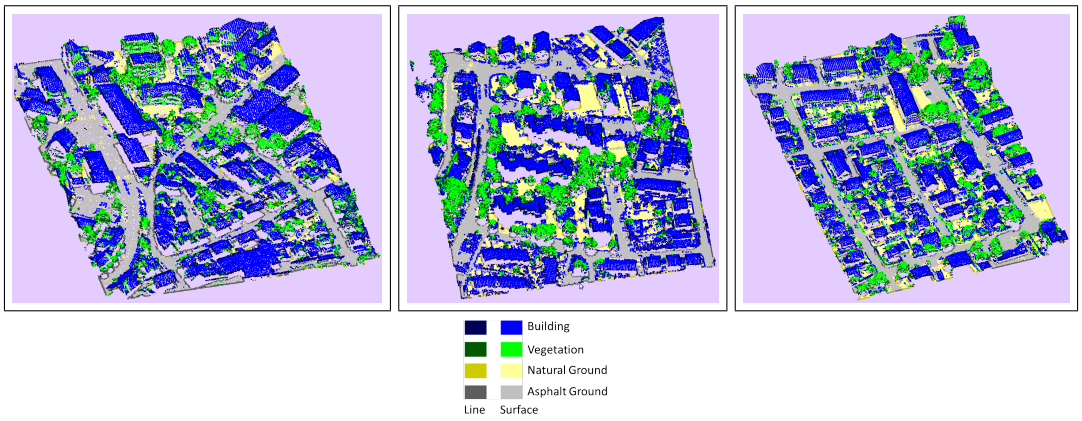


Figure FC4.13: Results of our augmented semantic classification of Areas 1, 2, and 3 (left-to-right) of the Vaihingen dataset.

CHAPTER 5

VISUAL ANALYTIC FRAMEWORK

"*Visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces*", says Thomas and Cook [38]. Visual analytics enables us to discover the embedded information inside the data, represent knowledge, interact with data and make decisions. In other words, visual analytics involves data representation, exploration, analysis, interaction, perception and decision making. Therefore, it becomes necessary to represent the data in useful form and make visual interfaces user friendly and interactive. In this chapter, we will discuss the design requirements for our visual analytic framework and its implementation.

5.1 Visual Analytic Tool

Our visual analytic (VA) framework enables user interactivity to do data mining, visualization, as well as selection of subsets of datasets for further exploration and analysis. In our framework, user can perform both the structural and semantic classification in an interactive manner and can combine both of them to get geometric-aware semantic labels. We get the following labels: $\{$(line, building), (line, vegetation), (line, natural ground), (line, asphalt ground), (surface, building), (surface, vegetation), (surface, natural ground), (surface, asphalt ground$\}$. The algorithm is already discussed in Chapter 4 and here, we will discuss the design requirement for our framework. Our proposed

framework has the following design requirements:

D1. **A user** must be provided with information on the various feature descriptors to make decisions for the interactive hierarchical clustering.

D2. The user must be able to change parameters for both classifications, and be able to visualize the results interactively, so that an optimum combination of parameters can be arrived for any new dataset.

D3. The user must be able to explore meaningful subsets of the point cloud. Meaningful subset means points in the same class or cluster given by the structural or semantic classification.

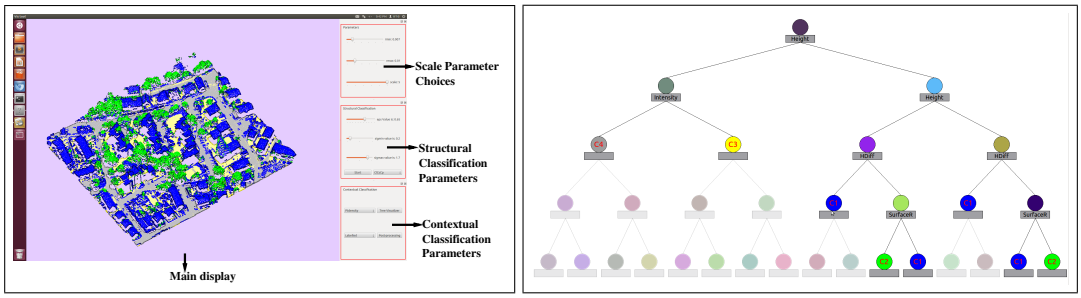


Figure FC5.1: [Left Image] The main window of our Graphical User Interface (GUI) for our visual analytic framework, showing main display on the left, and widget controls on the right, for scale parameters, structural classification and semantic classification (top-to-bottom). [Right Image] GUI for our *tree visualizer* for hierarchical clustering. In the active leaf nodes of our *tree visualizer*, (which are not dimmed/transparent), the blue ones indicate building, yellow natural ground, green vegetation, and gray asphalt ground. The points belonging to different clusters of the same class (e.g. blue clusters) get labeled the same (e.g. building).

We have implemented a visual analytic tool, as per the design requirements (D1-D3), which enables visualization of augmented semantic classification of airborne LiDAR point cloud data. We have done parallel implementation for structural classification which is discussed in Section 5.2. In our implementation, heatmaps of feature descriptors (Section 5.1.1) cater to D1; the *tree visualizer* for hierarchical divisive clustering (Section 5.1.2) cater to D2; class visualization to focus on points in selected

classes cater to D3. The main window of our GUI and Tree Visualizer are shown in Figure FC5.1. Our visual analytic tool comprises following things:

- Main display to visualize point clouds and feature descriptors.
- Tree visualizer.
- A group of widgets for multi-scale setting.
- Drop-down menu for heatmaps of feature descriptors.
- Drop-down menu for visualizing subsets of point cloud belonging to specific structural classes.
- A tab for opening files.

5.1.1 Heatmaps of Feature Descriptors

We provide the heatmaps of various feature descriptors, as shown in Figure FC4.2. In addition to the feature descriptors given in Section 4.3.1, the heatmaps for C_l, C_s, C_p (saliency values of structural features) are provided. A user can look at the heatmaps of chosen clusters in the active nodes and choose feature descriptor as clustering parameter, if the heatmap shows strong clustering tendency in the chosen feature descriptor(s). Summarizing, the heatmaps allows the user to make informed decisions on the clustering criteria in the hierarchical divisive clustering and thus, our proposed interactive hierarchical clustering is data-driven.

For data visualization, scale of display is an important aspect of visualization engine. In our work, we have not tested the scale of our visualization engine in terms of land measurements. We have tested our results in terms of the size of the point cloud. Due to limitation of the GPU memory (NVIDIA GeForce GTX480), the maximum size of points which we have currently used is 2,33,782 and the average point density of our

studied data is 4 Pts/m^2. We expect the industry-standard of 1km by 1km tiles can be handled using our system owing to its parallel implementation. Our classifications are computed real-time upon user selection of parameters for the classification. However, the computation of feature descriptors has not been parallelized, upon which the pre-processing time can reduce drastically, which is currently around 3 minutes for Area 1.

5.1.2 Tree Visualizer

We propose the *tree visualizer* as shown in right image of the Figure FC5.1, which is the visualization of the divisive clustering hierarchy as a k-ary tree, with a predetermined or permissible number of levels N_l. N_l can cause the visual clutter in the tree visualizer based on the value of k, and the optimal number of iterations in clusters required to refine the clustering. The nodes represent clusters and thus, the subset of points in the cloud that belong to the cluster. The visualizer displays all the nodes in N_l levels, where the root node represents the point cloud itself. Salient features of our *tree visualizer* are:

- **Node color**: Except for the root node, the default color of all the nodes is initially set to semi-transparent mode to indicate "inactive mode", until a cluster given by a clustering operation on its parent occupies the node. **Upon activation, a node is given a random color, which will be the color of the points belonging to the corresponding cluster, when visualized initially on the main display**. A option has been provided to a user to change color scheme of the cluster. Once clustering is done, a user can freeze/save the cluster of tree so that further deletion/clustering is not possible and assign the color to different classes as per standard color scheme.

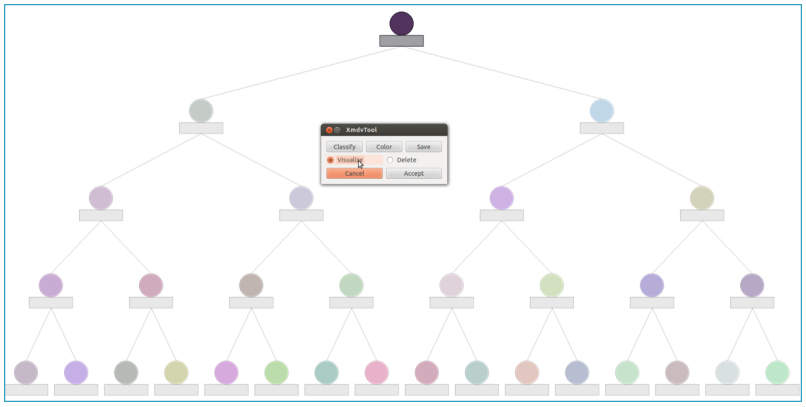


Figure FC5.2: Tree-Visualizer display to visualize the cluster.

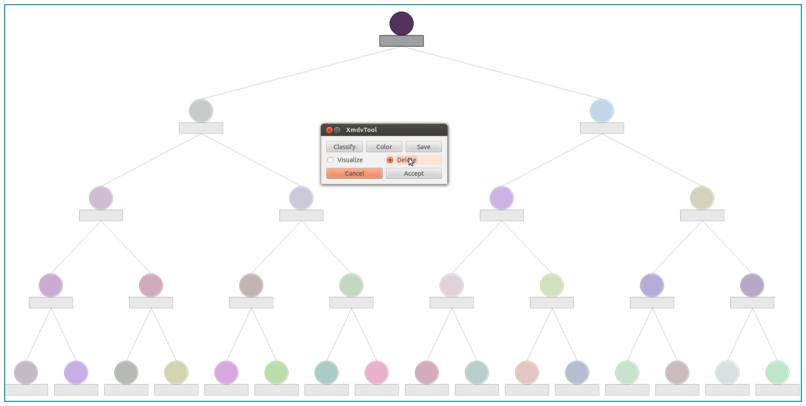


Figure FC5.3: Tree-Visualizer display to delete the cluster.

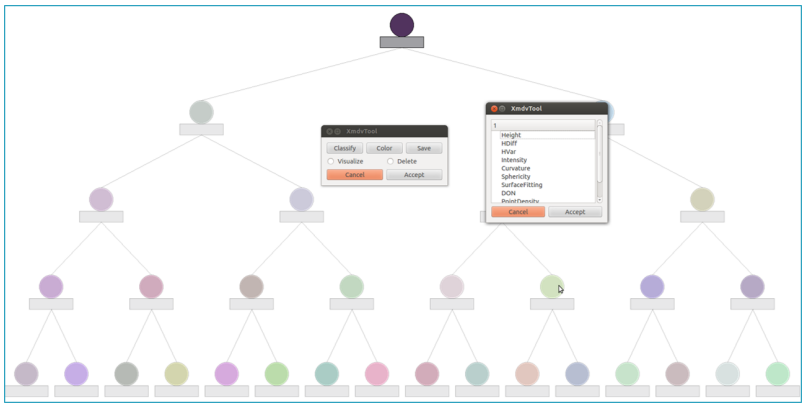


Figure FC5.4: Tree-Visualizer display to select clustering parameter.

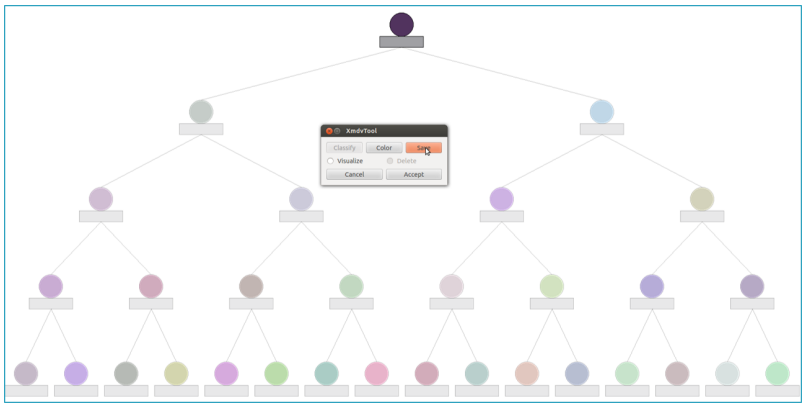


Figure FC5.5: Tree-Visualizer display to save the clustering of a subtree.

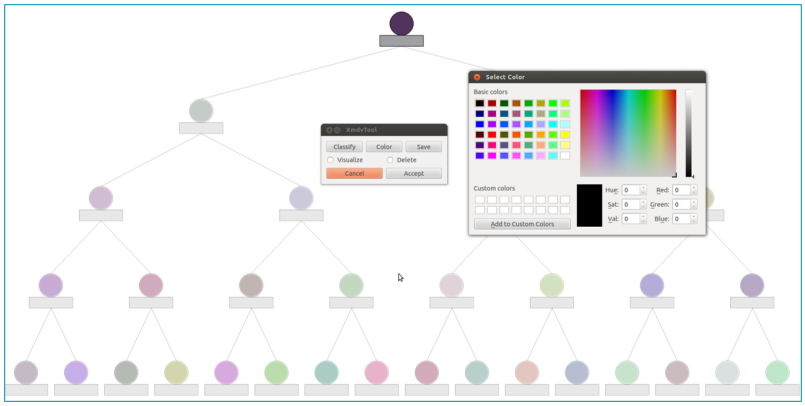


Figure FC5.6: Tree-Visualizer display to change the color of the points belonging to the same node (cluster).

- **GUI Support**: There are various options provided in the tree-visualizer tool. On clicking on an "active" node in tree visualizer, a small window pops up which includes following options:

  W1. To visualize the clusters corresponding to all the current active leaf nodes of the chosen subtree in the main display as shown in the Figure FC5.2 .

  W2. To delete the clusters in the subtree (Figure FC5.3), thus the node alone remains active and the rest of the subtree becomes inactive.

  W3. To select the subset of parameters for EM clustering from the drop-down menu of feature descriptors as shown in the Figure FC5.4.

  W4. To freeze the clusters of the subtree, thus preventing any further deletion of the clustering as shown in the Figure FC5.5.

  W5. To change the color of an active node, which resets the color of the points belonging to the cluster in the initial display on the main display as shown in the Figure FC5.6.

We have implemented tree visualizer for a binary tree with $N_l = 5$. The tree visualizer enables the users to create different models for the trees by choosing different sets of parameters for the hierarchical cluster. We show our results from two such trees, created from and applied to the same dataset as shown in the Figure FC5.7. The active nodes of the tree visualizer need not form a balanced tree and a user can continue activating nodes until **a user is satisfied with clustering**.

5.1.3 Class Visualization

The W1 widget allows the user to select specific clusters or classes obtained in the semantic and the structural classifications, respectively. Visualizing selected subsets of points helps the user to (a) make informed decision on the parameter to choose for the

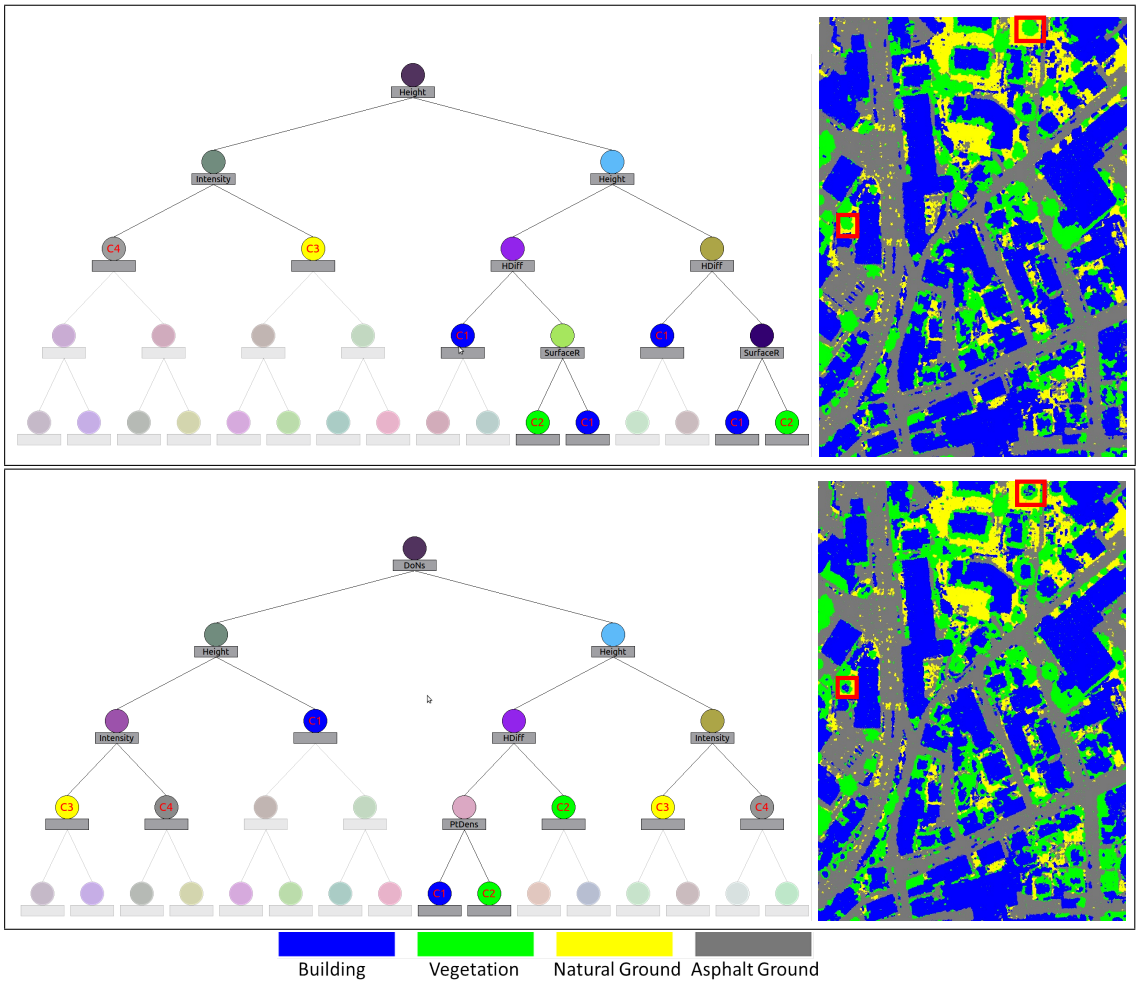


Figure FC5.7: Two possible clustering hierarchy for Area 1 of Vaihingen dataset from our tree visualizer – (left) the models on the GUI for our tree visualizer, and (right) the corresponding semantic classification results. There are subtle differences between the results; e.g. red boxes show the areas where hierarchy in the top tree-visualizer classified vegetation better than the one in the bottom tree-visualizer. In the top hierarchy, ground and non-ground points are clustered using height at the root level; while in the bottom, difference of normals (DoNs) is used to differentiate between planar and non-planar regions at the root level.

divisive EM clustering of that specific subtree for semantic classification, and (b) to visually debug to check the correctness of the implementation of the classification, in the case of structural classes.

