Ingenieurfakultät Bau Geo Umwelt
Photogrammetrie und Fernerkundung

Reconstruction of building objects from point clouds of built environment and construction sites

Yusheng Xu

Vollständiger Abdruck der von der Ingenieurfakultät Bau Geo Umwelt der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:	Prof. Dr. -Ing. André Borrmann
Prüfer der Dissertation:	1. Prof. Dr. -Ing. Uwe Stilla
	2. Prof. Dr. techn. Norbert Pfeifer Technische Universität Wien
	3. Prof. Dr. habil. Hans-Gerd Maas Technische Universität Dresden

Die Dissertation wurde am DD.MM.JJJJ bei der Technischen Universität München eingereicht und durch die Ingenieurfakultät Bau Geo Umwelt am DD.MM.JJJJ angenommen.

Abstract

Point clouds acquired via laser scanning and stereo matching of images are becoming popular dataset used in a wide variety of applications, proved to be one of the most suitable sources for the mapping the urban scene. The measured 3D points naturally equipped with spatial coordinates of geometric surfaces, which can significantly simplify the surface modeling and geometric reconstruction processes. However, raw point clouds usually contain no semantic, geometric, nor topological information of objects, which are deemed counterproductive to further applications. Thus, it is mandatory to transform those 3D measurements into a high-level, semantically rich representation. Here, the reconstruction of 3D objects in the scene of man-made infrastructure recovering the geometry of objects and labeling the reconstructed facilities is one of the solutions to solve this task.

The principal goal of this thesis is to discuss and analyze the use of 3D point clouds in the reconstruction of building elements from construction sites and urban scenes, with proposed algorithms and methods in the fields of segmentation, classification, and modeling presented. The contribution lies on three major aspects, including the development and use of voxel-based data structures, the featuring engineering for representing the local geometry of points, and the application of local and global graphical models for optimization problems.

First, the voxel structure is introduced to organize the entire point cloud, involving voxel grid-based filtering, octree-based voxelization, and over-segmented supervoxelization. The voxel structure enables indexing the unorganized point cloud with regular cubic structure. As down-sampling, it simplifies the dataset and suppresses the outliers and uneven density of points with grid-based representations, and defines neighborhood relations of voxels as well as points within them. While the over-segmented supervoxelization provides clusters of points with the preservation of boundaries of objects, determining appropriate neighborhoods for points. Second, the robust estimator and detrending strategy are introduced to create a linear feature enhance shape descriptor and a feature extraction method for estimating the unary features representing the local geometry for classification purpose. Besides, the adoption of perceptual grouping laws for assessing binary features to identify the connections between basic elements (i.e., voxels or supervoxels), enabling a purely geometric and unsupervised solution for segmentation, is also proposed. At last, both local and global graphical models are designed and applied to the segmentation and classification tasks, which encapsulate the contextual information and the interaction between the object and its neighborhoods. By optimizing the weighted graphical model, we can provide robust segmentation of points and refine the classification results with initial soft labels.

Experimental results reveal that our proposed framework combining the voxel-based data structures, designed features, and the optimization of graphical models can provide efficient and effective segmentation of points acquired from complex urban scenes and show considerable improvement to the classification results, which facilitate the further geometric modeling of objects and indicate satisfying and promising results.

Kurzfassung

Punktwolken, die durch Laserscanning und Stereoabgleich von Bildern erfasst werden, werden zu einem beliebten Datensatz, der in einer Vielzahl von Anwendungen zum Einsatz kommt, und erwies sich als eine der geeignetsten Quellen für die Kartierung der städtischen Szene. Die gemessenen 3D-Punkte sind natürlich mit räumlichen Koordinaten von geometrischen Flächen ausgestattet, was die Oberflächenmodellierung und die geometrischen Rekonstruktionsprozesse erheblich vereinfachen kann. Rohe Punktwolken enthalten jedoch in der Regel keine semantischen, geometrischen oder topologischen Informationen über Objekte, die für weitere Anwendungen als kontraproduktiv gelten. Daher ist es zwingend erforderlich, diese 3D-Messungen in eine semantisch reichhaltige Darstellung auf hoher Ebene umzuwandeln. Die Rekonstruktion von 3D-Objekten in der Szene einer künstlichen Infrastruktur, bei der die Geometrie von Objekten wiederhergestellt und die rekonstruierten Einrichtungen gekennzeichnet werden, ist eine der Lösungen, um diese Aufgabe zu lösen.

Das Hauptziel dieser Arbeit ist es, die Verwendung von Punktwolken bei der Rekonstruktion von Bauelementen von Baustellen und urbanen Szenen zu diskutieren und zu analysieren. Dabei werden Algorithmen und Methoden in den Bereichen Segmentierung, Klassifizierung und Modellierung vorgestellt. Der Beitrag bezieht sich auf drei Hauptaspekte: die Entwicklung und Verwendung von Voxel-basierten Datenstrukturen, das Feature-Engineering zur Darstellung der lokalen Geometrie von Punkten und die Anwendung lokaler und globaler grafischer Modelle für Optimierungsprobleme. Erstens wird die Voxelstruktur eingeführt, um die gesamte Punktwolke zu organisieren, einschliesslich Voxelgitter-basierter Filterung, Octree-basierter Voxelisierung und übersegmentierter Supervoxelisierung. Die Voxel-Struktur ermöglicht die Indizierung unorganisierter Punkte mit regelmässiger kubischer Struktur. Beim Downsampling vereinfacht es das Dataset, unterdrückt Ausreisser und ungleichmässige Punktdichten mit gitterbasierten Darstellungen und definiert Nachbarschaftsbeziehungen von Voxeln sowie Punkte innerhalb von ihnen. Während die übersegmentierte Supervoxelisierung Punktclustern mit der Beibehaltung der Objektgrenzen zur Verfügung stellt, werden geeignete Nachbarschaften für Punkte festgelegt. Zweitens werden der robuste Schätzer und die Detrending-Strategie eingeführt, um einen linearen Feature-Deskriptor und ein Feature-Extraktionsverfahren zu erstellen, das die unären Features der lokalen Geometrie für Klassifizierungszwecke schätzt. Ausserdem wird die Annahme von perzeptuellen Gruppierungsgesetzen zur Bewertung binärer Merkmale zur Identifizierung der Verbindungen zwischen Voxeln oder Supervoxeln vorgeschlagen, wodurch eine rein geometrische und nicht überwachte Lösung für die Segmentierung ermöglicht wird. Zum Schluss werden sowohl lokale als auch globale grafische Modelle entworfen und auf die Segmentierungs- und Klassifizierungsaufgaben angewendet, die die kontextuellen Informationen und die Interaktion zwischen dem Objekt und seinen Nachbarn kapseln. Durch die Optimierung des gewichteten grafischen Modells können wir eine robuste Segmentierung von Punkten bereitstellen und die Klassifizierungsergebnisse mit anfänglichen Softlabels verfeinern.

Die experimentellen Ergebnisse zeigen, dass unser vorgeschlagenes Framework, das die Voxelbasierten Datenstrukturen, die entworfenen Merkmale und die Optimierung grafischer Modelle kombiniert, eine effiziente und effektive Segmentierung von Punkten ermöglicht, die aus komplexen städtischen Szenen gewonnen wurden, und eine deutliche Verbesserung der Klassifikationsergebnisse zeigt, die das weitere Vorgehen erleichtern geometrische Modellierung von Objekten und zeigt zufriedenstellende und vielversprechende Ergebnisse.

Contents

List of Abbreviations

Abbreviation	Description	Page
ALS	Aerial Laser Scanning	17
BIM	Building Informaiton Model	15
CRF	Conditional Random Field	23
DON	Difference Of Normal	87
FPFH	Fast Point Feature Histograms	26
GGBC	Global Graph-Based Clustering	91
HT	Hough Transform	22
IoU	Intersection over Union	104
KNN	K-Nearest Neighbor	73
LCCP	Local Convex Connected Patches	107
LiDAR	Light Detection And Ranging	16
LRF/LRA	Local Reference Frame / Local Reference Axis	25
LSSHOT	Linear Straight Signature of Histograms of Orientations	67
MLESAC	Maximum Likelihood Estimation SAmple Consensus	28
MLS	Mobile Laser Scanning	17
MRF	Markov Random Fields	23
RF	Random Forest	16
PCA	Principle Component Analysis	16
PCL	Point Cloud Library	52
RANSAC	RANdom SAmple Consensus	16
RG	Region Growing	23
SAC	SAmple Consensus	39
SfM	Structure from Motion	96
SHOT	Signature of Histograms of Orientations	26
SLRF	Semi-Local Reference Frame	68
SVGS	Supervoxel and Graph-based Segmentation	47
SVM	Support Vector Machine	27
TLS	Terrestrial Laser Scanning	15
UHCG	Unsupervised Hierarchical Clustering of Gestalt principles	57
VIS	Voxel-based Incremental Segmentation	116
VCCS	Voxel Cloud Connectivity Segmentation	34
VGS	Voxel and Graph-based Segmentation	47
VPM	Voxel-based Probabilistic Model	57

List of Figures

List of Tables

1 Introduction

1.1 Motivation

In the fields of Architecture, Engineering and Construction/Facility Management (AEC/FM), the need for efficient and accurate progress monitoring of construction sites and change detection in the urban scene has increased in recent decades, driven by popular specialized applications in work progress tracking, productivity improvement, quality control, security assurance, accident investigation, collaborative communications[Bosché, 2010; Turkan et al., 2012]. Conventional approaches for progress tracking and change detection largely depends on visual inspections and require extensive manual collections of data and analysis of various documents. Such progress monitoring methods, therefore, not only rely heavily on the personal skills and experiences of professionals but also are fairly time-consuming. To solve this problem, much attention has been devoted to automatic construction monitoring via 2D imaging, photogrammetry, and Terrestrial Laser Scanning (TLS) [Tang et al., 2010; Turkan et al., 2012; Pătrăucean et al., 2015]. Recently, point clouds acquired via laser scanning and stereo matching of images are becoming popular dataset used for a wide variety of applications such as urban planning, 3D modeling, virtual reality, civil engineering, and forest monitoring [Vosselman & Maas, 2010]. Especially when mapping large-scale urban area, point clouds have been proved to be one of the most suitable data sources, because measured 3D points have spatial coordinates of geometric surfaces directly, which can greatly simplify the surface modeling and geometric reconstruction processes. Benefiting from the widely used point cloud dataset, at the moment the reconstruction of 3D man-made infrastructure in the urban scene, for example the reconstruction of as-built Building Information Model (BIM), using point clouds is increasingly widely utilized and becoming another essential alternative solution for the accurate progress monitoring of construction sites and change detection in the urban scene [Tang et al., 2010; Pătrăucean et al., 2015; Tuttas et al., 2017]. However, raw point cloud datasets usually contain numerous secondary and temporary objects, for example, temporal components, which are deemed counterproductive to the reconstruction work. Besides, the measured points can provide only 3D coordinates without topological and semantic information. In Fig. 1.1, we illustrate the gap between the acquired photogrammetric point cloud and the expected output of the semantically rich 3D model of objects in the scene of man-made infrastructures. As seen from the figure, we can find that the creation of the 3D model of an object in the scene of man-made infrastructure involves not only measuring the 3D points but not recovering the geometry of objects and labeling the reconstructed facility. In other words, we need to transform those 3D measurements into a high-level, semantically rich representation. During the entire process, the following questions should be resolved:

- How to interpret the scene with semantic information?
- How to extract the object of our interest from the scene?
- How to represent the object with geometric/semantic models?

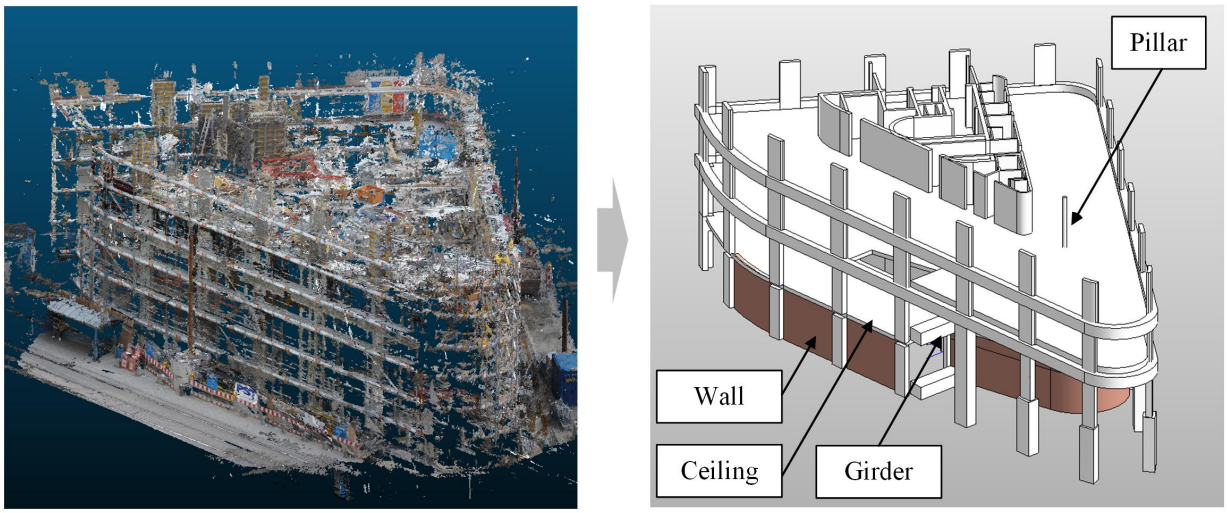


Figure 1.1: Reconstruction of the object with semantically rich 3D models from point clouds (Example using as-built BIM [Tuttas et al., 2017]).

To achieve this task, numerous attempts and contributions have been done in the field of shape or object detection and reconstruction from the point cloud of the man-made infrastructures. For example, in [Schnabel et al., 2007], a RANdom SAmple Consensus (RANSAC)-based algorithm is proposed to provide a valid solution for extracting several types of shapes from an oriented point cloud (i.e., point cloud with oriented surface normals). Employing statistical analysis and persistent histogram features estimation, in Rusu [2010], geometric shapes of objects are obtained in a household environment from the point cloud through semantic 3D object maps. Okorn et al. proposed an approach using the HT to extract planar segments from the point cloud [Okorn et al., 2010]. Moreover, in Bosché [2010] the object recognition is performed based on a threshold on the ratio of the covered area to the entire surface of the object. Additionally, Rothermel et al. [2012] reviewed the local supervised classifiers and statistical models for the object extraction from Light Detection And Ranging (LiDAR) points in urban areas. In [Niemeyer et al., 2014], Niemeyer detected buildings from point clouds via integrating the Random Forest (RF) classifier into the conditional random fields framework. In [Niemeyer et al., 2014] and [Lari & Habib, 2014], the extraction of planar and linear features from laser scanning data is conducted by utilizing the Principal Component Analysis (PCA) and its variations and extensions. Based on the previous work, Polewski et al. [2015] demonstrated that the local 3D feature descriptors and local supervised classifiers could be used to detect segments of fallen trees in LiDAR point clouds efficiently. Among all the mentioned ideas and approaches, the ones based on local surface features extracted by various 3D feature descriptors show a promising prospect, because the unique and representative features they extract are distinctive when recognizing different types of objects and they are commonly not affected by the scale, rotation, and translation factors [Guo et al., 2014]. Accordingly, 3D feature descriptors can play a vital role in the task of object recognition, especially in the condition of the 3D reconstruction owing to the complex structures and background. However, all the publication mentioned above only partially solved the problem of reconstructing 3D models in the complex scene, focusing on different aspects (e.g., classification, segmentation, modeling, or optimization process). To have an overview of the current state of the art, we will give a review of related work in the following section before we state the objective and contribution of our work.

1.2 State of the art

A wide variety of strategies approaches, and algorithms for the reconstruction of 3D building objects from point clouds have been reported in the past decades. For reconstructing objects from the 3D point clouds, according to the order of grouping and labeling procedures, the major strategies can be subdivided into two types: (I) Grouping-based strategy and (II) Labeling-based strategy. The difference between workflows of using these two strategies is given in Fig. 1.2.

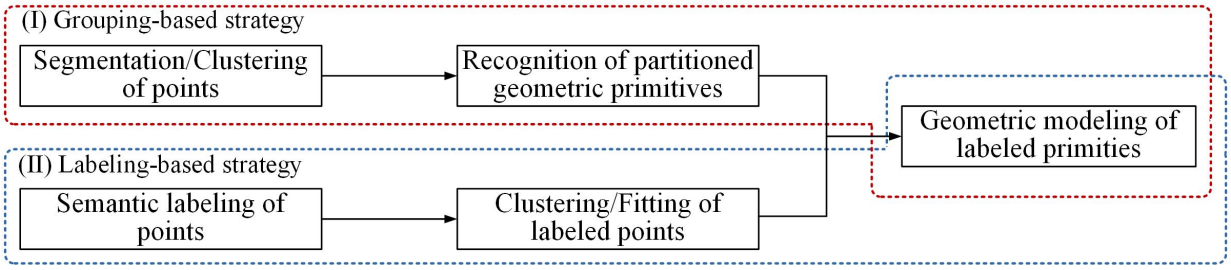


Figure 1.2: Different strategies for object reconstruction from point clouds.