5.2 Implementation

The implementation of our proposed method and framework includes the following components: (a) pre-processing of the data for classification (Section 5.2.1), (b) augmented semantic classification (Section 5.2.2), and (c) the visual analytic tool for classification and visualization (Section 5.1).

5.2.1 Pre-processing

Conventionally, either offline processing or high performance implementation are sought to increase the efficiency of processing large-scale datasets. We have used a combination of GPGPU (general purpose GPU) computing using CUDA [39] and offline implementation for the preprocessing step, which comprises of the following tasks:

1. **Height Correction**: We have performed height correction to obtain normalized height using an offline implementation. The elevation value given for each point in the airborne LiDAR data is an **orthometric height**, and not a normalized one. Orthometric height refers to an elevation that is measured relative to Geoid (geoid is a model of global mean level). For normalized height, we first convert the sloppy terrain into flat terrain and then compute normalized height which is the height of airborne LiDAR point above the flat ground. Normalized height is a very important feature descriptor as described in Section 4.3.1. We have computed the normalized height using LAStools [40].

2. **Octree Construction**: Data in .las file format is not spatially indexed, which implies that searching for local neighborhood of each point is time-consuming. Hence, we have used hierarchical spatial partitioning using an octree data structure for efficient search. We have done parallel implementation for octree construction using PCL library [41].

3. **Denoising**: LiDAR point clouds contain noise due to the instrumental error and atmosphere in which data has been collected. Generally, noise occurs due to reflection from irrelevant objects and under-sampling of the measurements corresponding to smaller objects. Noise in 3D LiDAR point cloud can be present in form of outliers or may corrupt the intensity. In our work, we have not considered the noise occurs in intensity of 3D point cloud. We have denoised the data by removing the outliers and it is implemented using PCL and CUDA. There are different outlier removal filters exist in literature. We have used conditional removal filter as it is computationally less expensive in comparison to other filters. It is known that statistical outlier removal outperforms other methods in terms of accuracy [8]. Conditional removal filter is a good trade-off between accuracy and speed, as the latter is required for interactive applications.

5.2.2 Augmented Semantic Segmentation

To obtain augmented semantic classification of the point cloud, we perform two classifications: (a) structural classification (Section 5.2.2.1) to get labels (line, surface) and (b) semantic classification (Section 5.2.2.2) to get labels (building, vegetation, asphalt ground, natural ground). The augmented semantic labels for the points are tuples of the two class labels from (a) and (b).

5.2.2.1 Structural Classification

We have implemented the structural classification as given in [8]. Since multi-scale implementation is not usually preferred owing to being time-consuming [9], we alleviate the problem by exploring the data parallel paradigm of the algorithm proposed by Keller et al. We have done the parallel implementation as discussed in Section 5.2.3. We have scaled the point cloud to be contained in a canonical volume (cube of size 2,

centered at (0,0,0)), and we have used 5 scales for radii in the range $[0.006, 0.01]$ for local spherical neighborhood. Thus, our structural classification labels the points as line- or surface- type features, in real time.

5.2.2.2 Semantic Classification

Semantic classification is a two-stage process, namely, (a) hierarchical divisive clustering, (b) region-growing.

Stage 1: Hierarchical Divisive Clustering We have used the multi-scale approach with the same scales used for structural classification, as in Section 5.2.2.1, for computation of local feature descriptors (Section 4.3.1) in case of cylindrical [22] as well as spherical neighborhoods, and average values across multiple scales. The values of feature descriptors for each scale are computed using CGAL [42] and PCL [41]. In our algorithm, we have used Hierarchical Expectation-Maximization (HEM) algorithm for clustering. We have used single feature descriptor as the parameter for clustering to split a node, even though we can give a subset of feature descriptors as parameters. This is because our heatmaps give trends for a single feature descriptor and in this work, we have not analyzed the correlation for more than one feature descriptor, to make an informed decision. It is known that using EM is not the best clustering solution for complex models where there is no inherent Gaussian mixtures present in the model in the parameter space. However, it has worked in our experiments as it purely functions as a clustering method in cases where clear clusters can be observed to exist without well-defined boundaries, as observed in the corresponding heatmaps (Section 5.1.1) in 1-dimensional subspaces of the parameter space (as we have used only one feature descriptor as clustering parameter in each iteration).

We have used the tree-visualizer tool (Section 5.1.2) to determine the parameters

for Hierarchical Expectation Maximization (HEM) clustering and to perform the clustering. Then, we have visually determined equivalence of some of the clusters obtained from different subtrees in the clustering hierarchy and merge them if it is required. For example, clusters C_1 in Figure FC5.7 are labeled as building.

Stage 2: Region Growing We have performed an intra-class region-growing [19, 41] for points belonging to the building and vegetation classes, exclusively. This gives us segments, also referred to as features, which essentially preserves the spatial context. For larger segments, determined using a threshold, we have performed a label correction by: (a) determining the class strength for "building" and "vegetation" in the segment, computed as number of points belonging to the class, and (b) assigning the stronger class label to the entire segment.

5.2.3 Parallel Implementation

The structural feature extraction module is computationally intensive and time consuming. In order to make the application interactive and real-time, we have proposed a parallel implementation for octree removal and point classification module of algorithm proposed by Keller et al. [8] using GPU computing techniques [43]. The outlier removal module and structural feature classification are slow, owing to their computationally intensive nature. However, since the operations related to each point are dependent on their local neighborhood and are independent of the others, they are embarrassingly parallel. Hence, we have exploited the data-parallel paradigm of these modules and achieved parallel implementation of the algorithm using PCL [41], and CUDA [39] library.

5.2.3.1 Outlier Removal

Outlier removal computation is reduced by implementing the algorithm using a data-parallel paradigm. Octree data structure is used for structural decomposition of data and we have built the octree in parallel using PCL library [41]. Search operations like k-nearest and r-nearest can be performed in a fast way in octree data structure. We have used conditional outlier removal filter for denoising. In the conditional outlier removal filter, density, i.e. number of points, within a sphere of an user defined radius r is measured, and if density is less than a particular threshold, then the point is considered as an outlier. $r-$nearest neighbors search and density measurement for a voxel is independent of the other voxels, thus making it embarrassingly parallel. Skeleton code for parallel implementation of outlier removal module is given in Algorithm 1.

Algorithm 1 Skeleton code of parallel implementation for outlier removal module

1: Built an octree data structure (*in parallel*) to store the data using PCL.
2: Perform batch processing to search $r-$nearest neighbors for all 3D points (*in parallel*) using PCL.
3: Copy all 3D points and their $r-$nearest neighbors from CPU memory to GPU memory.
4: Launch a cuda kernel with N threads, where N is equal to total number of 3D points.
5: **for** $k = 0 \rightarrow N$ *in parallel* **do**
6: If number of $r-$nearest neighbor of a point of is less than a threshold, mark the point as outlier.
7: **end for**
8: Copy the indices of outliers from GPU memory to CPU memory.
9: Delete the outliers from octree.

5.2.3.2 Point Classification

For each point in the point cloud, the likelihood of the local neighborhood of the point corresponding to the shape classes, namely, spherical, cylindrical, and disc-like neighborhood are computed. This is a stochastic point classification technique which classifies each point in the point cloud based on its specific structural properties, as discussed in Chapter 4. Stochastic point classification involves finding a spherical neigh-

Algorithm 2 Skeleton code of parallel implementation for stochastic point classification module

```
1: Built an octree data structure (in parallel) to store the data using PCL.
2: for j = 0 → scale do
3:    Perform batch processing to search r−nearest neighbors for all 3D points (in parallel)
      using PCL.
4:    Copy all 3D points and their r−nearest neighbors from CPU memory to GPU memory.
5:    Launch a cuda kernel with N threads, where N = total number of points in the data.
6:    for k = 0 → N in parallel do
7:       Build a covariance matrix for a point.
8:       Compute eigenvalues and eigenvectors of a covariance matrix for a point.
9:    end for
10:   for k = 0 → N in parallel do
11:      Compute the probability for a point as given in Equations Eqn 5.1, Eqn 5.2 and
         Eqn 5.3.
12:   end for
13:   Copy the result from GPU memory to CPU memory.
14: end for
15: for k = 0 → N do
16:   Average the probability for a point over the scale.
17: end for
```

borhood of each point and constructing the covariance matrix from the spherical neighborhood. Spectral decomposition of covariance matrix is performed to compute the eigenvalues and eigenvectors. Let λ_2, λ_1 and λ_0 be eigenvalues of the covariance matrix such that $\lambda_2 \geq \lambda_1 \geq \lambda_0$. Shape of the local structure is estimated by comparing the eigenvalues as given in Equations Eqn 5.1, Eqn 5.2 and Eqn 5.3.

$$L_{cp}(p) = \lambda_0 \geq \varepsilon\lambda_2 \quad \text{(Eqn 5.1)}$$

$$L_p(p) = \lambda_1 < \varepsilon\lambda_2 \quad \text{(Eqn 5.2)}$$

$$L_d(p) = \lambda_0 < \varepsilon\lambda_2 \quad \text{(Eqn 5.3)}$$

Equation Eqn 5.1, Eqn 5.2 and Eqn 5.3 give likelihood for a point having spherical, cylindrical and disc-type local structure respectively. Stochastic point classification has been done across multiple scales, where each scale is determined by the radius

of the point for local neighborhood. The values across scales are averaged to obtain the likelihood of the point falling into different feature classes. The nearest neighbors search and computation of eigenvalues are embarrassingly parallel tasks, which has been exploited in our parallel implementation of the algorithm proposed by Keller et al. [8]. Parallel implementation has been done for following tasks:

- Octree construction **[T1]**
- $r-$nearest neighbors search **[T2]**
- Probability measurement to classify each point into different classes **[T3]**

We have used PCL for tasks **T1** and **T2**. Batch processing is performed to compute probability of each point in the data to fall into different feature classes, and skeleton code of parallel implementation for probability measurement is given in Algorithm 2.

5.2.4 Performance

We have done serial implementation of algorithm proposed by Keller et al. [8] and compared it's performance with our proposed parallel implementation. All the experiments have been performed on Intel Xeon (R) processor at 3.2 GHz quad-core , 8 GB RAM with NVIDIA GeForce GTX480. We have tested the algorithm for the following las files [44].

Dataset(las files)		Timing measurement in CPU seconds	
	Size	Serial Implementation	Parallel Implementation
test2 [44]	11,765	1.76	0.06
test1 [44]	33,703	5.01	0.21
galvestone [44]	99,600	37.26	1.11
mscstsc [44]	1,60,101	71.01	2.01
spring2 [44]	2,01,474	101.31	4.19
srsota [44]	3,86,530	222.57	4.07
N144835 [44]	4,31,276	321.45	8.3
N440375 [44]	4,97,536	375.21	8.47
autzen-stadium [44]	6,93,895	399.22	16.21

Table TC5.1: Timing measurements in CPU seconds for serial and parallel implementation

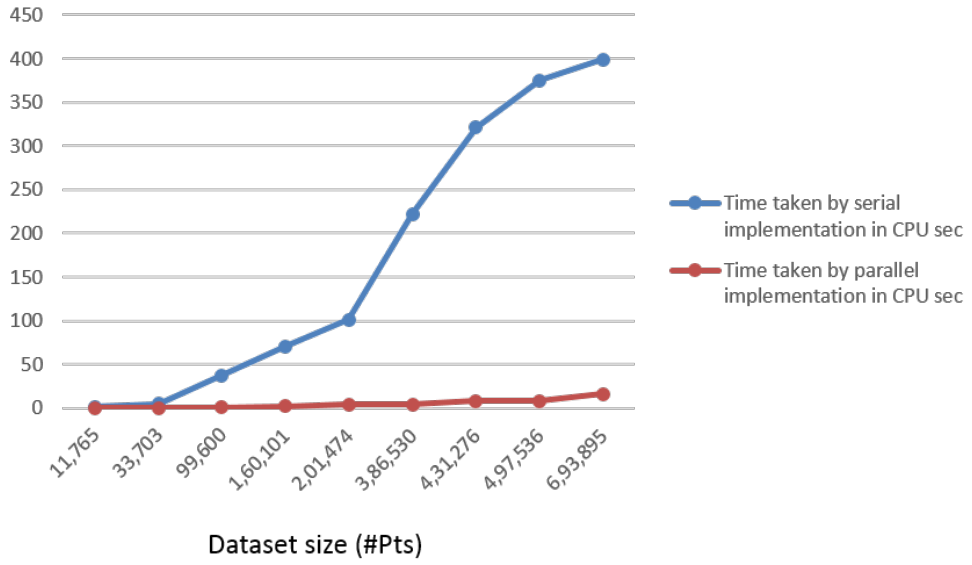


Figure FC5.8: Comparison of computation time in serial and parallel implementation. Here, the time taken for the following tasks make the difference in performance between serial and parallel implementation : (a) outlier removal (b) stochastic point classification.

Table TC5.1 shows the timing measurement in CPU seconds for serial and parallel implementation of algorithm proposed by Keller et al. We can observe that the time

taken by parallel implementation is relatively much lesser than serial implementation, as expected, as shown in the Figure FC5.8.

CHAPTER 6

STRUCTURAL CLASSIFICATION

Feature detection is a key operation in the process of surface reconstruction, shape detection, registration of point clouds, finding deformation in time-varying datasets, etc. The definition of feature is application-specific and hence, is subjective. Features are generally defined as entities which help the user to gain meaningful insights about the data. Additionally, features may be described as either a subset of data or derived data, where the strength of a feature is measured in terms of its persistence across multiple scales, time-steps, and/or other attributes which are the basis for multiple series of the same dataset. We have focused on structural feature detection in unstructured point cloud, since our use-case arises from point clouds procured from LiDAR scanning devices. However, our results are applicable to any unstructured point cloud data. Our goal is to detect following features: 1. point-, 2. line-, and 3. surface- type feature points in an unstructured point cloud.

As discussed in Chapter 4, we have explored second order symmetric semi-definite tensor based methods for structural classification, due to its higher accuracy and less computation cost as compared to other conventionally used methods. Across local tensor based methods, different types of second-order symmetric tensors have been used to encode the local geometry of a point, such as structure tensor, voting tensor, and anisotropic diffusion tensor. Keller et al. [8] have used covariance matrix as structure

tensor for feature extraction. However their algorithm requires user-defined parameters and we have observed that adjusting these parameters for different datasets is a difficult task.

Guy and Medioni [45] have proposed a tensor voting method to detect and classify the feature points in structured point cloud. The voting scheme of tensor voting uses the proximity and continuity principle of Gestalt psychology to propagate the votes. Park et al. [6] have used this method in unstructured data to detect the feature points. However, tensor voting has the limitation of not detecting accurately weak features from noisy data [46]. Wang et al. have introduced anisotropic diffusion tensor based voting scheme for triangle meshes which uses diffusion tensor derived from the normal voting tensor [46]. The advantage of using anisotropic diffusion voting tensor is that diffusion takes into consideration the global behavior of the rich local geometry data, which is recorded in the form of voting tensor.

We propose to use anisotropic diffusion tensor for structural feature extraction in unstructured point cloud. Our approach is an extension of the method proposed by Wang et al. [46] for triangle meshes to the unstructured point cloud. We introduce two different diffusion tensors, generated using different starting tensors for extraction of structural features in unstructured point cloud. Our aim is to compare the results of structural classification obtained using our proposed diffusion tensors with structure and voting tensor.

While comparing the feature classification results obtained from different geometrical tensors, **we have found that there is no well defined metric in the literature to compare the efficacy of feature detection methods**. In the feature detection methods of our interest [8] and [46], the point is classified based on the shape of the local point neighborhood of a point and the likelihood of a point falling into different feature classes. Thus, we propose to use: (1) likelihood of a point falling into different feature

classes and (2) shape of the local point neighborhood of a point as metrics for comparing the results of tensor based feature detection methods. We have used visualization techniques to study these parameters. Likelihood of a point falling into different feature classes, also knowns as saliency values, can be studied using RGB color spectrum (saliency map). Shape of local point neighborhood of a point can be studied by visualizing the shape and orientation of the second order tensor defining the local neighborhood, as it encodes underlying geometry of a point in the data.

From the existing visualization methods for second order tensor fields, topology-based and glyph-based methods have been found to be effective to study the shape and orientation of the tensor. Since voting tensors form a non-linear field, it is not possible to apply existing topology-based methods for symmetric second-order tensors, which are applicable only for linear fields. However, glyph visualization methods are applicable for both linear and non-linear tensor fields, as geometric objects (glyphs) are discrete in their direct application and are unaffected by the inherent nature of the field. Schultz and Kindlmann [47] have generalized superquadric tensor glyphs for all second-order symmetric tensors. We have studied the shape and orientation of the structure tensor, normal voting tensor, and anisotropic diffusion tensor fields by visualizing them using superquadric glyphs.

6.1 Second Order Symmetric Tensors Defining Local Geometry

Shape of the structure on a 3D manifold can be estimated by detecting the feature points. Feature point classification is essential for integrating points to generate line- and surface- type features and for 3D reconstruction of the dataset. As discussed earlier, each point in the data can be classified into point-, line- and surface- type features. The classification has been done based on eigen analysis of covariance matrix [30]. The covariance matrix can be represented as a second order symmetric tensor.