For the grouping-based strategy, the primary procedure is the segmentation of points to get primitives with the common attribute or geometric properties, then the recognition of the partitioned primitives is achieved by providing semantic labels, and finally, the geometric modeling of the labeled primitives is enforced. While the labeling-based strategy will conduct the semantic labeling directly on the points, then clustering the labeled points into geometric primitives, and finally apply the geometric modeling to the cluster of labeled points. However, the core processing steps of these two strategies always cannot avoid the segmentation, classification, and geometric modeling steps. In some cases, there is also no clear boundary between these three core steps, for example, the model-fitting step can be used for both the segmentation of the scene and the classification of different objects, providing parametric models simultaneously. In Table. 1.1 , we provide a statistic of selected representative publications for the topic about the reconstruction of man-made infrastructures from urban scenes, with the type of strategies, sensors for data acquisition, processing elements, tasks, core methods listed. In this table, the publication of Grouping-based strategy (i.e., Type I) represents the publication using the grouping-based strategy, whereas the one of Labeling-based strategy (i.e., Type II) represents the publication using the labeling-based strategy. For the various task, the 3D data can be acquired through LiDAR sensor (i.e., Aerial Laser Scanning (ALS) and Mobile Laser Scanning (MLS)), Multi-view stereo vision, depth camera, and TomoSAR. However, when applying different algorithms and methods, the 3D data are used in the form of original points, superpoints (e.g., pre-clustered points), pixels (for depth image), voxels, and supervoxels (e.g., pre-clustered voxels), which can enhance the performance of adopted methods by organizing the 3D data with specific data structures. Based on the reviewed literature in this table, to give a detailed analysis from the methodology aspect, we provide detailed reviews and discussions concerning the segmentation, classification, and geometric modeling algorithms and methods in the following sections.

Table 1.1: Selected studies of reconstructing built environment and construction sites.

Task	Type	Sensor	Element	Segmentation	Classification	Modeling	Publications
3D metal structure reconstruction	I	TLS	Points	Hough Transform	Knowledge-based classification	Parametric modeling	Cabaleiro et al. [2014]
3D metal structure reconstruction	I	TLS	Points	Region Growing-based octree for plane extraction		Surface modeling with boundary representation by cross-section analysis	Laefer & Truong-Hong [2017]
3D structural component recognition	I	TLS	Points	Hough Transform	Clustering with similar normal vectors and close proximity		Yeung et al. [2014]
3D structural component recognition	I	Stereo vision	Points	Color-based segmentation		Parametric modeling by shape matching	Son & Kim [2010]
Building reconstruction	I	Stereo vision	Points	RANSAC for model fitting		Surface modeling with energy minimization with intersecting planar faces	Nan & Wonka [2017]
Building reconstruction	I	ALS	Points	Unsupervised clustering	Energy-based boundary extraction	Surface modeling with boundary representation for roofs	Poullis [2013]
Building reconstruction	I	ALS	Points	Sub-surface growing	Iterative rule-based feature recognition	Surface modeling with boundary representation	Kada & Wichmann [2013]
Building reconstruction	I	TLS	Points	Segmentation via plane and boundary detection	Rule-based classification	Shape modeling by geometry size fitting	Wang et al. [2015a]
Building reconstruction	I	TLS	Points	Planar surface growing	Knowledge-based classification	Surface modeling with convex polygon and concave polygon fitting	Pu & Vosselman [2009]
Building reconstruction	I	TLS	Points	Vertical projection and line fitting with Hough Transform, plane detection with Region Growing	Context information of relationships + CRF	Surface modeling with boundary representation	Huber et al. [2011]
Building reconstruction	I	TLS	Points	RANSAC for model fitting	Manually classification	Surface modeling with boundary representation by boundary tracing	Jung et al. [2014]
Building reconstruction	I	TLS	Voxels	Region Growing	Contextual feature+Stacked learning	Surface modeling with opening detection	Xiong et al. [2013]
Building reconstruction (facades)	I	TLS	Points	Gaussian Image based plane extraction	Topological graph+Structure encoding tree	Parametric modeling for planes	Hao & Wang [2016]
Building reconstruction (facades)	I	TLS	Points	Depth plane layer based segmentation	ScSPM features + SVM	Surface modeling with boundary representation	Li et al. [2017]

Building reconstruction (facades)	I	RGB and MLS	Supervoxels	Rule-based segmentation	3D shape descriptors+Boosted decision tree		Babahajiani et al. [2017]
Building reconstruction (roof)	I	ALS	Points	"Blob" structure detection	Markov Chain Monte Carlo	Shape modeling from predefined roof library	Huang et al. [2013]
Building elements detection	I	TLS	Points	RANSAC for model fitting			Nahangi et al. [2015]
Building elements recognition	I	TLS	Points	Registration between objects and CAD models		Shape modeling by matching CAD database	Bosché [2010]
Building elements recognition	I	TLS	Points	Localized RANSAC for model fitting	Knowledge-based classification		Schnabel et al. [2007]
Building elements recognition	I	ALS	Points	Normal vector clustering and RANSAC	Fourier transform of repetitive structures	Parametric modeling by rectangular shape fitting	Tuttas & Stilla [2011]
Building elements recognition	I	ALS	Points	Region Growing	Lattice histogram-based classification		Mesolongitis & Stamos [2012]
Indoor scene reconstruction	I	MLS	Points	Line and plane detection	Graph-Cut	Surface modeling with cell decomposition	Ochmann et al. [2016]
Indoor scene reconstruction	I	TLS	Points	Extended Gaussian Image and Region Growing	Geometric features + CRF	Surface modeling with triangulation	Rusu et al. [2009c]
Indoor scene reconstruction	I	TLS	Points	Horizontal slicing and line fitting with Hough Transform	Ray-casting intersections and adjacency+Graph-Cut	Surface modeling with cell decomposition	Oesau et al. [2014]
Indoor scene reconstruction	I	TLS	Points	Horizontal slicing and RANSAC for line fitting	Ray-casting intersections and adjacency+Graph-Cut	Surface modeling with cell decomposition	Wang et al. [2017]
Indoor scene reconstruction	I	TLS	Points	Horizontal slicing, Plane projection and line fitting	Overlap Analysis	Surface modeling with cell decomposition	Li et al. [2018]
Indoor scene reconstruction	I	TLS	Points	Region Growing	MLESAC for plane identification	Surface modeling with polygonal fitting plane	Rusu et al. [2008]
Indoor scene reconstruction	I	TLS	Points	RANSAC for primitive fitting and Graph-Cut	Segment graph + multiple kernel learning SVM		Shi et al. [2016]
Indoor scene reconstruction	I	RGB-D	Supervoxels	Supervoxel oversegmentation	Decision Tree Field and Regression Tree Field learned in CRF		Kahler & Reid [2013]
Industrial instrumentation	I	TLS	Points	Curvature-based clustering	Knowledge-based classification	Shape modeling by matching CAD database	Son et al. [2015]
Outdoor object extraction	I	MLS	Voxels	Connected Components-based Region Growing	Local shape descriptor + SVM		Lehtomäki et al. [2016]
Outdoor object extraction	I	MLS	Points	Min-Gut	Shape descriptor+KNN, RF, SVM		Golovinskiy et al. [2009]

Outdoor object extraction	I	MLS	Supervoxels	Multi-scale supervoxel over-segmentation	Rule-based classification		Yang et al. [2015]
Outdoor object extraction	I	TLS	Superpoints	L0 Graph cut-based segmentation	PointNet++		Landrieu & Simonovsky [2018]
Pipeline-plant reconstruction	I	TLS	Voxels	Connectivity graph construction and Region Growing	Knowledge-based classification		Erdős et al. [2015]
Pipeline-plant reconstruction	I	TLS	Points	Gaussian Image	Detection of circles in the projection plane		Liu et al. [2013]
Building reconstruction	II	MLS	Superpoints	RANSAC for planar primitive fitting	Geometric feature + Adaboost	Connection Graph based hierarchical representation	Lin et al. [2013]
Building reconstruction	II	Stereo vision	Points	MRF-based segmentation	Statistical analysis + Regularized MRF formulation and Graph-Cut	Surface modeling with polygonal mesh for footprint	Li et al. [2016]
Building reconstruction	II	ALS	Points	Line fitting and Region Growing for plane	Discriminative feature + Graph-Cut	Shape modeling by the arrangements of geometric 3D-primitives and mesh patches	Lafarge & Mallet [2012]
Building reconstruction (roof)	II	ALS	Points	Coplanarity-based clustering	Height and planarity-based classification	Parametric modeling	Awrangjeb & Fraser [2013]
Building reconstruction (roof)	II	ALS	Points	Fuzzy k-means clustering	Local flattening and Voronoi neighbors-based classification		Sampath & Shan [2010]
Building reconstruction (facades)	II	Tomo SAR	Points	K-means clustering	Scatter density based classification	Parametric modelling with weighted least square	Zhu & Shahzad [2014]
Building reconstruction (facades)	II	Tomo SAR	Points	Gaussian image based mean shift	Scatter density based classification	Parametric modeling with MLESAC fitting with general polynomial equation	Shahzad & Zhu [2015]
Building element recognition	II	TLS	Points	Region Growing	PCA+ Rule-based classification	Parametric modeling with RANSAC model fitting	Sanchez & Zakhor [2012]
Construction sites reconstruction	II	TLS	Points	Iterative complete linkage clustering	Robust PCA+Rule-based classification	Parametric modeling	Maalek et al. [2018]
Construction sites reconstruction	II	Stereo vision	Points	Horizontal slicing, Plane projection and Region Growing	LSSHOT + Random Forest	Surface modeling with boundary representation	Xu et al. [2018c]
Pipeline-plant reconstruction	II	TLS	Points	Skeleton extrusion	Voronoi diagram-based linear skeleton identification	Parametric modeling with cylindrical fitting	Lee et al. [2013]

Outdoor object extraction	II	MLS	Points	Rules-based segmentation	Eigenvalue-based feature + SVM		Yang & Dong [2013]
Indoor object recognition	II	RGB-D	Pixels	Voxel-based CRF			Kim et al. [2013]
Indoor object recognition	II	RGB-D	Pixels		Energy formulation for joint global surface reconstruction and semantic classification		Hane et al. [2013]

1.2.1 Segmentation of 3D point clouds

The segmentation of point clouds, partitioning 3D points into multiple homogeneous regions having one or several common characteristics [Grilli et al., 2017], has been studied and explored for decades, with methods and algorithms in different disciplines exploited, including computer vision, computer graphics, computational geometry, robotics, photogrammetry, remote sensing, machine learning, and statistics.

Summarily, the relevant point cloud segmentation approaches can be classified into two major categories: the geometric-based methods and the attribute-based methods. The geometric-based methods group points according to the geometric homogeneity of the surfaces or structures that the points belong to, whereas the attribute-based methods use intensities or color information of points to cluster them into segments sharing same or similar attribute information. Both of these two kinds of methods have their merits and demerits. However, the intensity or color information is not always reliable because the quality of them mainly resorts to the recording technology of the sensor. On the other hand, the intensity and color information can easily be affected by the materials of surfaces, the illumination of objects, and the light conditions. Especially in urban areas, varying sunlight conditions and sophisticated environment of objects having similar materials, colors and illuminations make the attribute-based segmentation unreliable. Thus, in many cases of parsing building scenes, we have to face a purely geometric segmentation problem. This is also the reason why we focus on the geometric-based segmentation methods in this work. Roughly speaking, the geometric-based segmentation methods can be subdivided into three different categories: (i) the model-based methods, (ii) the region growing-based methods, (iii) the clustering-based methods, and (iV) Pre-clustering-based methods [Vo et al., 2015].

Model-based segmentation

The model-based methods evaluate connected points according to their geometric features (e.g., spatial positions and normal vectors) in a local or global scale through certain geometric models. Points meeting the criteria of fitting the same geometric model (either in the spatial or parametric domain) are extracted from the point cloud as single individual segment. To be specific, model-based methods mainly involve two extensively used approaches: the parameter domain-based approaches and the spatial domain-based approaches. The parameter domain-based approaches fit candidate points of objects according to the mathematically transformed geometric models in the parameter domain. The 3D Hough Transform (HT) and its variants are typical cases. HT is a voting strategy-based algorithm carried out in a parameter space transformed according to the geometric expression, with points of the object selected by the local maxima in an accumulator space. The HT has been used in segmentation by detecting planes [Vosselman et al., 2004], cylinders [Tarsha-Kurdi et al., 2007], and spheres [Rabbani et al., 2006] in the parameter domain. Besides, other similar methods such as Gaussian map [Liu & Xiong, 2008] and tensor voting [Schuster, 2004] can also be classified into this category since they also use the voting and accumulating strategy in the parameter space. Whereas the spatial domain-based ones directly estimate the optimal parameters of geometric models from points in the spatial domain. The optimal parameters are normally calculated by the use of robust estimators and least square-based algorithms. RANSAC and its extensions are the most popular robust estimators that used for shape fitting [Schnabel et al., 2007; Chen et al., 2012, 2014], which can extract shape primitives from the point cloud contaminated by noise or outliers. As for the use of least square-based algorithms, the least square fitting and its variations are utilized to identify surfaces and geometric primitives in [Marshall et al., 2001], but their work also points out that fitting higher order surfaces can be problematic and computationally expensive. The model-based methods are deemed to be robust to the noise and outliers and provide geometric models of segments simultaneously.

Nevertheless, the model-based methods require normally a large computational cost caused by the iteration process of using robust estimators or voting process, leading to high memory consumption [Vo et al., 2015]. Also, challenges arise as model-based methods can hardly deal with objects or surfaces that their representations have no explicit mathematical formulations like the irregular curved surface.

Region growing-based segmentation

The Region Growing (RG)-based methods are another alternative. They are implemented by an iterative process examining nearby points in the region of seed and judging whether they belong to the region of this seed or not. The growing criteria and the selection of seeds are two influential factors for this kind of methods. The consistency of normal vectors [Tóvári & Pfeifer, 2005], the smoothness of surface [Rabbani et al., 2006], and the curvatures of points [Besl & Jain, 1988] are commonly used growing criteria. Recently, in [Nurunnabi et al., 2012], the PCA-based local features are adopted as growing criteria for their distinctiveness. As for the selection of seeds, the density of seeds determines the granularity of segments while the location of seeds significantly affect the quality of segments. Normally, regions with the smallest curvature [Nurunnabi et al., 2012] or surfaces with the minimal residual of the plane fitting [Rabbani et al., 2006] are frequently identified as seeds, so that the location of seeds can avoid boundaries and edges. For instance, the seeds located on the edge or the corner of the surfaces will yield over-segmentation. Besides, over-segmentation can also easily occur for large curved objects (e.g., pipes with an extended radius elbow joint) [Su et al., 2016]. In theory, the region growing-based methods can preserve boundaries and edges of surfaces and objects well, but they are sensitive to noise and outliers. However, the computation of k-nearest neighbors for the growing candidates leads to a high computational complexity, which will primarily increase the computational cost.

Clustering-based segmentation

The last major category is the clustering-based segmentation. This kind of methods examines nearby points in a defined neighborhood by the proximity or similarity between them, according to their geometric characteristics and spatial coordinates. Points having proximity or similarity meeting the acceptable threshold will be identified as connected points. All the connected points will be aggregated into one cluster. Euclidean distance [Aldoma et al., 2012], normal vector [Vo et al., 2015], and density [Lu et al., 2016; Aljumaily et al., 2017] are representative instances that used as criteria for clustering. For the clustering ways, k-means [Morsdorf et al., 2003], mean-shift [Yao et al., 2009], and connected relations [Stein et al., 2014] are mostly adopted ones. Unlike region growing-based methods, the clustering-based ones require no selection of seeds. The computational cost of the clustering methods lies in the complexity of calculating the similarity or proximity as well as the optimization of cost functions. To develop a robust segmentation method, multiple clustering criteria are intensively applied, which will greatly increase the computational cost. Besides, the setting of optimal clustering thresholds is also influential to the granularity of segmented clusters. In the field of computer vision, the clustering of points is also formulated as graph construction and partitioning problems. The graph model can explicitly represent points with a mathematical sound structure [Peng et al., 2013], utilizing context for deducing hidden information from given observations [Yao et al., 2010]. Examples of such methods include the graph-based methods such as Normalized Cuts [Shi & Malik, 2000], Min Cuts [Golovinskiy & Funk, 2009], and Graph-based Segmentation [Felzenszwalb & Huttenlocher, 2004],[Green & Grobler, 2015] and the Markov-based approaches like the Markov Random Fields (MRF) [Hackel et al., 2016a] or Conditional Random Fields (CRF) [Rusu et al., 2009a]. For the graph-based methods, a large topology radius of the constructed graph can provide better results in segmen-

tation, but a dense and large graph yields a heavier computational burden as well [Cour et al., 2005].

Additionally, instead of using points as basic units, the patch-based segmentation methods are drawing increased attention, in which 3D patches consisting of points are used as essential elements, for example, voxels [Wang & Tseng, 2011; Truong-Hong et al., 2012; Wu et al., 2013], slices [Zolanvari & Laefer, 2016], and planar fragments [Vosselman et al., 2017]. The regular 3D voxelization is the most commonly used approach to cluster point clouds into small patches. In [Vo et al., 2015], the octree structure and the region growing process are combined for the fast surface patch segmentation. Similarly, in [Su et al., 2016], the octree-based voxel structure combined with graph-based splitting is applied to segment cylindrical objects in industrial scenes. In the work of [Vosselman et al., 2017], planar fragments detected by RANSAC or 3D HT are selected as seeds in the surface growing process for extracting entire planar surfaces in the scene. The use of patch structure can significantly reduce the computation cost [Yang et al., 2015] and suppress the adverse effect of outliers and different point densities. Even so, the resolution of patches will apparently affect the accuracy of segmentation results and the preservation of details.