Tensors are mathematical objects which describe physical properties like stress, strain, etc. Tensors are defined by *dimension* and *order*. A formal definition may be found in [48]. In this work, we have specifically studied second-order symmetric positive semi-definite tensors in $\mathbb{R}^3$ that encode the local geometry of a point in unstructured point cloud data. The shape of the local point neighborhood can be found by using eigenvalues obtained from spectral decomposition of tensor encoding local geometry of the points. Eigenvectors and eigenvalues of a tensor are invariant with respect to the rotation of the reference frame and provides principal components of the tensors. Thus, the unique representation of tensor using its eigenvalues and eigenvectors gives a more accurate analysis of the field. In our work, we have used structure, normal voting, and anisotropic diffusion tensor as the tensor encoding local geometry of a point in the unstructured point cloud.

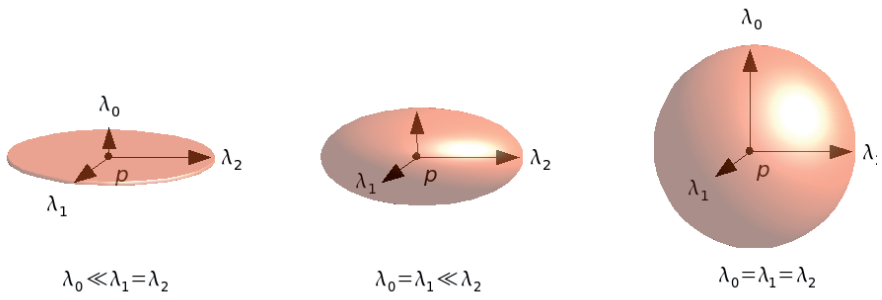
Structure Tensor : In point cloud geometry, 3D structure tensor at a point p is defined as covariance matrix. Hoppe et al. [49] have proposed formulating surface reconstruction as a total least squares problem, where tangent plane is estimated using principal component analysis (PCA). The covariance matrix at x is given by:

$$CV(x) \quad = \quad \sum_{y \in N(x)} (y - \bar{x})(y - \bar{x})^T, \qquad \text{(Eqn 6.1)}$$

where $\bar{x}$ is the centroid of the local neighborhood $N(x)$. Also referred to as correlation matrix [50], the covariance matrix is essentially constructed as an outer product of tangent vectors.

Keller et al. [8] has used covariance matrix to estimate local structure or shape at any point p by comparing the distribution of local neighborhood $N(p)$ of point p in each dimension. Let λ_0, λ_1 and λ_2 be eigenvalues of structure tensor such that $\lambda_0 \leq \lambda_1 \leq \lambda_2$. Local structure at point p will have planar or disc-like shape if $\lambda_0 \ll \lambda_1 \simeq \lambda_2$, cylindrical shape if $\lambda_0 \simeq \lambda_1 \ll \lambda_2$, and spherical shape if $\lambda_0 \simeq \lambda_1 \simeq \lambda_2$ as shown in the Figure FC6.1. Keller et a. [8] used the following classification for a given point

based on shape of the local point neighborhood of a point: point-type feature if shape of local neighborhood is spherical, line-type feature if shape of local neighborhood is cylindrical, and surface-type feature if shape of local neighborhood is disc.



(a) (b) (c)

Figure FC6.1: Stochastic method used for point classification based on shape of local neighborhood, as proposed by Keller et al. [8,51], namely, (a) planar or disc-like, (b) cylindrical, and (c) spherical neighborhood. Image courtesy: [51].

Normal Voting Tensor at any point p on a manifold is the outer product of unit vectors which span the normal space of a manifold, as given in [6, 52]. Let the normal space of a manifold be spanned by unit vector n_i in d-dimensional space $\mathbb{R}^d$, then normal tensor T at any point p on a manifold is given by:

$$
\begin{aligned}
T &= \sum_{i=1}^{d} n_i n_i^T \\
&= \sum_{i=0}^{2} \lambda_i e_i e_i^T \qquad \text{(Eqn 6.2)} \\
&= (\lambda_2 - \lambda_0) e_0 e_0^T + (\lambda_1 - \lambda_0)(e_0 e_0^T + e_1 e_1^T) \\
&\quad + \lambda_0 (e_0 e_0^T + e_1 e_1^T + e_2 e_2^T) \\
&= T_S + T_P + T_B \qquad \text{(Eqn 6.3)}
\end{aligned}
$$

Normal voting tensor at a point p can be expressed in terms of its corresponding eigenvectors and eigenvalues, obtained using its spectral decomposition, as shown in

Equations Eqn 6.2 and Eqn 6.3. It can be used to estimate the shape of local structure at a point p, and can be represented as an ellipsoid. Equation Eqn 6.2 shows the generalized form of T, in $\mathbb{R}^3$, in terms of its eigenvalues, λ_0, λ_1 and λ_2 and corresponding eigenvectors, e_0, e_1, and e_2. Eigenvalues are sorted as: $\lambda_0 \leq \lambda_1 \leq \lambda_2$. From the definition of Equation Eqn 6.3, T can be decomposed into three components: stick tensor T_S, plate tensor T_P, and ball tensor T_B. Stick, plate, and ball tensors at point p refer to the cylindrical, planar-, and spherical local structures at point p. The magnitude of each component in Equation Eqn 6.3, also known as *saliency* values, gives the likelihood of a point p falling into different feature classes.

Park et al. [6] have used tensor voting to detect features in an unstructured point cloud. Tensor voting is a three-step process. Initially each point is encoded as stick, plate, or ball tensor based on available input information. Points with normal information are encoded as stick tensors, points with tangent information are encoded as plate tensors and unoriented points are expressed as ball tensors. Afterwards, every component i.e stick, plate and ball tensors of the local point neighborhood of a point p separately cast votes to the point p which are stick, plate and ball vote respectively. The votes are defined as tensors that encode the most likely direction of the normal of the receiver, based on the voter's information in that component. Finally, the accumulated votes on the receiver are summed up which gives us normal voting tensor (NT) as given in the Equation Eqn 6.4. Then, the normal voting tensor is decomposed into stick, plate and ball tensor components. The tensor which has the highest saliency value will be dominant structure at that point.

$$
\begin{aligned}
NT(p,q) &= \sum_{q \in N(p)} (Ball_{vote}(p,q) + Stick_{vote}(p,q) + Plate_{vote}(p,q)) \\
&= \sum_{i=0}^{2} \lambda_i e_i e_i^T \\
&= (\lambda_2 - \lambda_0) e_0 e_0^T + (\lambda_1 - \lambda_0)(e_0 e_0^T + e_1 e_1^T) \\
&\quad + \lambda_0 (e_0 e_0^T + e_1 e_1^T + e_2 e_2^T) \\
&= T_S + T_P + T_B \qquad \text{(Eqn 6.4)}
\end{aligned}
$$

In our case, we can consider the LiDAR point cloud to be unoriented, Hence we initially encode each point as ball tensor. Therefore, the vote contribution of stick and plate tensor during voting process will be zero, and only ball component will cast the vote during the voting process as shown in the Figure FC6.2. The vote cast by a ball tensor, is given by a second order tensor which spans all possible normal space, and can be expressed as given in the Equation Eqn 6.5 [6, 52].

$$
Ball_{vote}(p,q) = \mu_q . \left(I_d - \frac{(q-p)(q-p)^T}{\|(q-p)(q-p)^T\|} \right) \qquad \text{(Eqn 6.5)}
$$

where the Gaussian function, $\mu_q = e^{\left(\|q-p\|^2/\sigma^2\right)}$ is the attenuation factor for the size of vote, $N(p)$ is the local neighborhood of the point p, and I_d is an identity matrix of size d. The variance of Gaussian distribution σ, can be considered as scale parameter in this case, which affects the propagation range and the size of $N(p)$. As the scale increases, more tokens can influence the candidate point due to increase in size of kernels, and smooth out the noisy data and weak features. Similarly, if it decreases, more details are preserved but more sensitive towards noisy points. It can be viewed as the regulator for smoothness as used by Pauly et al. [32].

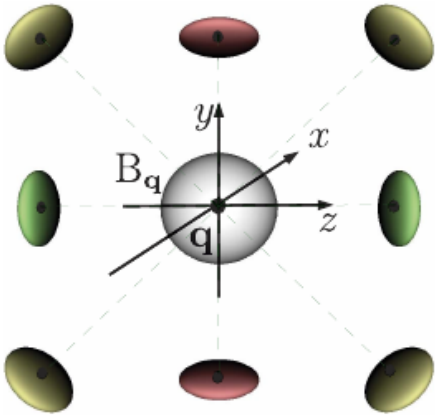


Figure FC6.2: Example of ball tensor voting. Image courtesy [53]

Anisotropic Diffusion Tensor Wang et al. [46] have introduced controlled anisotropic diffusion on normal voting tensor to exploit the advantages of local geometry tensor as well as global diffusion for triangle meshes. In this paper, diffusion process is considered in steady state and hence, temporal derivatives are zero. Therefore, diffusion process has been reduced to elliptic PDEs. Wang et al. have used diffusion tensor to encode the local geometry of the vertex and, diffusion velocity to control the direction of global diffusion. We have not explored the global diffusion process in our work as yet. We have used the initial assignment of tensor [46] to exploit the anisotropic diffusion tensor encoding the local geometry of the vertex, and we have extended it to the unstructured point cloud as explained in Section 6.4.

Wang et al. have derived the anisotropic diffusion tensor from normal voting tensor for triangle mesh. Normal voting tensor diffuses faster when voting is performed along the crossing of sharp edges, and slower when voting is done along the edges. However structural feature classification requires the tensor to diffuse slower along crossing of sharp edges and faster along edges. Therefore, normal voting tensor cannot be directly adopted as anisotropic diffusion tensor. Wang et al. have constructed diffusion tensor from normal voting tensor for triangle mesh. In [46], normal voting tensor T for triangle mesh is defined as:

$$T = \sum_{t_j \in N_t(v_i)} \mu_j n_{t_j} n_{t_j}^T, \qquad \text{(Eqn 6.6)}$$

where t_j is a triangle, $N_t(v_i)$ denotes the set of neighboring triangles of v_i. n_{t_j} denotes the normal of a triangle t_j and μ_j is the weight coefficient. Let λ_0, λ_1 and λ_2 be eigenvalues of normal voting tensor and e_0, e_1, and e_2 be eigenvectors corresponding to λ_0, λ_1 and λ_2. Anisotropic diffusion tensor D is constructed from normal voting tensor at each vertex in the triangle mesh in the [46] as:

$$D = \sum_{i=0}^{2} \widetilde{\lambda}_i e_i {e_i}^T \qquad \text{(Eqn 6.7)}$$

where $\widetilde{\lambda}_i = e^{\left(-\lambda_i/\delta_d\right)}$ for $i = 0, 1, 2$. δ_d is a diffusion parameter which is applied to eigenvalues of normal voting tensor, to control the diffusion velocity. The eigenvectors of anisotropic diffusion tensor determine the principal directions for diffusion, and eigenvalues $\widetilde{\lambda}_i$ give the strength of diffusion along these directions. Diffusion parameter δ_d controls the velocity of diffusion along the principal directions. Smaller values of δ_d imply more resistance for heat to diffuse through critical points and vice-versa. Differential behavior of δ_d, in the presence of critical points, makes the diffusion process anisotropic in nature.

Wang et al. have used principal diffusion direction to distinguish the weak features from noise, and growing or merging the curve features. The principal diffusion direction is the direction corresponding to the maximum diffusion velocity. The diffusion velocity, $v(p,e)$, from vertex p along a vector e is defined in [46] as:

$$v(p,e) = \frac{e^T D(p) e}{e^T e} \qquad \text{(Eqn 6.8)}$$

The anisotropic diffusion tensor can be thought as an ellipsoid at each vertex of the triangle mesh that encodes direction (eigenvectors e_i) and diffusion velocity v_i. Thus, the anisotropic diffusion tensor preserves the orientation of the tensor field as it eigenvectors are same as that of normal voting tensor. But, it distorts its shape according to the diffusion velocities.

6.2 Visual Analysis of Tensors

Our aim is to compare the output of classification obtained from structural feature detection methods. Since the method in [8] can be redefined to be based on the second order tensor. Our proposed method uses second order tensor field for feature detection, we propose to use saliency values, shape, and orientation of the tensor as metrics to compare the outcomes. We have used visualization techniques to study saliency values, and shape and orientation of the tensor. Advantages of visualization techniques is that they allow faster analysis and exploration of data, and helps in visually finding hidden patterns and understanding interesting features. We have used visualization techniques which can be discretely applied on points: (a) color-mapping to study the global trends in the saliency , in the case of structure tensor, voting tensor, and anisotropic diffusion tensor and (b) superquadric tensor glyphs for studying orientation and shape of feature descriptors. Thus, summarizing we have used two methods for comparing classification outcomes of local geometry-based tensor fields: 1) saliency value color maps and, 2) glyph visualization.

6.2.1 Saliency Value

Saliency value at a point p gives the likelihood of p falling into different feature classes: line-, surface-, and point- type feature points. We have used Cl, Cp and Cs as saliency value. Cl, Cs and Cp defines the likelihood of a tensor at a point p belonging to

each of the three shape classes [54] - cylindrical, spherical and disc , which correspond to the three feature classes of point clouds. If λ_0, λ_1 and λ_2 be corresponding eigenvalues of tensor (structure, normal and anisotropic diffusion tensor) such that $\lambda_0 \leq \lambda_1 \leq \lambda_2$, saliency values for different feature class can be defined as follows [54], [55]:

- Saliency value, C_l, for line-type feature: $\frac{(\lambda_2-\lambda_1)}{\lambda_2+\lambda_1+\lambda_0}$.
- Saliency value, C_p, for surface-type feature : $\frac{2*(\lambda_1-\lambda_0)}{\lambda_2+\lambda_1+\lambda_0}$.
- Saliency value, C_s for point-type feature : $\frac{3*\lambda_0}{\lambda_2+\lambda_1+\lambda_0}$.

.

Since C_l, C_p and C_s gives the likelihood of a point falling into line-, surface-, and point- type feature classes, sum of C_l, C_p and C_s should be equal to unity. Therefore, scaling factor 2 and 3 has been inserted for cp and cs such that $cl + cs + cp = 1$ and each of them independently $\in$ [0 1] [55].

We have mapped saliency value of each component of tensor at a point p to RGB model. The saliency-based color for a point is given by r, g, and b channel in the RGB model, which are computed using eigenvalues of the tensor at that point. We have mapped C_l to red channel, C_p to green channel, and C_s to blue channel to create saliency map.

6.2.2 Glyph Visualization

Glyph visualization is used to study the shape and orientation of tensors at each point. Generalized tensor glyphs, proposed by Schultz and Kindlmann [47], have been effective in depicting second-order symmetric positive semidefinite tensors. The shape and orientation of the glyphs are derived from the spectral decomposition of the tensors. Superquadric defines a family of shapes which resembles the shapes of ellip-

soid, superellipsoids, superhyperbolloids and supertolloids. The explicit equation of superquadric is

$$q = \begin{pmatrix} \cos(\theta)^{\alpha} \sin(\phi)^{\beta} \\ \sin(\theta)^{\alpha} \sin(\phi)^{\beta} \\ \cos(\phi)^{\beta} \end{pmatrix} \quad \text{(Eqn 6.9)}$$

where $0 \leq \phi \leq \pi$, $0 \leq \theta \leq \pi$ and, α and β control the shape of superquadic glyph.

Symmetry, continuity and disambiguity are general guiding principles for tensor visualization [47]. Disambiguity means that glyph should be able to distinguish between different tensors. Symmetry means that glyphs should have the same symmetry as the the underlying tensor. Superquadric glyphs should have mirror symmetry, as e_i and $-e_i$ are equivalent vectors of tensor due to lack of orientation. Continuity means that there should not be drastic changes in visualization due to slight perturbation in the data. Superquadric glyphs have a neat characteristic of disambiguating the glyph in any view direction, and they preserve the symmetric nature of the tensors as well as the continuity in the field [47]. The superquadric geometry effectively showcases the shape and orientation of a tensor, in comparison to other glyph techniques. Hence, we have used the superquadric glyphs to compare various geometrical tensors used as feature descriptors of the point cloud data.

6.3 Comparisons Between Structure and Normal Voting Tensor

In this section, we compare structure and normal voting tensor. Structure tensor and normal voting tensor are defined by Equations **??** and Eqn 6.2 respectively. Both have similar formulation and $\nabla I \nabla I^T$ in structure tensor plays similar role as nn^T in normal voting tensor. Gaussian convolution in structure tensor can be seen as voting process, where we define "*vote*" to be the strength given by the Gaussian function

(Equations **??**). Despite the similarity in formulation, there are differences in its construction such as the following:

- Structure tensor is an outer product of tangent vectors while normal voting tensor is an outer product of normal vectors. Since structure tensor is constructed using outer product of tangent vectors which is contravariant, it has 2 contravariant indices and 0 covariant indices. Similarly, normal voting tensor is constructed using outer product of normal vectors which is covariant, and hence it has 2 covariant indices and 0 contravariant indices. Thus, structure tensor and normal voting tensor are (2,0) and (0,2) tensors, respectively.

- Structure tensor does not use available orientation information like tangent or normal direction. It uses only spatial positions x, y, z. On the other hand, voting tensor encodes the underlying geometry of a point as a tensor based on available input information, spatial positions x, y, z, and orientation information like tangent or normal direction if it is available.

- Structure tensor estimates shape of local structure as cylindrical, planar or spherical. Similarly, normal voting tensor estimates shape of local structure as stick tensor, plate tensor or ball tensor. We have found that feature classes of normal voting and structure tensor can be mapped to each other. In the normal voting tensor, ball, stick and plate tensor represent the point-, line- and surface- type feature point respectively. Similarly, in case of structure tensor spherical, cylindrical and disc shape of the local point neighborhood indicate point-, line- and surface- type feature point respectively. Therefore, we can say that spherical, cylindrical and disc shape local structure in the structure tensor can be mapped to the ball, stick and plate tensor in normal voting tensor.

6.4 Our Proposed Diffusion Tensor

Since we are interested in performing structural classification in LiDAR point cloud, we have used the graph theory-based feature detection method proposed by Keller et al. [8]. However owing to its shortcomings related to computationally expensive routines as well as requirement for several user-controlled parameters, we have explored local tensor-based methods, which have shown promising results for triangle meshes in [46]. While Keller et al. have mentioned that user controllable parameters is a requirement, we have found fine-tuning these parameters for each dataset is difficult.