Pre-clustering-based segmentation

To segment the point cloud more in a more efficient way, recently, the pre-clustering-based segmentation has been developed and frequently adopted. The use of supervoxel structure is one of the representatives in this kind of methods, which will initially cluster points into over-segmented patches (e.g., voxels or supervoxels) with pre-defined structures or rules. The supervoxel strategy is generated on the basic voxel structure, using local k-means clustering [Papon et al., 2013], weighted distance [Yang et al., 2015], link-chain [Aijazi et al., 2013] and so on, which better preserve the boundary features of segments and further improve the computation efficiency. However, the supervoxel method is merely an over-segmentation of data, how to cluster over-segmented patches into complete segments is still a challenging task. Generally, there are two major ways can be used to address this problem. The first one is the supervised classification based approaches. With the voxel clusters or supervoxels achieved, geometric features [Aijazi et al., 2013; Plaza-Leiva et al., 2017; Aijazi et al., 2014], spectral information [Ramiya et al., 2016],or colors [Yang et al., 2015] can be extracted using points allocated in one voxel cluster or supervoxel. Considering that the use of voxel clusters or supervoxels has already pre-clustered the points having homogeneous characteristics, they can easily avoid the conundrum of choosing an appropriate neighborhood for estimating the features, because the edges of a supervoxel or voxel cluster has already been defined during the pre-clustering step, with isolated points and disturbances being removed. Besides, benefitting from the strength of using supervised learning, the label of each supervoxels of voxel cluster after the classification can have a very high level of accuracy. So that supervoxel or voxel clusters sharing the same labels can be easily clustered into complete segments, with semantic labels obtained simultaneously. However, such supervised methods require accurate and large numbers of training datasets, which take a tremendous amount of manual work and time-consuming. In contrast, another solution is to aggregate voxel clusters or supervoxels with local or global optimization algorithms in an unsupervised way. In Stein et al. [2014], the local convexity is combined with the region growing to cluster supervoxels into complete segments of objects. In Pham et al. [2016a], a global adjacency graph with geometric consistent is constructed by the supervoxel structure. The aggregation of supervoxels is achieved by the energy minimization of the designed binary cost function. Similarly, in Xu et al. [2016b], the supervoxels is arranged into a local adjacency graph with a vicinity of given size. The clustering of supervoxels is managed according to their connections achieved through Markov CLustering (MCL). The advantage of the unsupervised solution is that they do not need training datasets and usually

requires less computational costs. However, they cannot get semantic labels of segments and may have adaptivity problems when dealing with complex structures.

1.2.2 Classification of 3D point clouds

Compared with the segmentation, the classification is the essential step for interpreting the point cloud of 3D scenes, providing semantic labels for individual points or primitives of grouped points. Recent advances in the areas of machine learning and computer vision have also proven that a well-designed 3D point cloud classification approach is eligible for the labeling task in a complex real world. Especially for the building reconstruction extracting building objects from the entire measured 3D scene, the semantic interpretation is a vital step in serving the recognition of points belonging to buildings. To label points or primitives, the classification consists typically of three essential steps [Weinmann et al., 2015], namely (i) the recovery of the local neighborhood for the point or primitive, (ii) the description of the geometry based on 3D information of the local neighborhood, and (iii) the classification of all 3D points based on the respective geometric descriptions. Currently, researchers have reported plenty of achievements for solving problems relating to these three steps. However, there are still plentiful challenges for recognizing building objects from point clouds in complex scenes, for instance, random point sampling, different point density, complex structural component, and different data sources. Moreover, the computational cost should also be taken into consideration when coping with large-scale 3D point clouds. In the following sections, we plan to expound frequently used 3D point cloud classification approaches by comparing aforementioned three major steps.

Recovery of the local neighborhood

The recovery of the neighborhood, assigning each point or primitive of a given pointwise or area-based local neighborhood, is imperative for describing the detailed information of the point or primitive. With different aims, the description of various objective details relies on the local context of all the points within the chosen of the neighborhood. The definitions of neighborhoods can be roughly categorized into two types: single-scale neighborhoods and multi-scale neighborhoods. The first one extracts features from a neighborhood have a fixed size. In contrast, the second one adopts flexible sizes of neighborhoods. Most commonly used single-scale neighborhood definitions are the spherical [Lee & Schenk, 2002] or cylindrical [Filin & Pfeifer, 2005] neighborhood around a key point with Local Reference Frame or Local Reference Axis (LRF or LRA). Instead of fixing the shape of a neighborhood, the neighborhood of one point can also be defined by a fixed number of k nearest neighbors, in which the distance between two points can be either 3D distance [Linsen & Prautzsch, 2001] or 2D projective distance [Niemeyer et al., 2014]. In [Weinmann et al., 2015], the author proposes an approach relying on individually optimized neighborhoods in 3D scenes according to the estimation of entropy in the context. For the multi-scale neighborhoods, in [Dong et al., 2017], they use a feature selection strategy using a multi-scale neighborhood, to improve the performance of feature engineering. The multi-scale neighborhood can be defined as a combination of simple neighborhoods with various shapes and sizes, and identical features from multi-scale neighborhoods are separately extracted to conduct further feature encoding. Another alternative for defining an adaptive neighborhood is to use over-segmented or pre-clustered patches [Sun et al., 2018]. For example, in [Wang et al., 2015c], point-based hierarchical clusters are generated with a Latent Dirichlet Allocation (LDA) model, in which features of clusters are derived in order to classify objects of different sizes. Similarly, in [Yu et al., 2016], the author implements a multi-layer feature generation model consisting of various levels of sub-space in the octree partition structure to detect a specific object (i.e., vehicles).

Description of the local geometry

A description of the local geometry is to encapsulate local geometric information in the area surrounding the investigating point or primitive and encapsulates extracted features into a vector of a histogram [Guo et al., 2014], which is the input to the classifier for labeling inference. A wide variety of feature extraction algorithms have been presented in the last decade. Generally, they can be grouped into two major categories: (i) low-level and (ii) high-level descriptions.

Low-level description consists of only fundamental geometric properties of the neighborhood (e.g., dimensionality) and the spatial arrangement (e.g., the curvature of the surface) of 3D points within a neighborhood [Weinmann et al., 2015]. As a representative, the eigenvalue-based geometry [Jutzi & Gross, 2009; Chehata et al., 2009] is exploited from the tensor of coordinates encoding 3D structures, relating to the 3D covariance matrix derived from the coordinates of all points in a local neighborhood around a single point p. The 3D structure tensor, delineated by three eigenvalues of the tensor, can be regarded as a dimensionality reduction of the local structural information. More specifically, the combination of these three eigenvalues can characterize local features of 1D, 2D, and 3D shape primitives. Here, a set of local 3D shape features such as eigenentropy, scattering, and ominvariance [Weinmann et al., 2015] is developed, which enable a more intuitive representation of volumetric structures [Yao et al., 2017]. It is noteworthy that the features of the low-level description are generally sensitive to the size of the selected neighborhood, an optimal neighborhood size must be found in order to get optimal results. In [Weinmann et al., 2015], thorough investigations of how to increase the distinctiveness of low-level geometric features by changing the size and subset of neighborhoods for feature extraction have been carried out. By optimizing the size of neighborhoods, an adequate low-level geometric feature combination can provide a higher quality classification, with significant improvement achieved in processing time and memory consumption as well.

High-level description involves 3D local shape descriptor for feature extraction, which is a compact representation of points based on the characteristics of their support region (i.e., the neighborhood in our statement) [Tombari et al., 2010]. Commonly used 3D local shape descriptors can be divided into three major categories [Salti et al., 2014; Guo et al., 2014]: the descriptors based on the spatial distribution histograms of points in the neighborhood, the descriptors based on the geometric signature of the point in the local surface, and the descriptors featuring a hybrid structure between spatial distribution histogram and geometric signature. In the first category, the descriptor usually defines a LRF/LRA for the key point, under which a 3D support region is divided into a fixed number of bins. Here, the key point is the one whose features needs to be extracted and normally with the content of rich information. By accumulating the distribution of spatial positions of points in the 3D support region into these bins, a histogram is encoded. Spin Image (SI) [Johnson & Hebert, 1999], 3D Tensor [Mian et al., 2006], and 3D Shape Context (3DSC) [Frome et al., 2004] as well as its variations such as Unique Shape Context (USC) [Tombari et al., 2010] and Cylindrical-3DSC [Polewski et al., 2015] are belong to this category. In the second category, the histogram of the descriptor is generated by encoding descriptive geometric attributes (e.g., normal vectors or curvatures of points) of the key point considering its surrounding points in the 3D support region. Point Feature Histograms (PFH) as well as the improved-efficiency version of Fast Point Feature Histograms (FPFH) [Rusu et al., 2008], Local Surface Patch (LSP) [Chen & Bhanu, 2007], and radius-based surface descriptor (RSD) [Aldoma et al., 2012] are representatives of this type of descriptors. In the last category, the descriptor has a hybrid structure combining the spatial distribution histograms and geometric signatures. The signature of the histogram of orientations (SHOT) [Tombari et al., 2010] is an example of the hybrid descriptor, encoding histograms of the orientations of normal vectors in conjunction with different spatial locations of points in a spherical support region. In the output histogram

of SHOT, both the spatial distribution of points and the local histograms encoding the angles of the normal vector of points are encapsulated.

For either low-level description or high-level description, the definition of the local reference frame and the neighborhood size would significantly influence the performance of the description of local geometry. A repeatable and robust local reference frame should be invariance to rigid transformations, facilitating the extraction of robust features [Salti et al., 2014]. For the size of the neighborhood, a large support region encodes more information, but on the contrary, makes the local shape descriptor more sensitive to occlusion and clutter [Weinmann et al., 2015; Hackel et al., 2016b], which can significantly affect the robustness and accuracy of the extraction of features. For the situation that the point cloud has been contaminated with outliers and noise, the definition of LRF/LRA can be biased, directly influencing the accumulation of spatial positions of points. Moreover, the shape of support region defining a respective range of the neighborhood encapsulating all considered 3D points is also crucial to the representation of features. Developing a 3D local shape descriptor with a robust LRF and specific shaped support region could be a possible solution for achieving improvement of accuracy in specific applications.