For triangle meshes, Wang et al. [46] have used a structure tensor, which is a weighted covariance matrix computed from neighboring triangles, as the normal voting tensor, on which diffusion is applied on, as given in Equation Eqn 6.8. We propose two different computations of diffusion tensor for point cloud data, where the initial second order symmetric tensor, T, is (a) structure tensor (ST) [8], and (b) normal voting tensor (NT) [6]. The anisotropic diffusion tensors, which we generate by using ST and NT as initial structure tensor, will be hereafter referred to as, *SDT* and *NDT*, respectively.

Computation of diffusion tensor (DT) is derived from an initial second order symmetric semi-definite tensor, T. Let λ_0, λ_1 and λ_2 be eigenvalues of T corresponding to eigenvectors e_0, e_1 and e_2 respectively, and δ_d is a diffusion parameter to control the diffusion velocity. Diffusion parameter δ_d is applied to eigenvalues of ST, $\widetilde{\lambda}_i = e^{\left(-\lambda_i/\delta_d\right)}$ for $i = 0, 1, 2$. Diffusion tensor and diffusion velocity are computed using the Equation Eqn 6.7 and Eqn 6.8 respectively. Let v_0, v_1 and v_2 be diffusion velocities corresponding to e_0, e_1 and e_2 respectively. Our diffusion tensor, DT, is defined as:

$$DT(T) = \sum_{i=0}^{2} v_i e_i {e_i}^T \qquad \text{(Eqn 6.10)}$$

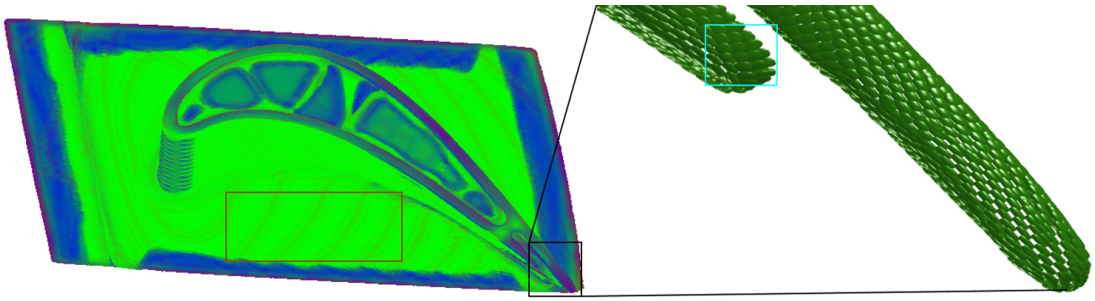
Diffusion velocity, v_i, in Equation Eqn 6.10 defines the length of the eigenvector e_i projected onto ellipsoid. Therefore, we can say that v_i are eigenvalues of anisotropic diffusion tensor (DT), and used for saliency map and glyph visualization. Structure diffusion tensor, SDT, is derived from structure tensor (ST), thus $SDT = DT(ST)$ from Equation Eqn 6.10. Normal voting diffusion tensor (NDT) is derived from normal voting tensor (NT), thus $NDT = DT(NT)$ from Equation Eqn 6.10.

6.5 Experiments

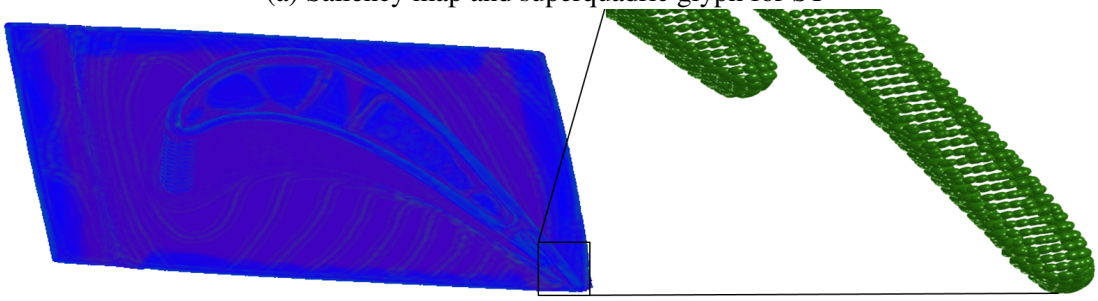
We have used saliency map to visualize likelihood of a point falling into different feature classes and superquadric glyph visualization to study the shape and orientation of tensor. Using saliency map and glyph visualization, we have compared the feature classification results obtained from ST, NT, SDT and NDT tensors. We have tested output of different tensors on scanned point cloud data like dragon, blade and child models in ply format [56], and Autzen stadium [44] and area 1 of Vaihingen datasets [1] in las format. In case of diffusion tensor, we have kept diffusion parameter δ_d constant and it is equal to 0.08 for all the experiments. In saliency map, saliency values for line-, surface- and point- type features, which are normalized and fall in the interval $[0,1]$, are mapped to red, green, and blue channels respectively.

6.5.1 Saliency Map

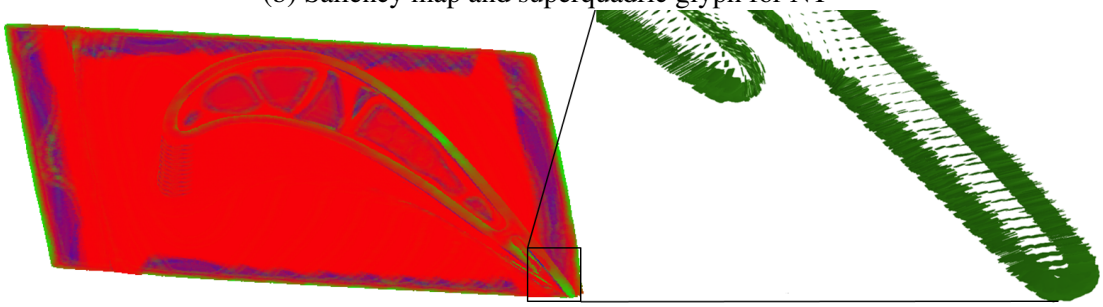
We observe that introducing anisotropic diffusion to structure tensor (i.e., ST) gives SDT, and causes drastic changes in the nature of the second order tensor ST. These changes include (a) a point more likely to be a point-type (blue) or line-type (red) feature changes to surface-type (green) feature; and (b) surface-type (green) feature change to line-type (red) feature. Since the diffusion process changes the classification and makes the classes less prominent, SDT is not a desired tensor for structural feature detection



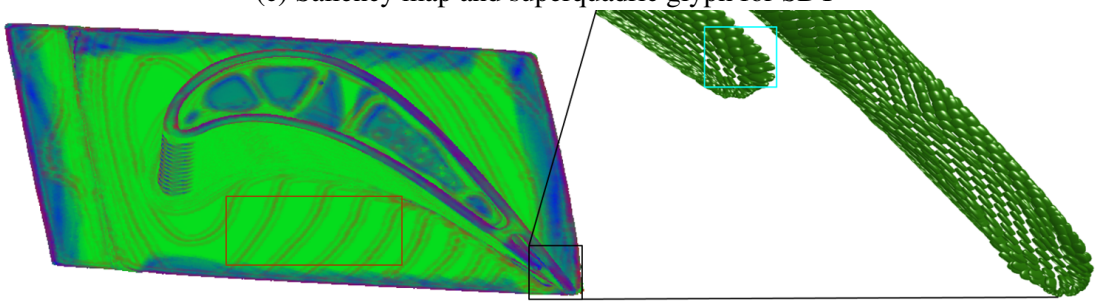
(a) Saliency map and superquadric glyph for ST



(b) Saliency map and superquadric glyph for NT

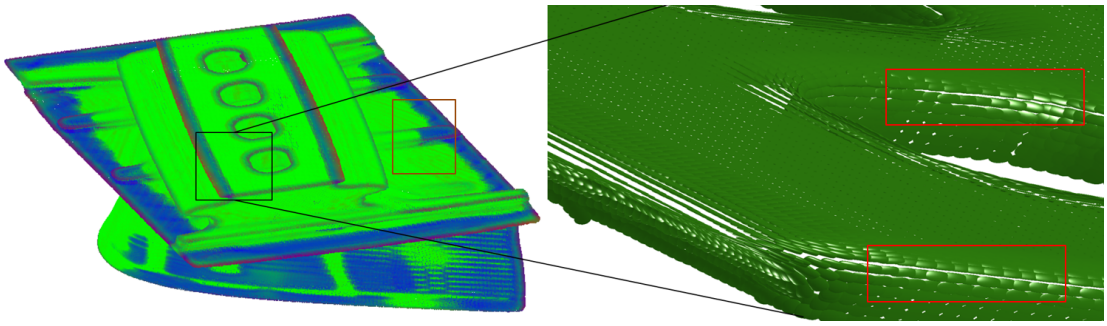


(c) Saliency map and superquadric glyph for SDT

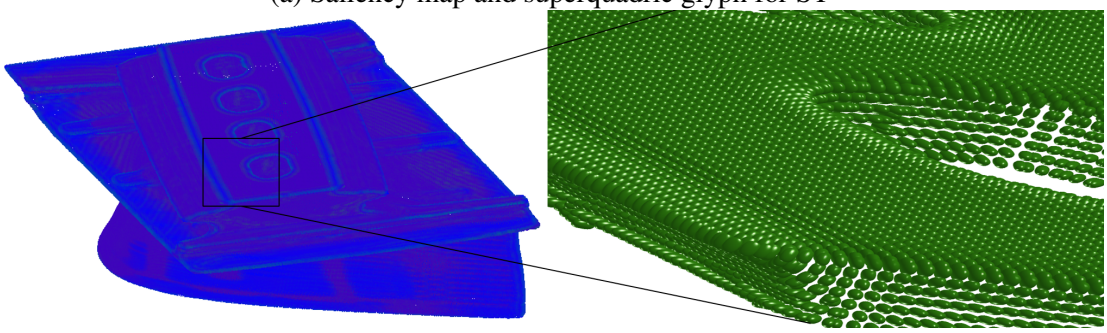


(d) Saliency map and superquadric glyph for NDT

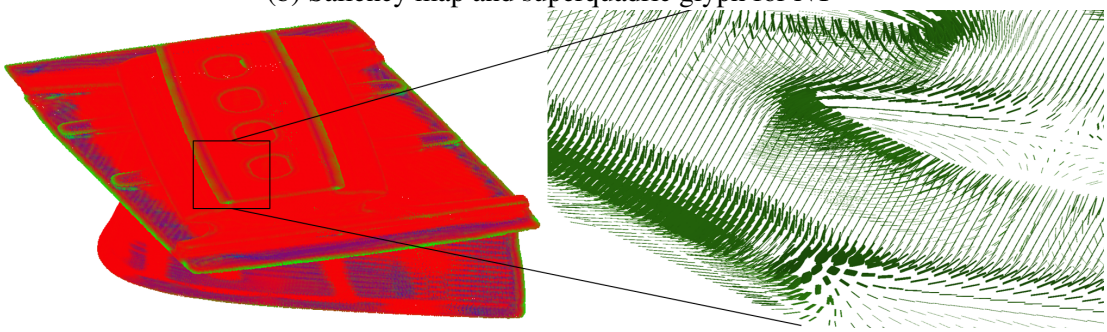
Figure FC6.3: Saliency map where R, G, B channels correspond to line-, surface- and point-type feature points, and superquadric glyphs for tensor orientation for the blade model (8,82,954 points). The blue boxes show where the shape of glyphs for point-type features are spherical in NDT while they have disc-type shape in ST. The brown boxes show where more scales (variations) are captured as line-type features in saliency map of NDT as compared to saliency map of ST.



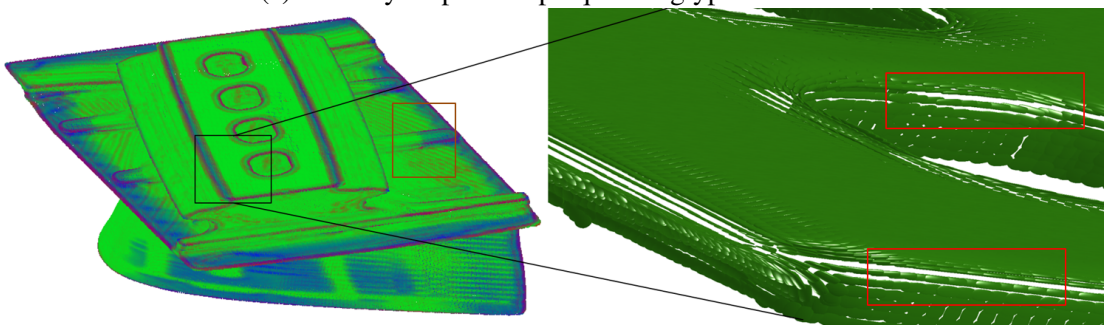
(a) Saliency map and superquadric glyph for ST



(b) Saliency map and superquadric glyph for NT

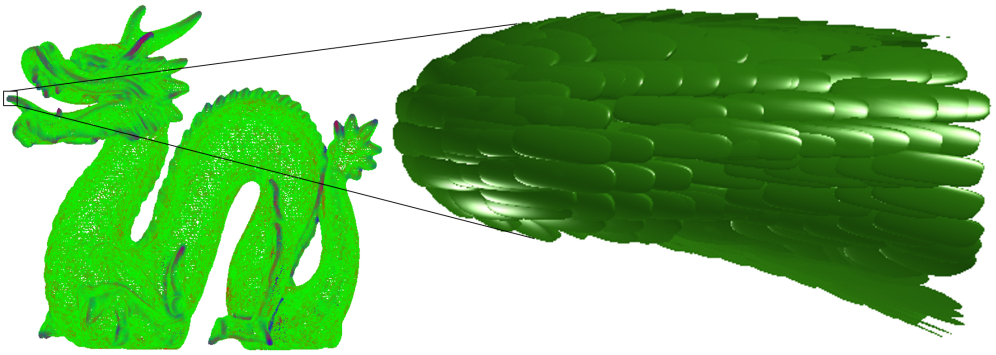


(c) Saliency map and superquadric glyph for SDT

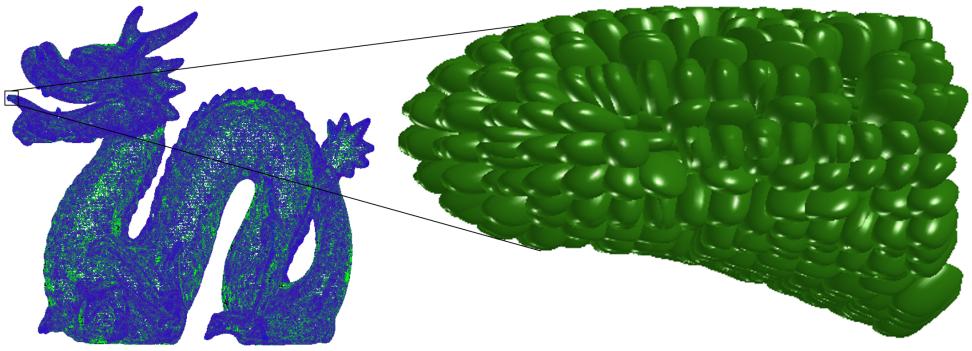


(d) Saliency map and superquadric glyph for NDT

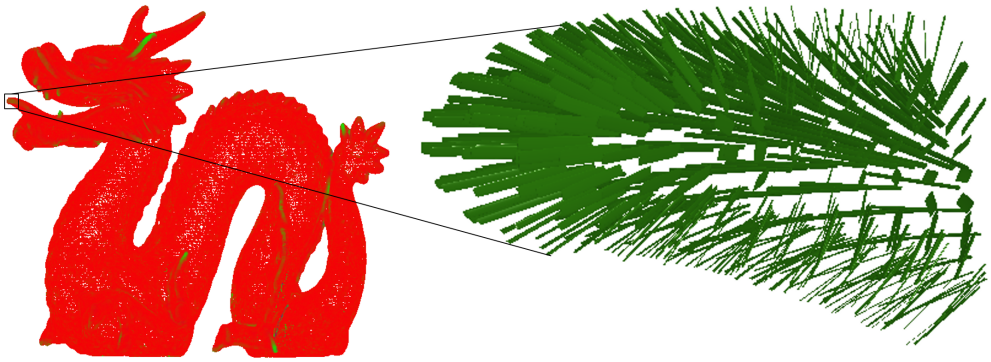
Figure FC6.4: Saliency map where R, G, B channels correspond to line-, surface- and point-type feature points, and superquadric glyphs for tensor orientation for blade model (8,82,954 points). The red boxes show few examples where the shape of glyphs for line-type features are more elongated in NDT than ST. The brown boxes show few examples where more scales (variations) are captured as line-type features in saliency map of NDT as compared to ST.