Classifier used for labeling inference

Derived feature vectors describing the geometric properties will be finally inputted into the classifier to get semantic labels of the interest points. Currently, the majority of labeling approach prefers to adopt a supervised classification strategy. The principle of supervised classification is to train a classifier by exploiting the training data containing its feature vectors and corresponding data labels. There are several different choices for such kind classification, including two major strategies: (1) point-based classification and (2) segment-based classification. The point-based classification is one of the traditional solutions, in which each point will gain a label during the inference process [Hackel et al., 2016b]. By contrast, for the segment-based classification is also drawing increasing attention due to its advantage of separating individual objects from the scenes simultaneously, in which pre-clustering or segmenting is conducted in advance to generate primitives with homogeneity [Guinard & Landrieu, 2017]. However, regardless of which kind of strategy utilized, classifiers always play an essential role in the final classification performance.

Commonly used classifiers include almost all the supervised learning classifiers such as Support Vector Machines (SVM) [Lodha et al., 2006; Secord & Zakhor, 2006], AdaBoost [Lodha et al., 2007], RF [Chehata et al., 2009], and CRF [Lim & Suter, 2009; Niemeyer et al., 2014]. In the work of [Lodha et al., 2006], a classification method using SVM is proposed for an ALS dataset measured in a large area, and the performance of several variations of SVM algorithm is compared, with comprehensive analysis for results of multi-class classification reported. In [Wei et al., 2012], the author utilizes the AdaBoost classifier combined with the contribution ratio to provide both classification results and measures of the feature relevance. A glance of the comparison of different approaches is obtained via a comparison with the results from RF classifier. In [Chehata et al., 2009] more than twenty features are derived from LiDAR points via the iterative backward elimination of features. Obtained features are divided into five categories. With the help of RF classifier, a reliable labeling result of points in large-scale urban scenes is obtained, with the variable importance for urban scenes estimated. Further investigations are presented in [Guo et al., 2011] with the utilization of RF classifier. In this work, both the importance of features and the strength of the classifier are assessed by permutation accuracy criteria.

In [Lim & Suter, 2009], the implementation of the multi-scale CRF is given to improve the classification accuracy of TLS points. Different from other classifiers, CRF offers an incremental labeling precision over logistic regression. For example, in [Yao et al., 2017], a context-based

labeling method with constrained CRF is developed for an ultra-high point dense MLS dataset in complex urban scenes. Combining the result of the point-wise labeling by RF classification, the strength of CRF are statistically evaluated. Similarly, in [Niemeyer et al., 2014], the author integrates an RF classifier into a CRF framework to improve classification accuracy. Here, both an analysis of the feature importance by RF and the classification of 3D scenes by a hierarchical CRF are carried out In the work of [Dong et al., 2017] a multi-scale neighborhood selection strategy is performed, dividing neighbors into subspaces of three different scales and merely combining features with lower correlations. Better classification performance is consequently obtained, with RF classifier used. Recently, for the reduction of data amount and the improvement of computational efficiency, over-segmentation based on the pre-clustered data structure is often used and combined with the classifier as well [Sun et al., 2018].

1.2.3 Geometric modeling of 3D primitives

Geometric modeling is to generate simplified representations of the 3D shape of the labeled primitives, such as walls, ceilings, and decks. Generally, the type of representation for the output of the reconstruction can be either parametric modeling(e.g., model fitting or matching), surface modeling (e.g., boundary representation) or volumetric modeling (e.g., Constructive Solid Geometry (CSG) representation). According to [Tang et al., 2010], the volumetric parametric methods are the most relevant to as-built BIM reconstruction, since BIMs are usually represented primarily using volumetric presentation. Unfortunately, according to the current state of the art in the area of reverse engineering, the creation of models with parametric and surface representation is more prevalent. This is because that in the volumetric modeling, the solid geometry representing structural components of as-built BIMs cannot be sufficiently achieved due to occlusion from outside with measuring and the lack of topological relations (e.g., the way of connections) between structures. Only observable surfaces can be generated with visible geometric connections and accumulations. Thus, we usually create the geometric representation of objects with parametric and surface modeling and convert them to the volumetric model with additional information (e.g., CAD base) or prior knowledge (e.g., grammars of structural components).

Parametric modeling

The most noticeable feature of parametric modeling is that the geometric representation can be concisely described with mathematical expressions of regular shapes (e.g., cylinder, cubes, or planes). Two major strategies can be used to achieve the parametric modeling. The first one is model fitting, which has been discussed in the review of segmentation techniques, which examine points according to their geometric features (e.g., spatial positions and normal vectors) in a local or global scale through certain geometric models. According to the domain for fitting the mathematical equations, HT [Vosselman et al., 2004; Tarsha-Kurdi et al., 2007; Rabbani et al., 2006] and sample consensus (e.g., RANSAC [Schnabel et al., 2007] and Maximum Likelihood Estimation SAC (MLESAC) [Shahzad & Zhu, 2015]) are representatives. Details of these related methods will not be repeated here. However, as we have stated, challenges arise as model-fitting can hardly deal with objects or surfaces having complex and irregular surfaces. To handle the objects with surfaces of complex mathematical expressions, one of the essential solution is shape matching. This technique is directly related to the application of local 3D shape descriptors we discussed before. For implementing this method, we need a set of reference objects with the known mathematical expression as a library and match the geometric primitives with those references according to the geometric features extracted via shape descriptors (e.g., feature histogram) [Guo et al., 2015]. Then, the corresponding pair of the primitive and the reference will be regarded as

the matched one, and the primitive will be assigned with the model of the reference, including the parameters as well as the mathematical expression.

Surface modeling

Surface modeling is a kind of non-parametric representation. Surface modeling is useful for modeling objects with complex geometries (e.g., incomplete structures during construction). Unlike parametric modeling, the representation of surface modeling can also be quantized with parameters, but there will be no fixed mathematical model to describe the geometric shape of primitives. Instead, the description of surfaces (e.g., with polygons or meshes) is generated adaptively based on the actual geometry of the primitive. The commonly used ways of surface modeling include the boundary-based representation and the mesh-based representation [Tang et al., 2010]. The boundary-based representation will identify the 2D or 3D contour of the primitives, and then symbolize the boundary of the primitive with approximate or implicit lines. These lines form a closed polygon as the description of the surfaces, and a combination of surfaces compose the 3D model. In the work of [Xu et al., 2018c; Wang et al., 2017], the boundaries of primitives (e.g., structural components) are generated by the use of the alpha-shape algorithm and then describe with polygon by the use of Rotating Calipers [Toussaint, 1983] and Cell Decomposition [Kada, 2007], respectively. The modeled surfaces constitute 3D models by the use of energy minimization [Nan & Wonka, 2017], horizontal slicing and vertical projection [Oesau et al., 2014; Wang et al., 2017; Previtali et al., 2018], Stochastic analysis [Huang et al., 2013], or graph edit dictionary [Xiong et al., 2014]. Surface modeling is simple and easy to be implemented. However, they always try to approximate the complex geometry of object with assembling of simple surfaces of polygons (e.g., planes or curved surface). In the process of generating simple surfaces, it is always a trade-off between the details and the abstraction of polygons. Besides, the surface modeling merely focuses on the visible part of the structure, which requires further conversion to a grammar-rich representation of building components. Besides, the mesh-based representation is also a feasible alternative. The mesh-based representation describe the surface by meshes (e.g., triangles [Rusu et al., 2009b], patches [Lafarge & Mallet, 2012] or cubes [Aldoma et al., 2012]). Mesh-based representation is simple and describes complex surfaces accurately. However, it is difficult to parameterize the meshes as no explicit mathematical expressions available.

Volumetric modeling

By now, volumetric modeling has not been fully exploited in creating 3D models of the object from point clouds. It needs typically prior knowledge to assist the inference of the volumetric geometry since only parts of the object are visible in the majority of cases. The commonly used strategy is to assume that objects can be represented by a combination of a small set of volumetric primitives (e.g., planes, superquadrics, and generalized cylinders) [Tang et al., 2010]. These volumetric primitives can be stored with the general format in a CAD database [Son et al., 2015; Bosché, 2010], and then these models of primitives are matched to objects by seeking the best alignment of the model surface. The volumetric primitive have the best fitting score will be chosen to represent the matched part of the object, and a set of such volumetric primitives will form the complete representation of the entire object. Currently, it is also possible to achieve the volumetric representation by converting the surface-based representation [Kolbe, 2009], which could be the mainstream tendency benefiting from the mature techniques from surface modeling.

1.3 Objectives and contributions

In this thesis, we present a frame of using 3D point clouds in the reconstruction of building elements from construction sites and urban scenes, with novel algorithms and methods in the fields of segmentation, classification, and modeling presented. We attempt to introduce techniques and approaches into the reconstruction of building elements and address some specific issues. Subsequently, we will discuss and analyze the pros and cons of the proposed approach as well as the possible extensions and further development in the future.

The tasks included in this thesis consists of two major parts. The first part is about the reconstruction of structural elements in the small-scale construction site. In this part, we will introduce algorithms and methods related to the detection, extraction, and modeling of structural elements from 3D point clouds in a construction site. Whereas, the second part is about the reconstruction of building elements in large-scale urban scenes. In this part, we will present novel algorithms and methods designed for the segmentation, semantic labeling, and modeling of building objects from 3D point clouds in different urban scenes. More specifically, in this work, the explicit aim of the reconstruction of building elements in the small-scale construction site and large-scale urban scenes is to answer the following questions, which have not yet been considered or fully addressed in state of the art literature:

- Is it possible to propose a general strategy and framework for reconstructing objects from construction sites and urban scenes?
- How to organize unstructured point clouds with optimized data structures, which can reduce the number of elements while keep the geometric and topological information of points?
- How to design hand-crafted features for both segmentation and classification tasks, which is robust to noise, outliers, and unevenly distributed densities while being discriminative?
- What is the possible approach for post-processing and refinement of segmentation and classification results, which is reasonable and effective in the 3D scene?

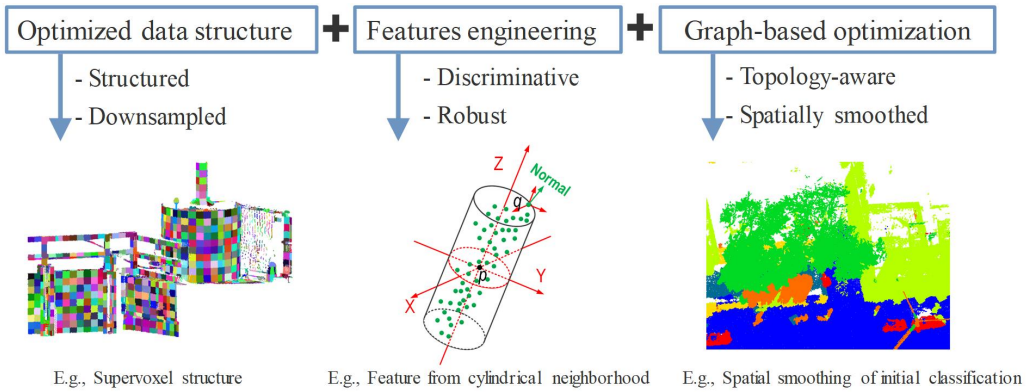


Figure 1.3: Solutions to questions.

To answer these questions, we proposed a solution (see Fig. 1.3) focuses on aspects of optimized data structure, the feature engineering, and graph-based optimization, which could help us addressing the task of reconstructing objects from construction sites and urban scenes.With the proposed solution, we developed a set of algorithms and methods for various applications. Fig. 1.4 gives an overview of algorithms and methods with involved publications, presenting solved tasks,

core strategies, employed data structures, and their relations. By combining these methods together, we can propose a detailed workflow for the reconstruction task in a certain scenario. It is noted that the work about registration is not included in this thesis but only use the voxel-based data structure we developed, so for keeping the completeness of the entire research, we still keep them in this diagram.

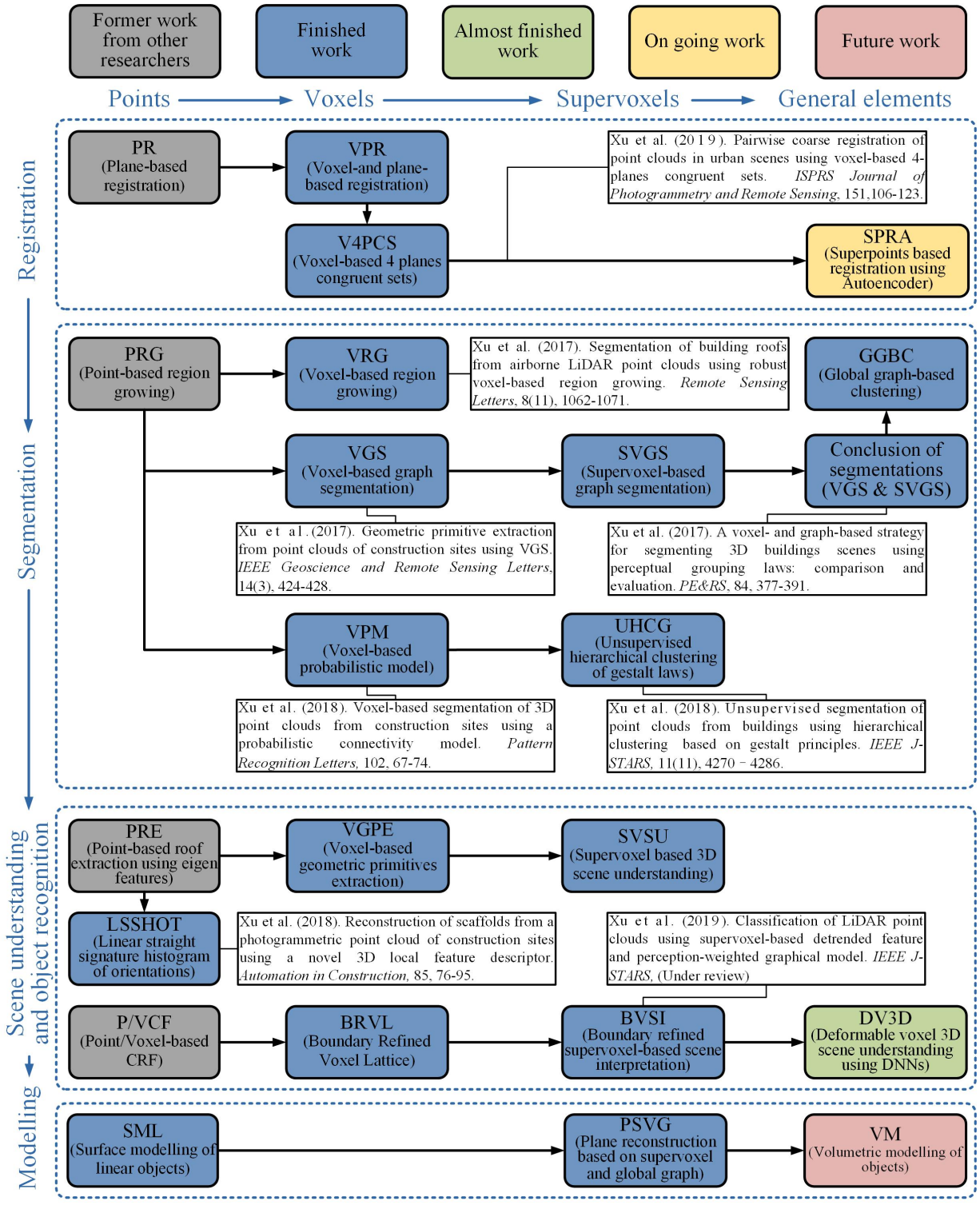


Figure 1.4: An diagram of algorithms and methods with involved publications of solved tasks, core strategies, employed data structures, and their relations.

1.4 Structure and organization

This thesis is structured as follows. Chapter 2 addresses the theoretical aspects of voxel structure, Graphical Models, and Sample consensus-based modeling. Chapters 3-6 is the core of this thesis and describes the employed methods and how the methodological steps are interrelated as a means for attaining the goals presented above. Chapter 7 presents the experiments, as well as the study site and the dataset used in this thesis. Chapter 8 presents the experimental results and discussions from all performed experiments. Chapter 9 closes the thesis by drawing the main conclusions and making suggestions for future works.

2 Theoretical basics

In this chapter, specific techniques and algorithms about the basic voxel-based data structure, geometric feature extraction, geometric modeling, and the optimization with the graphical model are described and introduced, and the relative concept and development of these techniques will be extensively used in the following methodology work. An appropriate notation is also introduced which will be used throughout the remaining chapters of this thesis.

2.1 Voxel-based data structure for 3D point clouds

2.1.1 Octree-based voxelization of 3D space

Voxels for the 3D space, like pixels in the image, is a basic rasterized unit of the three-dimensional space, representing a position on a regular cubic grid. Voxelization refers to point clouds, converting a point cloud into a voxel-grid, approximates the geometries of objects with a certain spatial resolution. In [Vo et al., 2015], the input point cloud is organized with an octree-structure, which partitions each piece of space into eight equal subspace. As a consequence, each sub-space namely voxel at a single level can be found along the octree. A general representation of the voxel-structure is shown in Fig. 2.1.