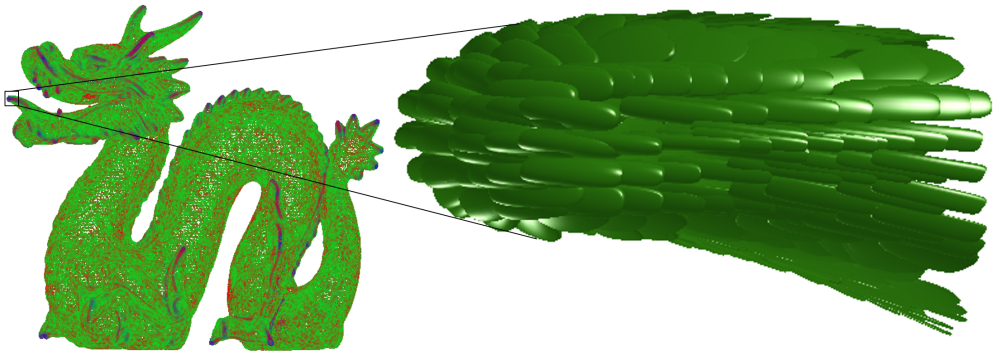
(a) Saliency map and superquadric glyph for ST



(b) Saliency map and superquadric glyph for NT

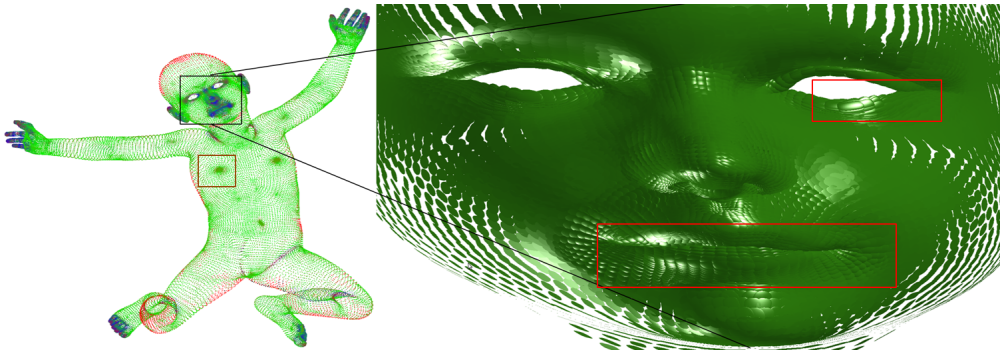


(c) Saliency map and superquadric glyph for SDT

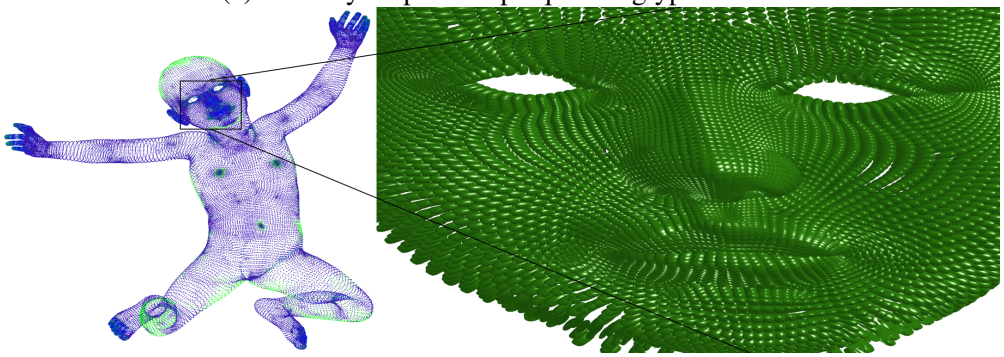


(d) Saliency map and superquadric glyph for NDT

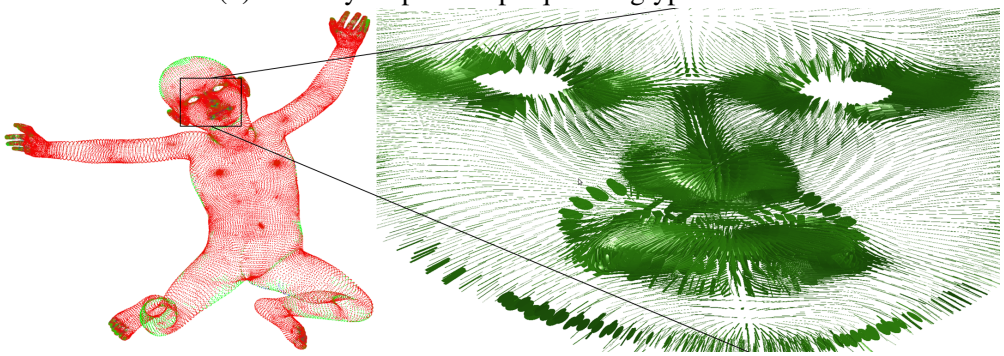
Figure FC6.5: Saliency map where R, G, B channels correspond to line-, surface- and point-type feature points, and superquadric glyphs for tensor orientation for dragon model (4,37,645 points), We can clearly see that glyphs are oriented along outward normal for high curvature region in NDT while, in case of ST, glyphs are directed along the tangent.



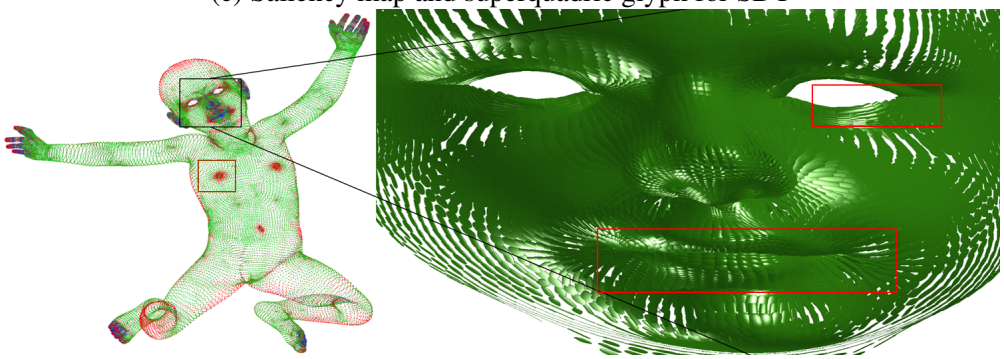
(a) Saliency map and superquadric glyph for ST



(b) Saliency map and superquadric glyph for NT

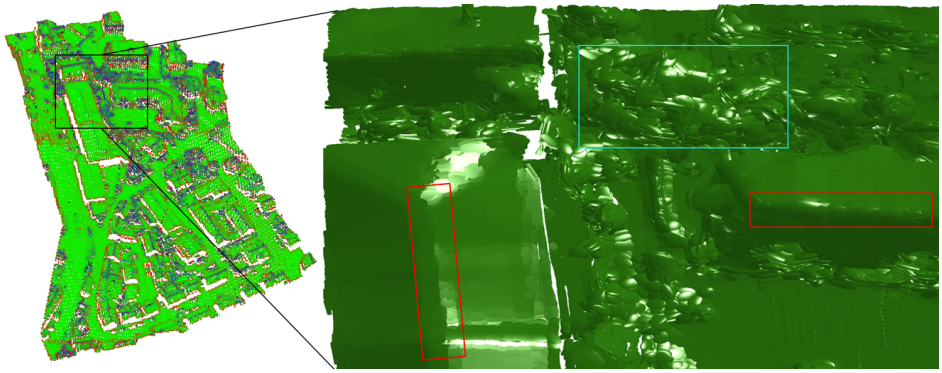


(c) Saliency map and superquadric glyph for SDT

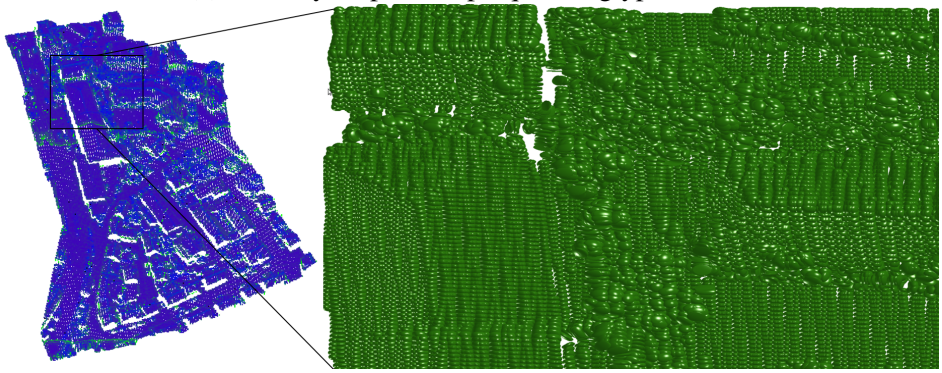


(d) Saliency map and superquadric glyph for NDT

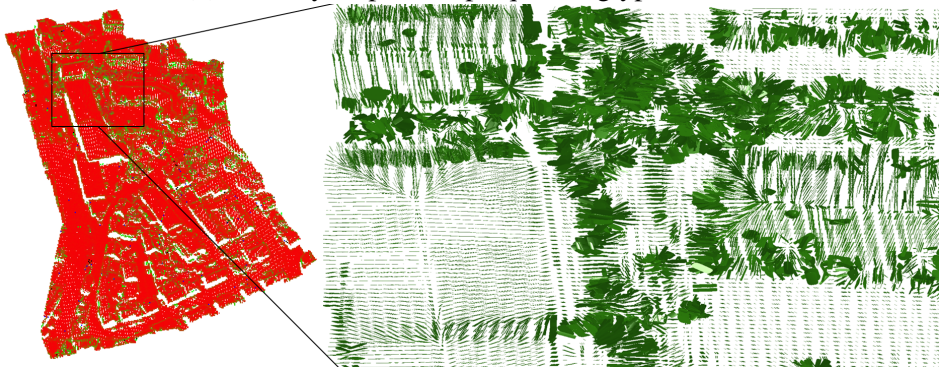
Figure FC6.6: Saliency map where R, G, B channels correspond to line-, surface- and point-type feature points, and superquadric glyphs for tensor orientation for child model (59,727 points). The red boxes show where NDT captures shape of eye socket and lip better in NDT than ST. The brown boxes show where nipple in the childs torso is captured more accurately in NDT than ST.



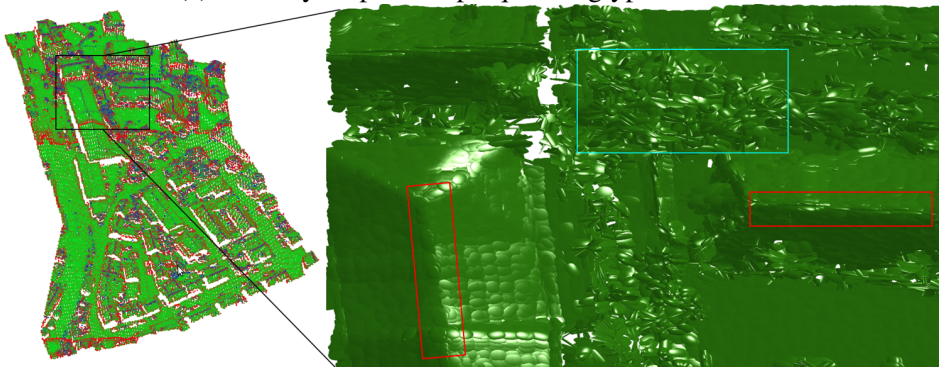
(a) Saliency map and superquadric glyph for ST



(b) Saliency map and superquadric glyph for NT

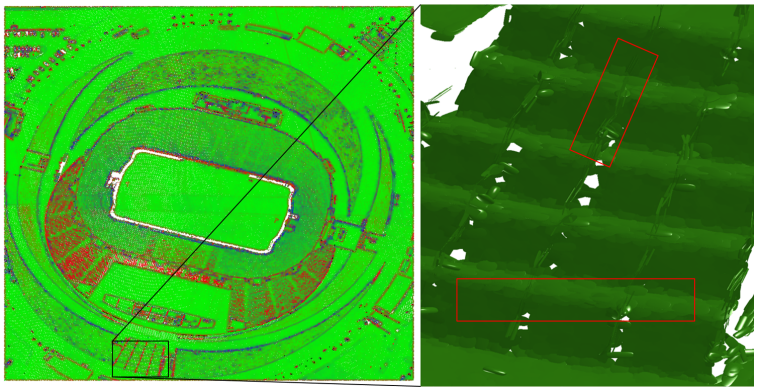


(c) Saliency map and superquadric glyph for SDT

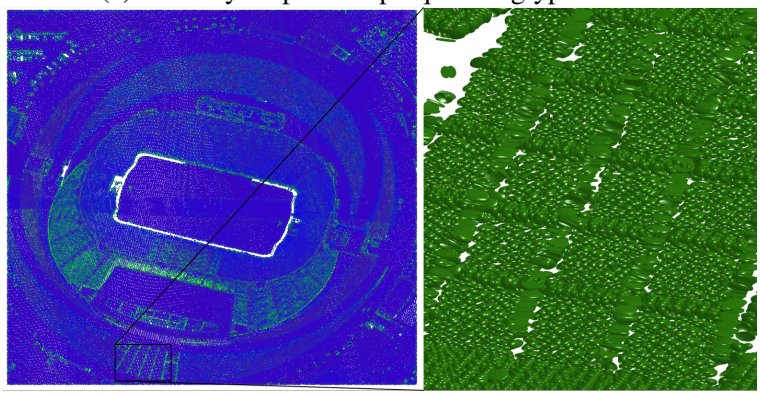


(d) Saliency map and superquadric glyph for NDT

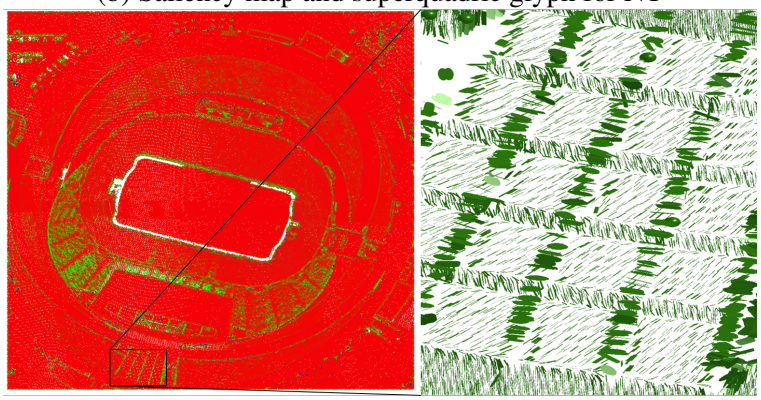
Figure FC6.7: Saliency map where R, G, B channels correspond to line-, surface- and point-type feature points, and superquadric glyphs for tensor orientation for Area 1 of the Vaihingen (1,79,997 points). The blue boxes show where tensors at trees have random orientation in case of NDT as compared to ST. The red boxes show where intersection of planes in the roofs of buildings are sharper in NDT than ST.



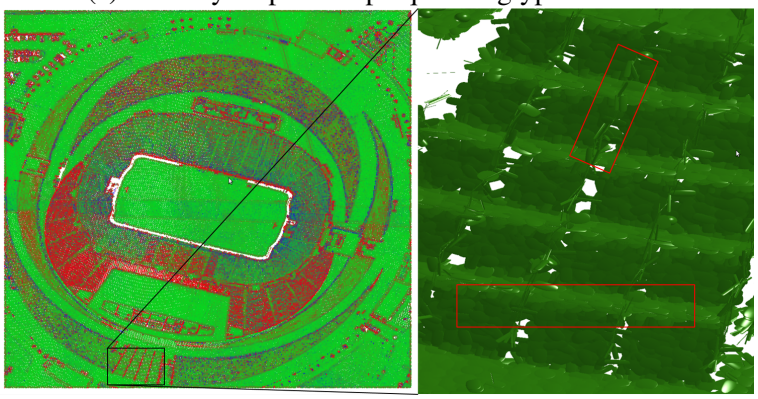
(a) Saliency map and superquadric glyph for ST



(b) Saliency map and superquadric glyph for NT



(c) Saliency map and superquadric glyph for SDT



(d) Saliency map and superquadric glyph for NDT

Figure FC6.8: Saliency map where R, G, B channels correspond to line-, surface- and point-type feature points, and superquadric glyphs for tensor orientation for Autzen stadium dataset (6,93,895 points). The red boxes show few examples where boundary of the seats are sharper in NDT than ST.

in unstructured point cloud.

In the case of normal tensor (NT), in comparison to structure tensor (ST), we find that once again the classes are interchanged: (a) surface-type (green) and point-type (blue) features in ST appear predominantly to be point-type (blue) features in NT; and (b) line-type (red) and some of the surface-type (green) features in the neighborhood of line-type (red) features in ST are captured as surface-type (green) features in NT. However, upon applying diffusion on NT to obtain NDT, point-type (blue) features convert to surface-type (green) features, and surface-type (green) features convert to line-type (red) features. Thus, compositing the comparison between ST and NT; and NT and NDT, we can conclude that the surface-type (green) features in NDT corresponds to surface-type (green) features in ST, point-type (blue) features corresponds to point-type (blue) features in ST and line-type (red) features in NDT corresponds to line-type (red) features in ST. Thus, the tensors ST and NDT show similar classification.

Since there is an improvement in accurately finding surface-type feature points in NT, which leads to line-type feature points in NDT. Therefore, NDT shows more accurate structural classification in comparison to ST. Also, diffusion velocities used in NDT control the diffusion process such that it diffuses slowly along the crossing of sharp edges or junction. This results in better detection of point- and line- type features.

In the Figures FC6.3 and FC6.4, we can clearly see that variations in the surface of the blade like creases and ridges are captured more accurately by NDT than ST as highlighted by brown boxes. In the Figure FC6.5, the dragon model is expected to be predominantly colored in red and green i.e. have more line- and surface- type feature points, owing to the presence of scales on the body of the dragon model. We observe that saliency map of NDT detects higher spatial frequencies than ST which is indicated by more reddish tint in the body of the dragon as shown in the Figure FC6.5.

We can clearly see in the brown boxes of the Figure FC6.6 that **nipples in the**

child's torso are detected as line-type features by NDT, While it is detected as surface-type features by ST. Also, facial features and belly button in the child model are better captured by NDT as compared to ST. Similarly, weak features in the saliency maps of Figures FC6.7 and Figure FC6.8 are captured by NDT better than ST.

From our experiments, we observe that ST and NDT show similar results for feature classification. In some cases, NDT outperforms ST by detecting higher spatial frequencies better than ST. It is evident from nipples and belly button in the child's torso in the Figure FC6.6, and presence of scales on the body of the dragon model, and in the blade model. Therefore, we can say that in unstructured point cloud, NDT encodes underlying geometry of a point better than ST.

6.5.2 Glyph Visualization

In NT, line-type (red) and some of the surface-type (green) features in the neighborhood of line-type (red) features are being classified as surface-type (green) features. Surface-type (green) and point-type (blue) features are being classified as point-type (blue) features; owing to which predominantly there are glyphs tending to spheres in the NT tensor field evident from inset of child in the Figure FC6.6. NT tensor is not oriented along the underlying geometry of the points while continuity in the field is preserved.

SDT gives good results for triangle meshes, but they are not very good feature descriptor for unstructured point clouds due to two reasons. Firstly, the points in the 1-ring neighborhood of points in triangle mesh is a subset of the points in the local neighborhood which are considered in the case of point cloud data. Due to the constrained behavior of the local neighborhood, which also exploits the connectivity information, the diffusion velocity is more controlled and better oriented in the case of triangle meshes than that of unstructured point cloud. Secondly, SDT in triangle meshes are vector prod-

uct of normals to the neighboring triangles, whereas in the case of point cloud, they are vector product of tangents to the vector joining the concerned point to the other points in the neighborhood.

We have observed that ST and NDT show similar behavior for feature classification in unstructured point cloud. In some cases, NDT is a better feature descriptor to encode local geometry of a point than ST. In the Figure FC6.3, the blue boxes show the region for point-type features where glyphs have spherical shape for point-type features in case of NDT, while glyphs have disc shape for point type feature in ST. Similarly, glyphs are expected to be oriented randomly for trees which we can see for NDT than ST as highlighted by blue boxes in the Figure FC6.7.

In the Figure FC6.4, red boxes show area where line-type features are captured more accurately by NDT than ST. Glyphs at higher curvature region are directed towards the outer normal of the surface in case of NDT, while they are directed along the tangent of the surface in ST as shown in the Figure FC6.5.

The borderlines of the eyes and shape of lips in child model are captured as line-type features, and glyphs are elongated and oriented along the borderlines in case of NDT than ST as shown by red boxes in the Figure FC6.6. Thus, NDT can define the shape of eye socket much better than ST. Similarly, at nose and lips of the child model, glyphs are oriented towards outward normal of the face, and facial features are more clearer in NDT. On the other hand, in the case of ST, glyphs are oriented along the surface of nose and facial features are not very clear. This can be due to the line-type features being captured as surface-type features and eigenvalues are small. NDT captures ridges and creases associated with relatively high curvature better than ST.

The intersection of the planes in the roofs of the buildings are sharper in NDT than ST as shown by the red boxes in the Figure FC6.7. The shape of the seats and creases are more visible in NDT than ST as shown by red boxes in the Figure FC6.8. The inter-

sections are captured as point-type features in NT, which upon anisotropic diffusion, are preserved to be point-type in NDT. The intersections are sharper in NDT as opposed to ST, as in the local neighborhood of the line of intersection of planes, the point-type features in NT get diffused to be surface-type features in NDT; whereas no improvements are made on the probability based classification identified by PCA in ST.

NDT performs better than SDT in the case of point cloud data, owing to the starting vector being normal in the former case than tangential in the latter case. NDT performs better than NT in superquadric glyph representations, owing to scaling the eigenvalues using an exponential function. This results in switching of major and minor eigenvectors which leads to the glyphs aligning tangential to the surface of the model, as opposed to being normal, as in the case of NT. Overall, NDT and ST show similar performance. From the experiments, we have predominantly observed that in some cases NDT captures high spatial frequencies better than ST such as the seats in the stadium, higher spatial frequencies in the blade and dragon model, and facial features in the child model.

6.5.3 Effects of Scale Parameter

For structural feature detection, radius r of local neighborhood of a point p is a scale parameter [8]. It is an user-defined parameter and very crucial to correctly determine the underlying geometry of the points in the dataset. As we have discussed in Sections 6.5.2 and 6.5.1, ST and NDT show similar performance for feature classification and detect features better than NT and SDT. Hence, we have performed an experiment to compare the saliency maps at different scales only for ST and NDT.

In the Figure FC6.9, we observe that, for radius = 0.02, NDT and ST both do not detect features very accurately, and at some places surface-type feature points are being misclassified as line-type features. However, NDT captures features more accurately

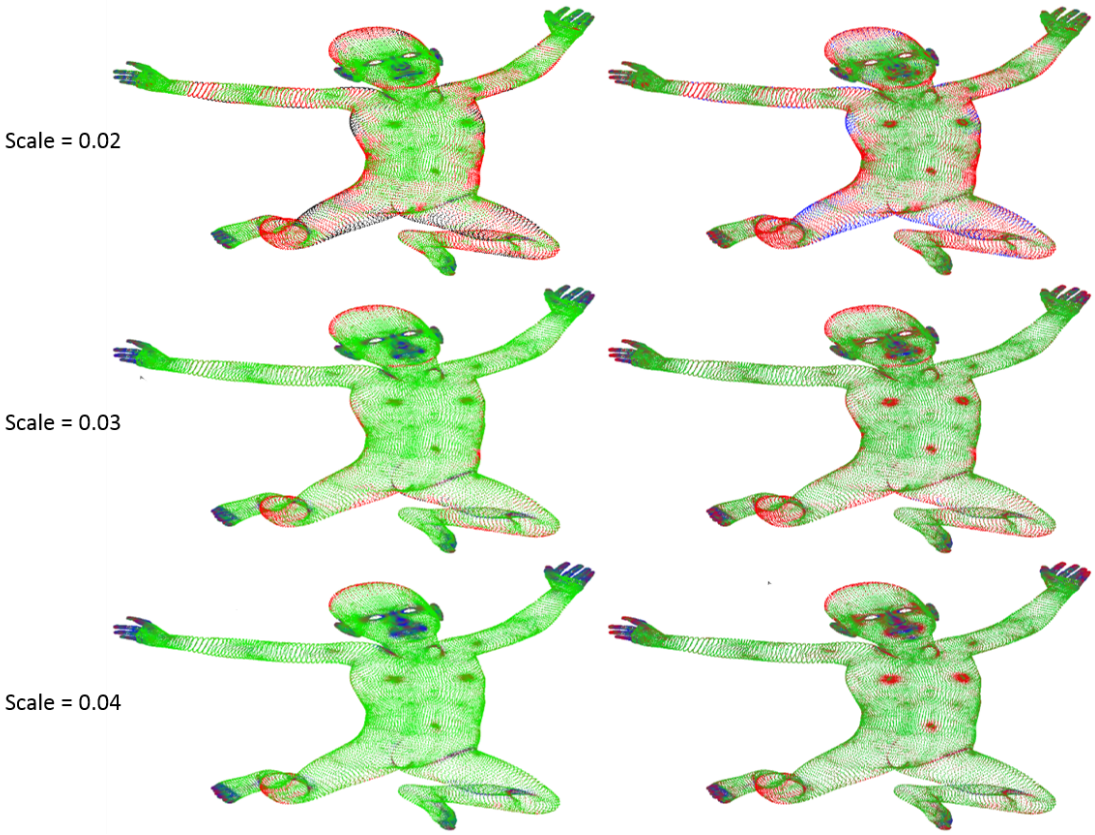


Figure FC6.9: Saliency maps of ST and NDT for different value of scales for child model (59,727 points).

at radius =0.03 and 0.04. Also, NDT captures high spatial frequencies better than ST, as we can see on the face of the child model in the Figure FC6.9 and on the seats in the Autzen stadium in Figure FC6.10. Therefore, we can say that NDT detects weak features more accurately than ST.

In the Figure FC6.10, ST at radius = 0.008 and NDT at radius = 0.012 show similar feature classifications visually. Therefore, we can say that it may not be necessary that same value of scale will be optimal for feature classification for all the tensors. Also, we observe that thickness of creases or ridges (line-type features) that is captured depends upon the the scale, which can be seen to be varying with the scale. The width of the line-type features at the facial features and fingers of the child in Figure FC6.9, are

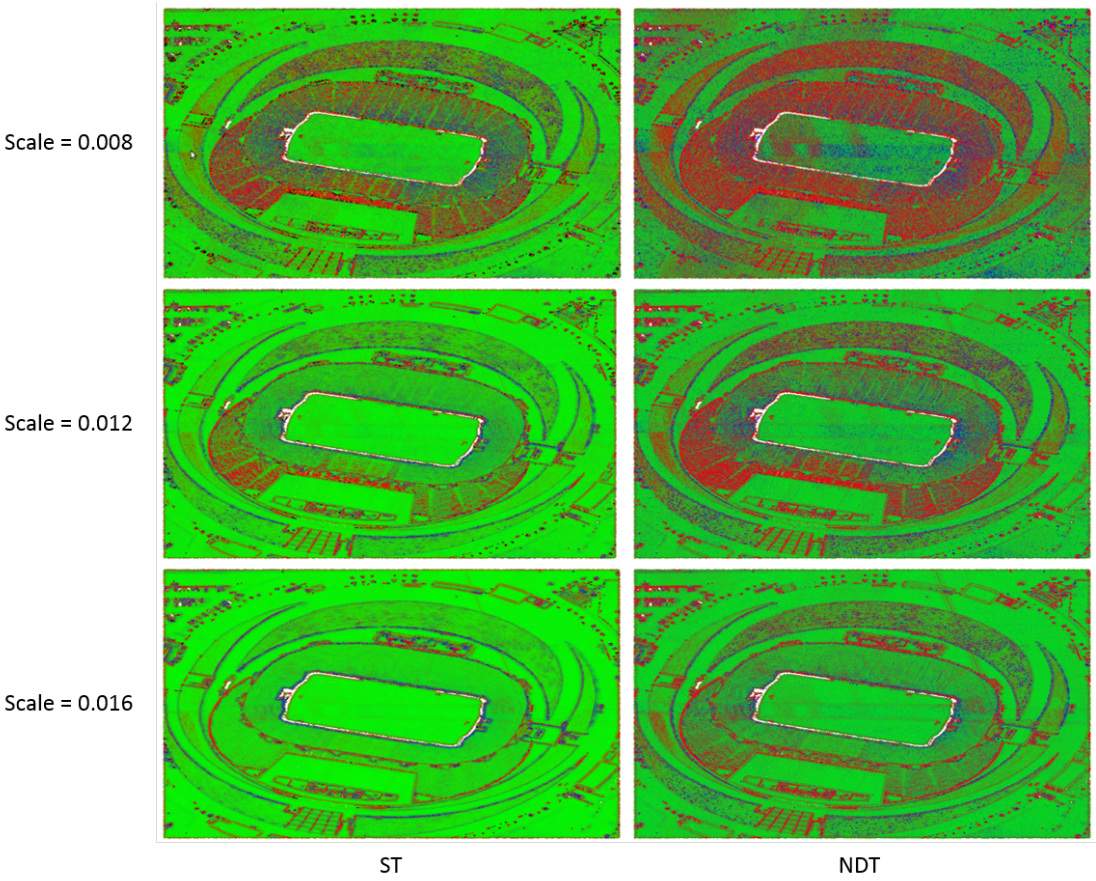


Figure FC6.10: Saliency maps of ST and NDT for different value of scales for Autzen stadium dataset (6,93,895 points)

varied with the scale. Similarly, in the Figure FC6.10, we can visually infer that the number of points detected as line-type features vary with the scale. Therefore, we need to fine tune this parameter to correctly classify the points into different feature classes.

Summarizing, we have observed that NDT encodes underlying geometry of the point better than ST. However, our computing of NDT is an initialization and it requires iteration based on global diffusion field [46]. Also, we have not studied the effect of diffusion velocity on structural classification. We can tweak this parameter to improve the classification. Currently, we have used structure tensor (ST) in our work for structural classification of points into surface- and line- type classes. As future work,

we will refine tensor field from NDT using Wang et al.'s algorithm in [46] for "refined classification", and compare ST field to the refined NDT field. Thus, we will be able to improve our results from "initial classification" as the "refined classification" will include information of both the local geometry and global field.

CHAPTER 7

RESULTS AND EXPERIMENTS

In this chapter, we have discussed the results for augmented semantic classification. We have tested our algorithm on the following airborne LiDAR system (ALS) datasets: Data 1 and Data 2 of Vaihingen dataset, from the ISPRS benchmark data [36]. Data 1 is an inner city of Vaihingen consisting of old historical buildings with a complex structure with vegetation. Data 2 is a residential area which contains high rising buildings of different sizes with less vegetation. The point density of the data varies between 4 and 7 $points/m^2$.

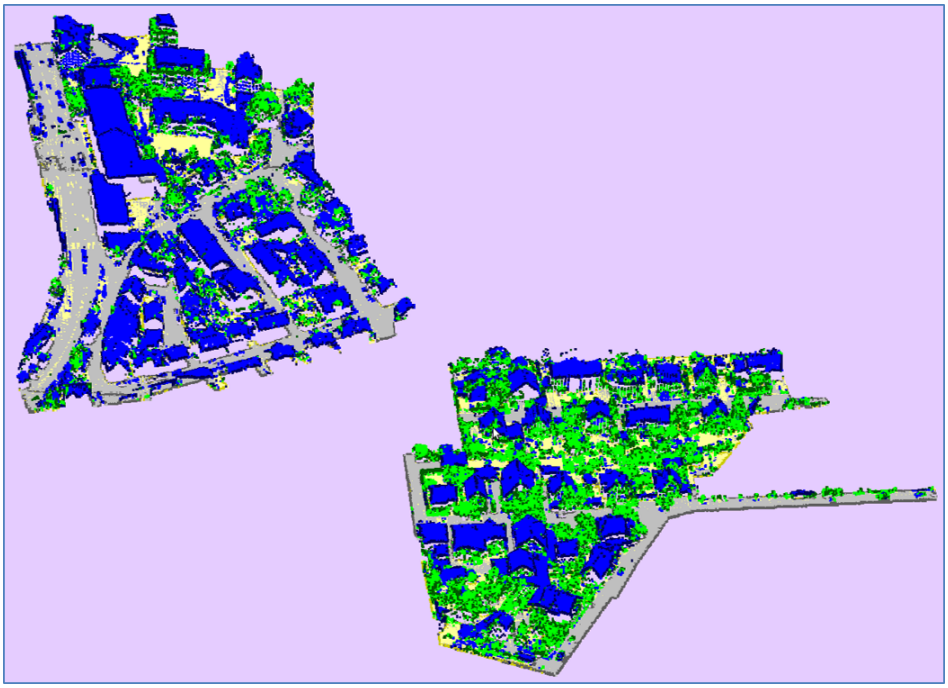
We have done evaluation of our algorithm using reference labels of Data 2 provided by ISPRS. Reference labels provided with test data consider the following nine classes: powerline , low vegetation, impervious surfaces, car, fence/hedge, roof, facade, shrubs and trees. They have labeled asphalt ground (or road) as impervious surface and natural ground as low vegetation. Since we are classifying points into following four classes: building, vegetation, natural ground and asphalt ground, we have logically merged few classes of above nine classes into a single class. We have combined the classes of shrubs and trees to consider equivalent to the vegetation class from our algorithm. Similarly, we have combined the classes of facade and roof to consider equivalent to the building class. We have labeled low-vegetation as natural ground and impervious surfaces as asphalt ground. Therefore, we have total seven classes: powerline, cars, building,

fence/hedge, vegetation, natural ground and asphalt ground. However, using our algorithm, we have classified points into four classes: building, vegetation, natural ground and asphalt ground. Hence, we have computed classification accuracy for building, vegetation, natural ground and asphalt ground classes.

Figure FC7.1 shows results of augmented semantic classification for Data 1 and Data 2 of Vaihingen dataset. The geometric-awareness of augmented segmented classification can be clearly seen in Figure FC7.1, where we can see that the points in (line,building) and (line,asphalt ground) classes showcase delineation of the building boundaries and road banks. For visualization purposes, differentiating (line, vegetation) and (surface, vegetation) gives perceptually better rendering of points in the vegetation class.

We have found that the augmented semantic classification to be visually accurate with very few misclassification. It is known from the existing literature that the disambiguation between the building and tree classes is challenging, owing to the similarity of values of feature descriptors of points belonging to these classes. At some places, buildings are misclassified as vegetation and vice-versa. The misclassification of tree as buildings could be attributed to the fact that in the case of dense trees, the foliage can be approximated to planar region. Therefore, trees are classified as buildings. Similarly, we have observed that while we are able to classify planar roofs correctly, the gabled roofs get misclassified as trees, as the intersection of planes on the roof shows line-type features just like trees. Thus, feature space of vegetation and buildings are not easily separable. Therefore, at some places buildings and trees are mislabeled with each other.

For quantitative evaluation, we have computed confusion matrices by comparing the output of our algorithm for Data 2 with the reference label provided by ISPRS. Confusion matrix, also known as error matrix, contains information about actual and predicted classifications done by a classification system. We have derived the following



(a) Data 1 of Vaihingen dataset

(b) Data 2 of Vaihingen dataset

Figure FC7.1: Results of our augmented semantic classification for Data 1 and Data 2 of Vaihingen dataset.

measured per class-wise from confusion metrics: completeness (recall), correctness (precision) and F-1 score. We have also computed overall accuracy of our method. To compute these measures, true positive (TP), false positive (FP), and false negatives (FN) are computed per class-wise. True positive indicates number of points are correctly classified. False positive indicates number of points are incorrectly classified. False negative tells us number of points are incorrectly rejected. Diagonal of confusion matrix gives TP. FP are computed along the row of confusion matrix and FN are computed along the column of confusion matrix. It has been discussed in more details in Section 7.1. The overall accuracy of our algorithm is 78.2%. In case of building and vegetation classes, points are misclassified classified with each other as their feature space are not very separable.

7.1 3D Evaluation

Using our tree visualizer tool, we can generate different choices of parameters in the tree, for hierarchical classification, which in our case is the HEM algorithm. We have experimented with two such sets of parameters as shown in the Figure FC7.2, which we will refer to as tree-1, and tree-2, in the rest of this chapter.

Classification accuracy of algorithm is measured using following metrics: precision, recall and F1-score. Precision is the number of TP divided by the number of TP and FP. It is the fraction of the predicted values that are relevant. It can be thought of as a measure of a classifier's exactness.

$$Precision = \frac{TP}{TP + FP}$$

Recall is the number of TP divided by the number of TP and the number of FN. It is the fraction of relevant instances that are predicted. It can be thought as a measure of

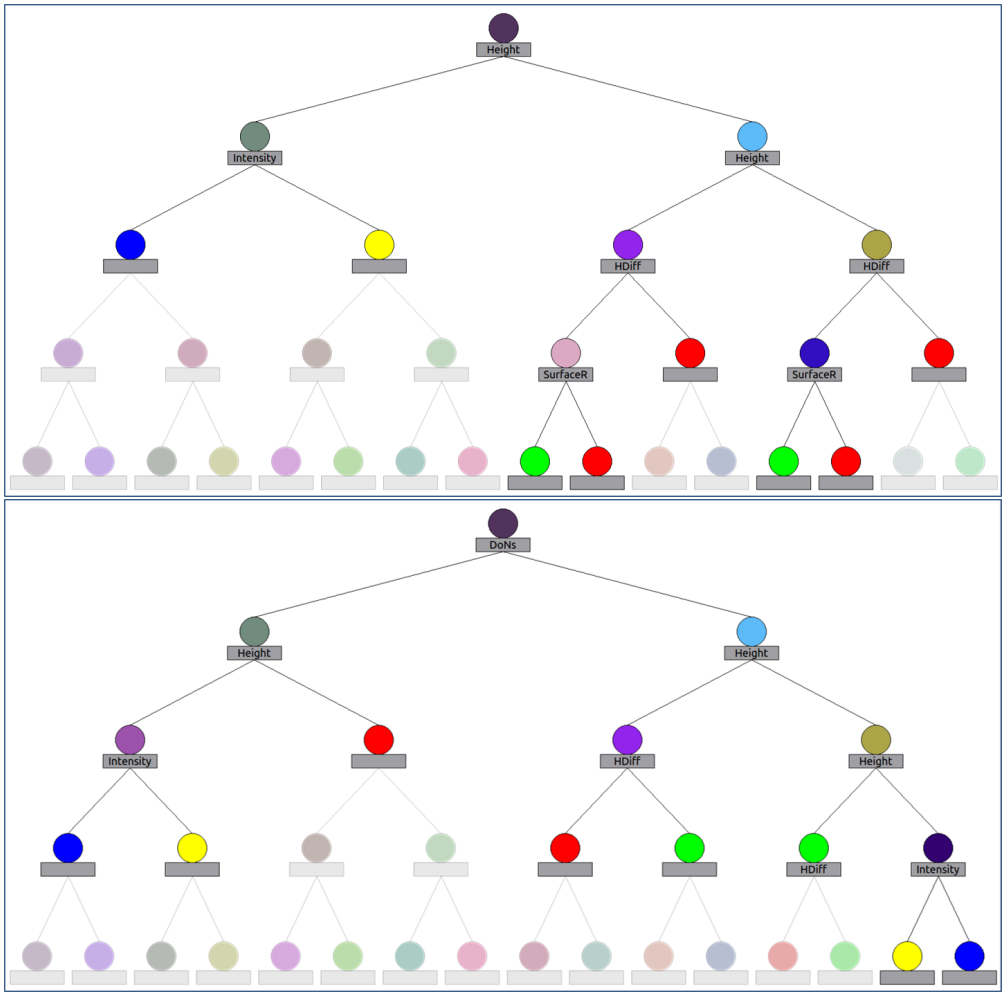


Figure FC7.2: Two possible clustering hierarchy for Data 2 of Vaihingen dataset from our tree visualize. Leaf nodes highlighted as red show building, leaf nodes highlighted as green show vegetation, leaf nodes highlighted as yellow show natural ground, and leaf nodes highlighted as blue show asphalt ground.

a classifier's completeness. .

$$Recall = \frac{TP}{TP + FN}$$

F1-score is a weighted harmonic mean between precision and recall. It gives the balance between the precision and the recall. It can have maximum value 1 (good performance of the system) and minimum 0 (bad performance of the system).

$$F1 - score = \frac{2 * (precision * recall)}{precision + recall}$$

In the domains of photogrammetry and remote sensing, precision is referred as correctness and recall as completeness or sensitivity. Correctness, completeness and F1-score are the metrics to measure the performance of the system for detection of a particular class. We have computed correctness, completeness and F1-score per class for tree-1 and tree-2, which are discussed in Sections 7.1.1, 7.1.2, 7.1.3 and 7.1.4.

We have also confusion matrix of Data 2 of Vaihingen dataset for tree-1 and tree-2. Confusion matrix, also known as error matrix, contains information about actual and predicted classifications done by a classification system. Tables TC7.1 and TC7.2 show confusion matrix for tree-1 and tree-2 for Data 2 of Vaihingen dataset. In the Table TC7.1, we can see that cars and powerline are classified as buildings. Therefore, FP will increase for buildings. On the other hand, cars and powerline are approximately equally misclassified as buildings and vegetation for tree-2 in the Table TC7.2. But, the number of points labeled as cars or building in the reference labels of Data 2 are very less as compared to other classes in the reference data. Therefore, it should not affect the accuracy of the algorithm to a great extent.

In Table TC7.1 (confusion matrix for tree-1), FP and FN are more for building and

vegetation due to buildings being misclassified as vegetation and vice-versa. Our algorithm fails in regions which demarcate buildings from vegetation. Further, FN are more for natural ground also due to natural ground being misclassified as asphalt ground.

In Table TC7.2 (confusion matrix for tree-2) misclassification of vegetation and natural ground as buildings are slightly reduced, while misclassification of buildings and natural ground as vegetation are slightly increased. This could be due to the use of different sets of parameters to generate hierarchical clustering in tree-1 and tree-2. In tree-2, first we classify points into planar and non-planar points. Then we classify planar points into building and ground points, and non-planar points into buildings and vegetation. Further, ground points are classified into natural and asphalt ground. While in case of tree-1, first we classify points into ground and non-ground points, then we classify non-ground points into buildings and vegetation, and classify ground points into natural and asphalt ground.

Reference Labels→ /Predicted Labels↓	Building	Vegetation	Natural Ground	Asphalt Ground	Powerline	Cars	Fence/ Hedge
Building	156471	51149	9938	1204	389	3432	9613
Vegetation	21400	124926	2909	176	121	961	1996
Natural Ground	512	4747	129516	13373	7	80	380
Asphalt Ground	912	1956	38487	178970	29	141	81

Table TC7.1: Confusion matrix obtained by 3D point classification of Data 2 using method tree-1. 3D reference labels provided by ISPRS used for construction of confusion matrix are provided by ISPRS.

7.1.1 Building

Table TC7.3 shows the correctness, completeness and F1-score for building class using tree-1 and tree-2. We observe that outcomes of different parameters in hierarchical classification for building class are different. In Figure FC7.3, black boxes show regions where vegetation are being misclassified as buildings using tree-1 while it is

Reference Labels→ /Predicted Labels↓	Building	Vegetation	Natural Ground	Asphalt Ground	Powerline	Cars	Fence/ Hedge
Building	141204	29014	3760	191	402	1941	2594
Vegetation	36695	147166	9107	1191	108	2457	9035
Natural Ground	454	4425	127346	13064	29	145	95
Asphalt Ground	942	2173	40637	179277	7	71	346

Table TC7.2: Confusion matrix obtained by 3D point classification of Data 2 using method tree-2. 3D reference labels provided by ISPRS used for construction of confusion matrix are provided by ISPRS.

correctly labeled using tree-2. Therefore, FP is more for tree-1 than tree-2. Thus, tree-2 shows higher correctness than tree-1. The red boxes in Figure FC7.3 show where buildings are correctly labeled using tree-1 than tree-2. Therefore FN is less for tree-1 than tree-2. Consequently, completeness for tree-1 is almost 11% more than tree-2. Since, F1-score is a weighted harmonic mean of correctness and completeness, therefore, it is comparable for both tree-1 and tree-2.

Method	TP	FP	FN	Correctness	Completeness	F1-score
tree-1	156471	75725	22824	0.673875	0.872701	0.760508
tree-2	141204	37902	38091	0.788382	0.787551	0.787967

Table TC7.3: Evaluation of tree-1 and tree-2 for building class using 3D reference labels of Data 2 provided by ISPRS.

7.1.2 Vegetation

Table TC7.4 shows the correctness, completeness and F1-score for vegetation class using tree-1 and tree-2. We observe that outcomes of different parameters in hierarchical classification are different. In Figure FC7.3, red boxes show where buildings are being mislabeled as vegetation using tree-2 than tree-1. The black boxes in Figure FC7.3 show where vegetation are correctly labeled for tree-2 but it is being misclassified as building using tree-1. Therefore, FP are more for tree-2 than tree-1, and FN are less for tree-2 than tree-1. Hence, tree-1 shows higher correctness than tree-2 and tree-2 shows

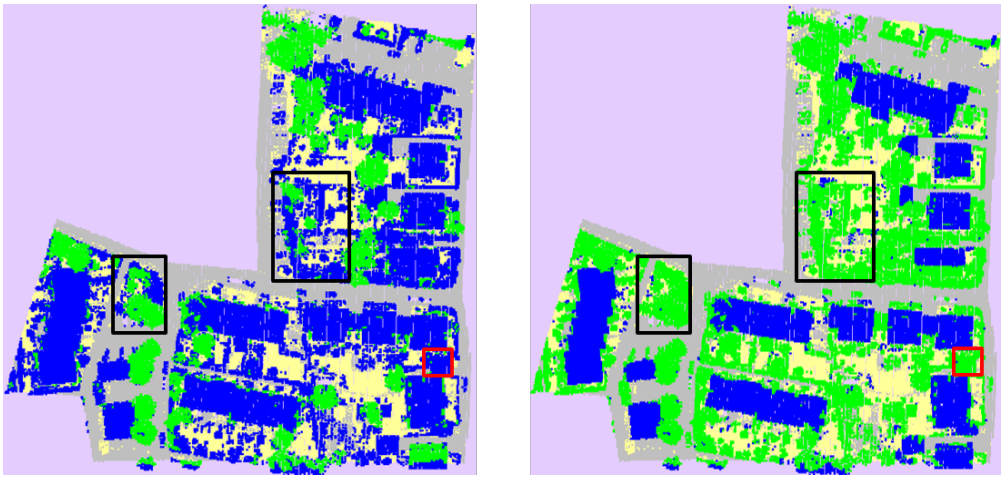


Figure FC7.3: Left and right image shows 3D evaluation for a small portion of Data 2 using tree-1 and tree-2. The black boxes show where vegetation are correctly classified using tree-2 than tree-1. The red boxes show where buildings are correctly classified using tree-1 than tree2.

higher completeness than tree-1. F1-score is comparable in the outcomes of tree-1 and tree-2.

Method	TP	FP	FN	Correctness	Completeness	F1-score
tree-1	124926	27563	57852	0.8192	0.6834	0.7452
tree-2	147166	58593	35612	0.7152	0.8052	0.7575

Table TC7.4: Evaluation of tree-1 and tree-2 for vegetation class using 3D reference labels of Data 2 provided by ISPRS.

7.1.3 Natural Ground

Table TC7.5 shows the correctness, completeness and F1-score for natural ground class using tree-1 and tree-2. We have observed that correctness, completeness and F1-score of natural ground class are comparable for tree-1 and tree-2.

Method	TP	FP	FN	Correctness	Completeness	F1-score
tree-1	129516	19099	51334	0.871487	0.716152	0.78622
tree-2	127346	18367	53504	0.873951	0.704153	0.779917

Table TC7.5: Evaluation of tree-1 and tree-2 for natural ground class using 3D reference labels of Data 2 provided by ISPRS.

7.1.4 Asphalt Ground

Table TC7.6 shows the correctness, completeness and F1-score for asphalt ground class using tree-1 and tree-2. We have observed that correctness, completeness and F1-score of asphalt ground class are comparable for tree-1 and tree-2.

Method	TP	FP	FN	Correctness	Completeness	F1-score
tree-1	178970	41606	14753	0.811376	0.923845	0.863965
tree-2	179277	44021	14446	0.80286	0.92543	0.859798

Table TC7.6: Evaluation of tree-1 and tree-2 for asphalt ground class using 3D reference labels of Data 2 provided by ISPRS.

7.1.5 Summary of comparison of outcomes of tree-1 and tree-2

We observe that our classification algorithm labels buildings more accurately when using tree-2 as opposed to tree-1. The vegetation class is labeled more accurately when using tree-1 as opposed to tree-2.

Overall accuracy is comparable for our classification algorithm when using tree-1 and tree-2, which are 78.25% and 78.92%, respectively, for Data 2. We have computed the overall accuracy as the ratio of number of points correctly classified to total number of points. Total points in Data 2 are 753876, 589883 points were correctly classified when using tree-1 and 594993 points, when using tree-2.

Niemeyer et al. have used CRFs (conditional random field) in conjunction with random forest classifier for semantic classification of 3D point cloud. They have used CRFs to incorporate the context information in their classifier. They have classified points into following classes: grassland, road, building with gable roof, low vegetation, facade, building with flat roof, and tree [16]. In our case, we have classified points into building, vegetation, natural ground and asphalt ground. For comparison purpose, we have logically combined few classes proposed by Niemeyer et al. into a single class.

Building with flat roof and building with gable roof are combined into building class. Similarly, low vegetation and tree are combined into vegetation class. In our case, road is labeled as asphalt ground and grassland is labeled as natural ground. Therefore, we compared the road and grassland class with our asphalt ground and natural ground classes respectively. Since, we have merged building with flat roof with building with gable roof, we have taken average of their completeness and correctness to compare with our building class. Similarly, We have merged low vegetation class with tree class, we have taken average of their completeness and correctness to compare with our vegetation class. Table TC 7.7 shows comparison between method proposed by niemeyer et al. and our method.

Method	Completeness in %	Correctness in %
	Building	
Niemeyer et al.'s method [16]	92.95	84.9
Our Method	87.27	67.39
	Vegetation	
Niemeyer et al.'s method [16]	71.10	75.75
Our Method	68.34	81.92
	Natural Ground	
Niemeyer et al.'s method [16]	84.4	80.9
Our Method	71.61	87.15
	Asphalt Ground	
Niemeyer et al.'s method [16]	87.1	93.0
Our Method	92.34	81.37

Table TC7.7: Comparison between method proposed by Niemeyer et al. [18] and Our method

The correctness of vegetation and natural ground classes obtained using our method is better than niemeyer et al.'s method. However, their algorithm performed better than our method in case of building class. The overall accuracy of their algorithm is 83.4% while overall accuracy of our method is 78.2%. Currently, we are using single parameter for clustering. We can improve the accuracy of our algorithm by finding optimum set of feature descriptor. We have not explored different clustering algorithms, and currently we are using EM-GMM. We can further improve accuracy of our algorithm by changing

the clustering algorithm.

7.2 Domain Expert User Evaluation

Our tool has been evaluated by domain experts in GIS, visualization, and machine learning [please refer acknowledgement section for details on experts]. The GIS expert mentioned that the attractiveness of tool is the compelling use case of identifying line-type features and overlaying it on the object-based classification. However, he commented on the requirement to scale up the application to larger areas. He mentioned that the accuracy of the footprints, shown in the Figure FC7.4, generated from our tool in comparison to the corresponding orthoimages can be visually rated at 80-85%.

The visualization expert commented on the effective use of heat maps for deciding on parameters for semantic classification, and the possibility of extending the visual analytics capability to study covariance of parameters taken pairwise.

The machine learning expert, who reviewed our work, commented on the limitations of using EM algorithm owing to the underlying data model not being a Gaussian mixture model, and encouraged us to explore the effects of changing the clustering algorithm.

7.3 Discussions

In this section, we discuss some of the salient properties of our algorithm.

Unsupervised machine learning approach: We have used two independent unsupervised multi-class classification methods, whereas, several existing methods combine geometric and object-based features into a single feature vector, and use a supervised

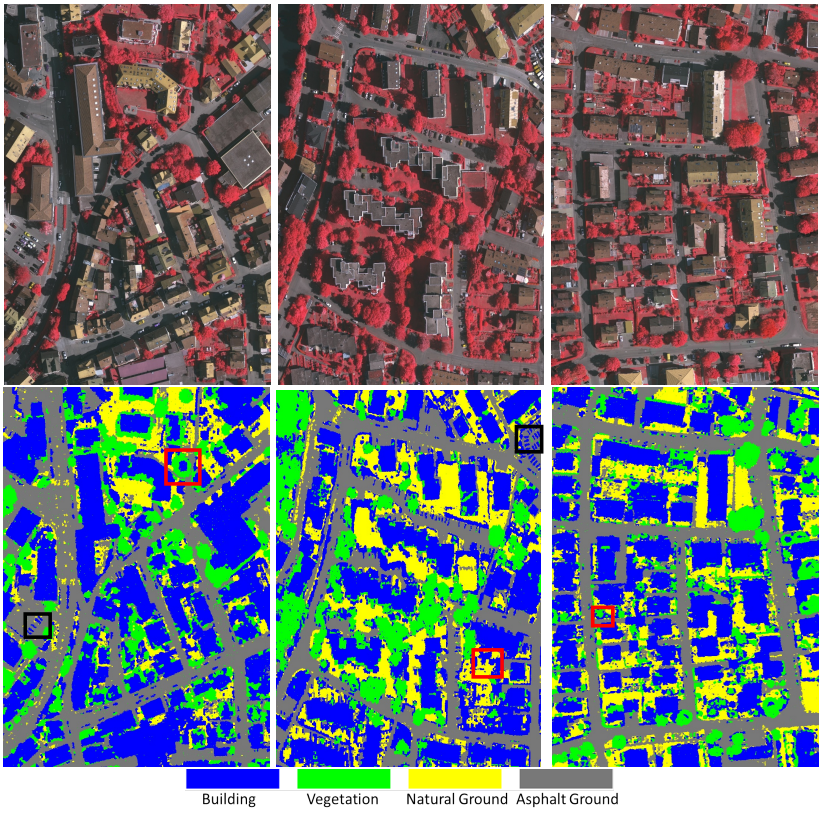


Figure FC7.4: Comparing footprint images of semantic classification of Areas 1, 2, and 3 of Vaihingen dataset (top-to-bottom) in the right to the corresponding orthoimages in the left. While the classification look qualitatively correct by visual analysis, there are few misclassification. The red boxes show few examples of misclassification of vegetation and building; the black ones show misclassification of cars, which is not our targeted class.

or a semi-supervised classifier. We used unsupervised method owing to the absence of a training set to work with. Our work is, in that regard, a proof-of-concept to show that an unsupervised classifier can work well for semantic labeling.

Need for using two independent multi-class classification: While it is well known that semantic labeling, obtained from supervised learning on LiDAR parameters and geometric features, is essential for object classification, the geometry-based classification is not usually preserved. Line-type features are essential in determining important topological structures in terrain and non-terrain, such as breaklines, ridges, valleys, etc., which are conventionally computed from the grid image, obtained from LiDAR eleva-

tion data, rather than from the point clouds [29]. Using our geometry-based classification, these features can be extracted from the point clouds directly. Semantic labeling which preserves the geometry-based class, helps in retrieving these features easily.

Using combined geometric and object-based classification, as done by Zhang et al. [57], gives equal weightage to all the various features. Zhang et al. have used thirteen features, which include geometric, echo, radiometric, and topological features, and used them in a SVM (support vector machine) classifier. In combined classification, unless each feature is normalized, the distance metric used for classification will be biased in each feature dimension. In the combined feature vector, where importance of variables is known a priori, features are weighted according to the number of features belonging to each class being used. While, our method ensures that each logical partition of features (i.e. geometric and radiometric, in our case) are given equal weightage.

Chehata et al. [13] have used several features and a grouping of feature types, namely, height, echo, eigenvalue, local plane, and full-wave LiDAR features, with given variable importance to these feature types. They have additionally computed the variable importance of LiDAR data using mean decrease permutation accuracy. They have reported that the most important features are height-based. We have used normalized height parameter to distinguish ground points from non-ground points. Additionally, the supervised classification tends to reduce the contribution of the geometric dimensions, as shown in [13]. The geometric features are computed using the multi-scale approach as well as for performing eigen-analysis. Hence, they are relevant local descriptors computed from local reference frame, for automatically finding object-to-object correspondences. Using the independent classifications and retaining the individual labels in the semantic label, our method effectively preserves the geometric information.

For instance, the fence data shown in [15] can be easily derived as a (line, building) class using our method. Similarly, the building footprint and boundary are usually

retrieved using Delaunay triangulation of the clustered points, and finding a bounding polygon using the points [11]. However, using our method, the line-type features could be directly used as seed for finding the building boundary and footprint. Similarly, the building class can be directly used for extracting the surface.

Limitations The current limitation of our framework is that a user has to look into heat-map of various feature descriptors and decides the clustering parameter. In order to avoid the multiple inputs from a user, multi-dimensional visualization techniques such as scatter plot and parallel coordinates can be used to visualize all feature descriptor together in a single view space. Also, we can explore different machine learning techniques to discard the redundant features and provide a user with an optimum set of feature descriptors. To make the process semi-automate, we can do feature ranking and provide a score to a user which will help a user in selecting clustering parameter. We have observed that the choices of parameters made in the tree-visualizer for a dataset can be reused for other similar data-sets. An option can be provided to user such that user can save the choices of clustering parameters of tree-visualizer and use the same setting for other data-set. This will make the process automatic for other similar dataset, as opposed to finding these parameters from scratch.

Software Release: The initial version of software has been developed for remote visualization of LiDAR data and its various features as a part of a project funded by **National Resource Data Management System (NRDMS), Department of Science and Technology, Govt. of India**. We have released the first version of our tool to NRDMS in August 2015. For semantic classification, work is still in progress, and hence is not ready for release. We are working on improvements in semantic classification and parallelization of more modules to make the application more interactive. This work will be continued in the Graphics-Visualization-Computing Lab over a year's time.

CHAPTER 8

CONCLUSIONS & FUTURE WORK

Several supervised and unsupervised methods exist in the literature for semantic labeling of objects in 3D urban LiDAR point cloud. The associated problem with these approaches is the lack of guarantee in accessibility and availability of domain expert. **We have used visualization to guide unsupervised technique for semantic labeling of objects in urban LiDAR point cloud, and proposed a visual analytic framework for the same**. Our method enables a user to detect objects in urban LiDAR point cloud in the absence of training data. A user selects classification parameter at each active node of tree-visualizer by looking into heat-maps of feature descriptor for semantic classification. In other words, a user guides classifier by selecting the clustering parameter which can seen as alternative to training given to the classifier in supervised methods. In supervised classification, classifier produce an inferred function from training data and use this function to perform classification on testing data, and in unsupervised method, classifier learns from data itself. In our case, classifier gets inputs from a user for clustering parameter, and use this clustering parameter to find natural cluster in the data using unsupervised method EM-GMM. Similar to supervised classification, our method gets training in form of clustering parameter and termination criteria by a user.

Semantic labeling method uses geometric features and LiDAR parameters for semantic classification. These geometric features are not preserved with the semantic

labels. We attempted to preserve geometric features with semantic labels, and proposed a new classification namely, augmented semantic classification, for LiDAR point cloud data by combining structural and semantic classifications. We have used a multi-scale approach for extracting structural features from the point cloud directly, which ensures persistence of features across multiple scales. We have explored local second order symmetric semi-definite tensor based feature detection methods for structural classification.

We have used anisotropic diffusion tensor for structural feature extraction in unstructured point cloud. We have computed two diffusion tensors derived from structure and normal voting tensor for structural feature extraction in unstructured point cloud, and compared the feature classification results obtained from these tensors with structure and normal voting tensors. Our approach is an extension of the method proposed by Wang et al. [46] for triangle meshes to unstructured point cloud. We have used saliency values, and shape and orientation of the tensor as metrics for comparison of the feature classification results obtained from ST, NT, SDT and NDT. We have used visualization techniques, saliency map and superquadric glyph visualization, to visually assess different local tensor-based feature detection methods.

In structural classification, we have shown how anisotropic diffusion tensor, derived from normal voting tensor in point cloud, is a good feature descriptor than the structure and voting tensor. Our computation of normal diffusion tensor is an initialization to encode the structural characteristic of the point and requires iteration based on global diffusion process [46]. As future work, we will refine tensor field obtained from normal diffusion tensor using the method proposed by Wang et al. [46]. Currently, we have used structure tensor for structural feature extraction as proposed by Keller et al. [8]. We have identified that the algorithm proposed by Keller et al. is embarrassingly parallel, and performed parallel implementation of the algorithm using data-parallel paradigm using GPGPU computing.

For semantic classification, we have used a new unsupervised machine learning approach, namely, an interactive hierarchical divisive clustering algorithm followed by region growing. We have used our proposed *tree visualizer* tool to perform clustering in an interactive way in the feature space, using Expectation Maximization (EM) algorithm. Currently, we are using single parameter for clustering at each active node of tree visualizer for semantic classification. In our future work, we are planning to find optimum set of parameters using visualization and use them for clustering at each active node of tree visualizer.

Augmented semantic classification is obtained by combining structural and semantic labels as tuples. Thus, we have labels for each points as tuples such as (line, building), (surface, building), etc., by combining (line, surface) structural classes with four semantic classes buildings, vegetation, natural ground and asphalt ground. The accuracy of our semantic classification algorithm is 78.2%, and the geometric-awareness of our augmented semantic classification has benefits in visualizing, and identifying boundaries and structure. Our classifications are computed in real-time upon user selection of parameters for the classification. However, the computation of feature descriptors has not been parallelized, upon which the preprocessing time can reduce drastically, which is currently around 3 minutes for Area 1 of Vaihingen dataset.

In addition, we have implemented a visual analytic framework for augmented semantic classification as a proof-of-concept. Domain experts who have reviewed our tool have appreciated the usefulness of interactivity in determining the hierarchical clustering, and the interactive updates of the visualizations owing to the parallel implementation. However, the machine learning expert commented on the limitations of using EM algorithm owing to the underlying data model not being directly representable as a Gaussian mixture model, and encouraged us to explore the effects of changing the clustering algorithm.

Bibliography

[1] Michael Cramer. The dgpf-test on digital airborne camera evaluation–overview and test design. *Photogrammetrie-Fernerkundung-Geoinformation*, 2010(2):73–82, 2010.

[2] Xiaoli Sun. Space-based lidar systems. In *CLEO: Applications and Technology*, pages JW3C–5. Optical Society of America, 2012.

[3] Keil Schmid, Kirk Waters, Lindy Dingerson, Brian Hadley, Rebecca Mataosky, Jamie Carter, and Jennifer Dare. Lidar 101: An introduction to lidar technology, data, and applications. *NOAA Coastal Services Center*, 2012.

[4] B. Lohani. Airborne Altimetric LiDAR: Principle, Data Collection, Processing and Applications, 2007.

[5] Emmanuel P Baltsavias. A comparison between photogrammetry and laser scanning. *ISPRS Journal of photogrammetry and Remote Sensing*, 54(2):83–94, 1999.

[6] Min Ki Park, Seung Joo Lee, and Kwan H Lee. Multi-scale Tensor Voting for Feature Extraction from Unstructured Point Clouds. *Graphical Models*, 74(4):197–208, 2012.

[7] Christopher Weber, Stefanie Hahmann, and Hans Hagen. Methods for Feature Detection in Point Clouds. In *OASIcs-OpenAccess Series in Informatics*, volume 19. Schlöss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2011.

[8] Patric Keller, Oliver Kreylos, Marek Vanco, Martin Hering-Bertram, Eric S Cowgill, Louise H Kellogg, Bernd Hamann, and Hans Hagen. Extracting and visualizing structural features in environmental point cloud LiDaR data sets. In *Topological Methods in Data Analysis and Visualization*, pages 179–192. Springer, 2011.

[9] R Blomley, M Weinmann, J Leitloff, and B Jutzi. Shape distribution features for point cloud analysis-a geometric histogram approach on multiple scales. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1:9–16, 2014.

[10] Norbert Haala and Claus Brenner. Extraction of buildings and trees in urban environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2):130–137, 1999.

[11] Mohammad Awrangjeb, Guojun Lu, and Clive S Fraser. Automatic Building Extraction from LiDAR Data Covering Complex Urban Scenes. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1:25–32, 2014.

[12] Jeong-Heon Song, Soo-Hee Han, KY Yu, and Yong-Il Kim. Assessing the Possibility of Land-cover Classification Using LiDAR Intensity Data. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/B):259–262, 2002.

[13] Nesrine Chehata, Li Guo, and Clément Mallet. Airborne lidar feature selection for urban classification using random forests. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(Part 3):W8, 2009.

[14] J Niemeyer, F Rottensteiner, and U Soergel. Conditional random fields for lidar point cloud classification in complex urban areas. *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, 1(3):263–268, 2012.

[15] Joachim Niemeyer, Franz Rottensteiner, and Uwe Soergel. Classification of urban LiDAR data using conditional random field and random forests. In *Urban Remote Sensing Event (JURSE), 2013 Joint*, pages 139–142. IEEE, 2013.

[16] Joachim Niemeyer, Franz Rottensteiner, and Uwe Soergel. Contextual classification of lidar data and building object detection in urban areas. *ISPRS journal of photogrammetry and remote sensing*, 87:152–165, 2014.

[17] Aleksey Golovinskiy, Vladimir G Kim, and Thomas Funkhouser. Shape-based recognition of 3d point clouds in urban environments. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2154–2161. IEEE, 2009.

[18] Markus Eich, Malgorzata Dabrowska, and Frank Kirchner. Semantic labeling: Classification of 3d entities based on spatial feature descriptors. In *IEEE International Conference on Robotics and Automation (ICRA2010) in Anchorage, Alaska*, 2010.

[19] Tahir Rabbani, Frank van den Heuvel, and G Vosselmann. Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5):248–253, 2006.

[20] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Computer Vision–ECCV 2010*, pages 356–369. Springer, 2010.

[21] AM Ramiya, Rama Rao Nidamanuri, and R Krishnan. Semantic Labelling of Urban Point Cloud Data. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1:907–911, 2014.

[22] Zahra Lari and Ayman Habib. Segmentation-based classification of laser scanning data. In *ASPRS: Annual Conference, Sacramento, California, USA*, 2012.

[23] Suddhasheel Ghosh and Bharat Lohani. Heuristical Feature Extraction from LiDAR Data and Their Visualization. In *Proceedings of the ISPRS Workshop on Laser Scanning 2011*, volume 38. University of Calgary Canada, 2011.

[24] Florent Lafarge and Clément Mallet. Creating large-scale city models from 3d-point clouds: a robust approach with hybrid representation. *International journal of computer vision*, 99(1):69–85, 2012.

[25] Diansheng Guo, Donna Peuquet, and Mark Gahegan. Opening the black box: interactive hierarchical clustering for multivariate spatial patterns. In *Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, pages 131–136. ACM, 2002.

[26] Eli Packer, Peter Bak, Mikko Nikkila, Valentin Polishchuk, and Harold J Ship. Visual analytics for spatial clustering: Using a heuristic approach for guided exploration. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2179–2188, 2013.

[27] Sujin Jang, Niklas Elmqvist, and Karthik Ramani. Gestureanalyzer: visual analytics for pattern analysis of mid-air hand gestures. In *Proceedings of the 2nd ACM symposium on Spatial user interaction*, pages 30–39. ACM, 2014.

[28] Reinhold Preiner, Oliver Mattausch, Murat Arikan, Renato Pajarola, and Michael Wimmer. Continuous projection for fast l-1 reconstruction. *ACM Transactions on Graphics (TOG)*, 33(4):47, 2014.

[29] Xiaoye Liu. Airborne LiDAR for DEM generation: some critical issues. *Progress in Physical Geography*, 32(1):31–49, 2008.

[30] Jérôme Demantké, Clément Mallet, Nicolas David, and Bruno Vallet. Dimensionality based Scale Selection in 3D LiDAR Point Clouds. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(Part 5):W12, 2011.

[31] Adrien Gressin, Clément Mallet, Jérôme Demantké, and Nicolas David. Towards 3D LiDAR Point Cloud Registration Improvement Using Optimal Neighborhood Knowledge. *ISPRS Journal of Photogrammetry and Remote Sensing*, 79:240–251, 2013.

[32] Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale Feature Extraction on Point-Sampled Surfaces. In *Computer graphics forum*, volume 22, pages 281–289. Wiley Online Library, 2003.

[33] Nicolas Brodu and Dimitri Lague. 3d terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology. *ISPRS Journal of Photogrammetry and Remote Sensing*, 68:121–134, 2012.

[34] Ch Hug, P Krzystek, and W Fuchs. Advanced lidar data processing with lastools. In *XXth ISPRS Congress*, pages 12–23, 2004.

[35] Franz Rottensteiner, Gunho Sohn, Markus Gerke, and Jan Dirk Wegner. Isprs test project on urban classification and 3d building reconstruction. *Commission III-Photogrammetric Computer Vision and Image Analysis, Working Group III/4-3D Scene Analysis*, pages 1–17, 2013.

[36] Franz Rottensteiner, Gunho Sohn, Jaewook Jung, M Gerke, Caroline Baillard, Sebastien Benitez, and Uwe Breitkopf. The isprs benchmark on urban object classification and 3d building reconstruction. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences I-3*, pages 293–298, 2012.

[37] Mohammad Awrangjeb and Clive S Fraser. Automatic segmentation of raw lidar data for extraction of building roofs. *Remote Sensing*, 6(5):3716–3751, 2014.

[38] Kristin A Cook and James J Thomas. Illuminating the path: The research and development agenda for visual analytics. 2005.

[39] NVIDIA. *NVIDIA CUDA Programming Guide 2.0*. 2008.

[40] H. Butler, M. Loskot, P. Vachon, M. Vales, and F. Warmerdam. libLAS: ASPRS LAS LiDAR Data Toolkit, 2011.

[41] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[42] Andreas Fabri and Sylvain Pion. Cgal: The computational geometry algorithms library. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 538–539. ACM, 2009.

[43] John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips. GPU computing. *Proceedings of the IEEE*, 96(5):879–899, May 2008.

[44] Howard Butler and Mateus Loskot. Sample Datasets at libLAS - LAS 1.0/1.1/1.2 ASPRS LiDAR data translation toolset, 2013.

[45] Gideon Guy and Gérard Medioni. Inference of Surfaces, 3D Curves, and Junctions from Sparse, Noisy, 3D Data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(11):1265–1277, 1997.

[46] Shengfa Wang, Tingbo Hou, Shuai Li, Zhixun Su, and Hong Qin. Anisotropic Elliptic PDEs for Feature Classification. *Visualization and Computer Graphics, IEEE Transactions on*, 19(10):1606–1618, 2013.

[47] Thomas Schultz and Gordon L Kindlmann. Superquadric Glyphs for Symmetric Second-order Tensors. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1595–1604, 2010.

[48] Andrea Kratz, Cornelia Auer, Markus Stommel, and Ingrid Hotz. Visualization and Analysis of Second-Order Tensors: Moving Beyond the Symmetric Positive-

Definite Case. In *Computer Graphics Forum*, volume 32, pages 49–74. Wiley Online Library, 2013.

[49] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph.*, 26(2):71–78, July 1992.

[50] Stefan Gumhold, Xinlong Wang, and Rob MacLeod. Feature Extraction from Point Clouds. In *Proceedings of 10th international meshing roundtable*, volume 2001. Citeseer, 2001.

[51] Patric Keller, Oliver Kreylos, Bernd Hamann, Louise H. Kellogg, Eric S. Cowgill, Martin Hering-Betram, and Hans Hagen. Extraction of features from high-resolution LiDaR point cloud data. In *Abstract Proceedings of American Geophysical Union (AGU) Fall Meeting 2008, Eos. Trans. AGU 89(53), Fall Meeting Suppl., AGU Meetings Department, Washington D.C.*, number G53B-0636 (poster presentation), 2008.

[52] Philippos Mordohai and Gérard Medioni. Dimensionality Estimation, Manifold Learning and Function Approximation Using Tensor Voting. *The Journal of Machine Learning Research*, 11:411–450, 2010.

[53] Rodrigo Moreno, Miguel Angel Garcia, Domenec Puig, Luis Pizarro, Bernhard Burgeth, and Joachim Weickert. On improving the efficiency of tensor voting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(11):2215–2228, 2011.

[54] Gordon Kindlmann. Superquadric Tensor Glyphs. In *Proceedings of the Sixth Joint Eurographics-IEEE TCVG conference on Visualization*, pages 147–154. Eurographics Association, 2004.

[55] Carl-Fredrik Westin, Sariel Peled, Hakon Gudbjartsson, Ron Kikinis, Ferenc A Jolesz, et al. Geometrical diffusion measures for mri from tensor basis analysis. In *Proceedings of ISMRM*, volume 97, page 1742, 1997.

[56] Marc Levoy. The Stanford 3D Scanning Repository, 2013.

[57] Jixian Zhang, Xiangguo Lin, and Xiaogang Ning. SVM-based classification of segmented airborne LiDAR point clouds in urban areas. *Remote Sensing*, 5(8):3749–3775, 2013.