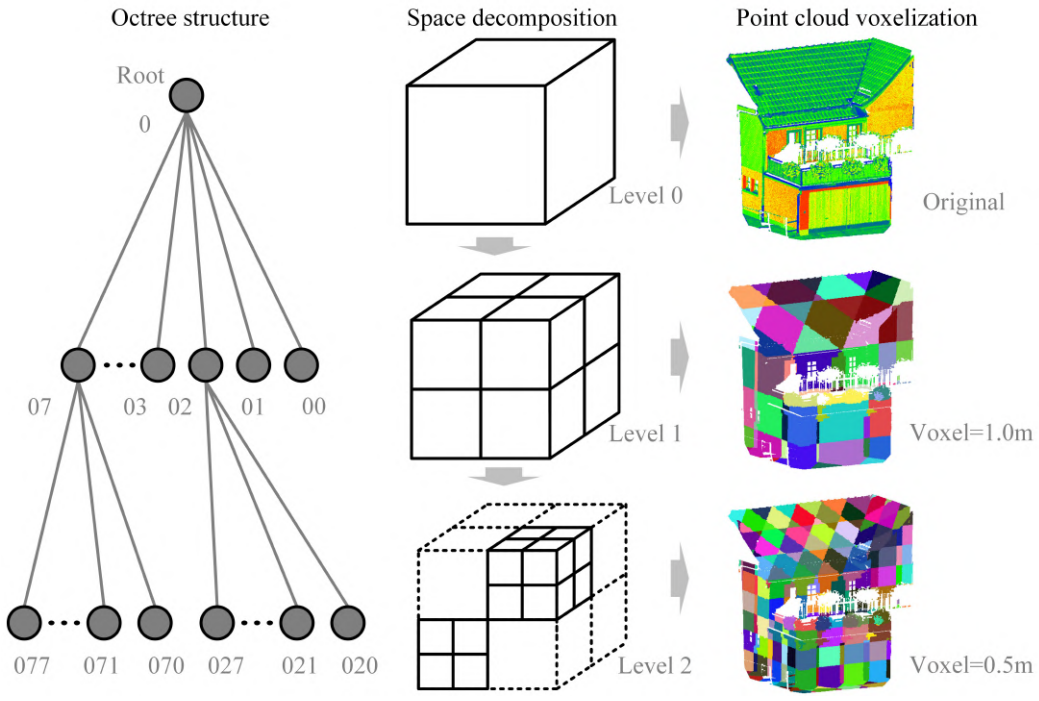


Figure 2.1: Representation of the octree-structure.

Compared with previous point-based data organization method, voxel structures result in the simplified representation of the elaborate scenes. It can be regarded as a down-sampling process, the computational cost is thus reduced. Meanwhile, the tree structure used to divide the space can accelerate the searching process, which can improve efficiency during the traversal as well. Furthermore, negative influences resulted from the inhomogeneous density of point clouds, which is very common in point cloud datasets, are eliminated, and the point clouds are reorganized with approximated planes. It is also worth mentioning that the noise and outliers are suppressed by points sharing the same voxel bounding. However, it is also a double-blades sword of using voxel-structures. It is necessary to have heuristic or empiric knowledge of the scene of objects during voxelization due to the variety of the proper setting for different scenes. On this background, a possible solution is the further development of the voxel-structure, namely the supervoxel-structure.

2.1.2 Voxel cloud connectivity segmentation-Supervoxelization

As mentioned before, empiric or heuristic knowledge is required when using voxel structures. Therefore, a developed data structure namely supervoxel structure is explored on the basis of voxels by [Lim & Suter, 2009]. In order to organize the entire point cloud into a supervoxel structure, space is firstly divided into a small 3D cubic grid employing octree partitioning, which splits each node into eight equal child nodes, in order to generate the octree-based voxel structure. For the further generation of supervoxel structures, the Voxel Cloud Connectivity Segmentation (VCCS) algorithm is adopted to cluster voxels concerning geometric as well as the spatial and spectral distance between the seed and candidate voxels [Papon et al., 2013]. In Fig. 2.2, we illustrate the generated supervoxels from the original point cloud using VCCS.

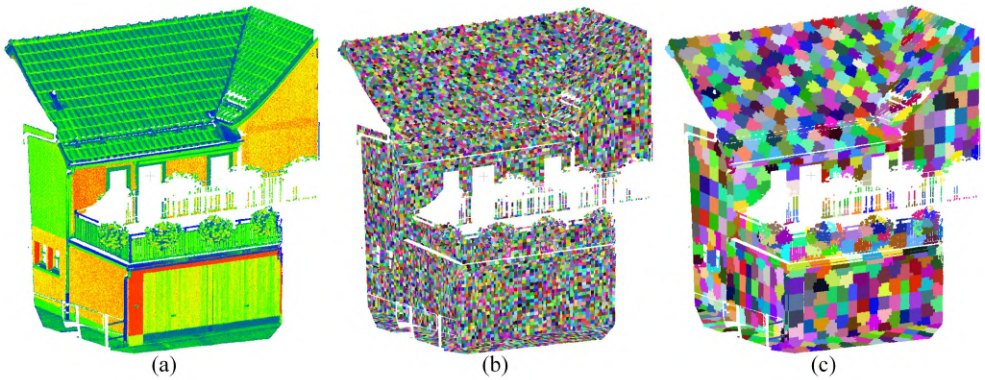


Figure 2.2: Sketch of the supervoxel-structure.(a) Original point cloud. (b) voxelized point cloud. (c) Generated supervoxels.

VCCS is one of the most recent supervoxel methods generating volumetric over-segmentation of a 3D point cloud. As stated in [Papon et al., 2013], Supervoxels of VCCS will adhere to object boundaries better than other state-of-the-art methods. In the meantime, VCCS can remain efficient, and even available to be used in online applications. To be specific, VCCS utilizes a combination of region growing strategy and the local k-means clustering. Here, supervoxels are evenly distributed across the entire 3D voxel space, which is achieved by setting the seeding voxels with a regular grid. Besides, the growing of voxels will be stopped at the boundaries of supervoxels only if the underlying voxels are judged as spatially connected ones after the local

k-means clustering. For judging the connections, a distance D is measured in the feature space defined by the spatial positions, RGB colors, and normal vectors.

$$D = \sqrt{w_c {D_c}^2 + w_s \frac{{D_s}^2}{{R_{seed}}^2} + w_n {D_n}^2}, \tag{2.1}$$

where D_c, D_s, and D_n are the distances in the corresponding Euclidean, color, and normals spaces, while w_c, w_s, and w_n are the weighting factors. However, it is note that in this work, the RGB colors are not used as the cues for generating supervoxels, thus in these cases, the weighting factor w_c is set to zero.

Here, the judgment of spatial connections is conducted between the adjacent leaves of the octree structure, which is an adjacency relation in 3D space with a maximum of 26 adjacents (i.e., neighboring) voxels.

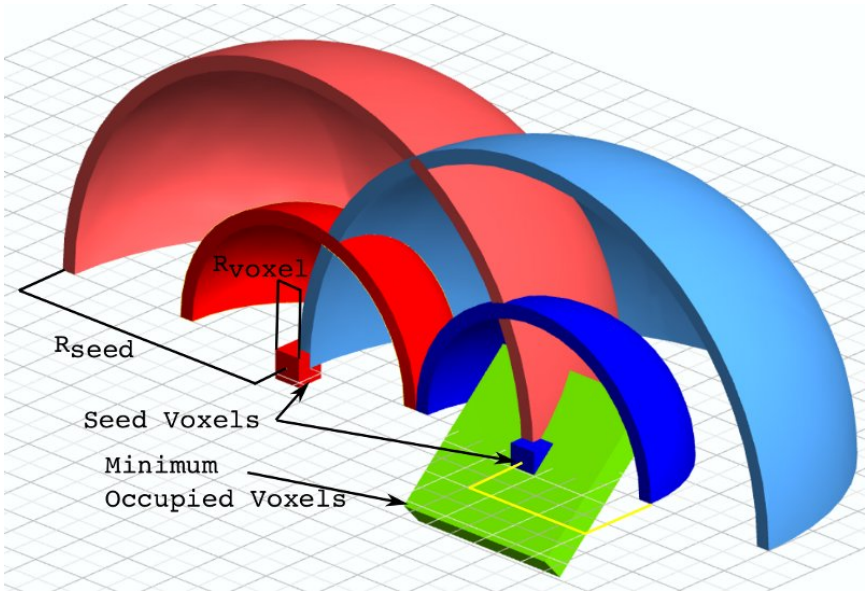


Figure 2.3: The various sizing parameters which affect supervoxel clustering. Figure courtesy of [Papon et al., 2013].

As a consequence of the generation of supervoxels, conforming spatial and geometric connectivities, objects boundaries are better captured. Besides, the operating efficiency using supervoxel structures is comparably higher than using voxel structures, due to the drastic reduction of computational elements. Finally, only normal vectors and spatial distance are considered during supervoxelization, which shows better performance at preserving real geometric boundaries of objects than that implemented at the voxel level.

2.2 3D shape descriptor for features extraction

As we have discussed in the literature review part, 3D shape descriptor is crucial to the supervised classification of point clouds and can be generally grouped into two categories: low-level description and high-level description. In this thesis, we have developed and utilized two types of 3D shape descriptors with both high-level and low-level description of the local geometry respectively, for the tasks of classification and segmentation. To be specific, we have developed a novel local 3D shape descriptor for the high-level description of the linear-shaped objects. Besides, we

applied the renown eigenvalue-based geometric features to the representation of local geometry for patches in segmentation.

2.2.1 SHOT: Signature of histograms of orientations

The SHOT descriptor is a typical high-level feature extraction algorithm, which represents the signature and histogram of local geometry explicitly. One of the major advantages of SHOT is that by intersecting between signature and histogram based feature extractors, SHOT would retain high robustness to noise and achieve strong descriptiveness benefiting from as an essential property of histograms and an inevitable result of introducing signatures simultaneously [Tombari et al., 2010].

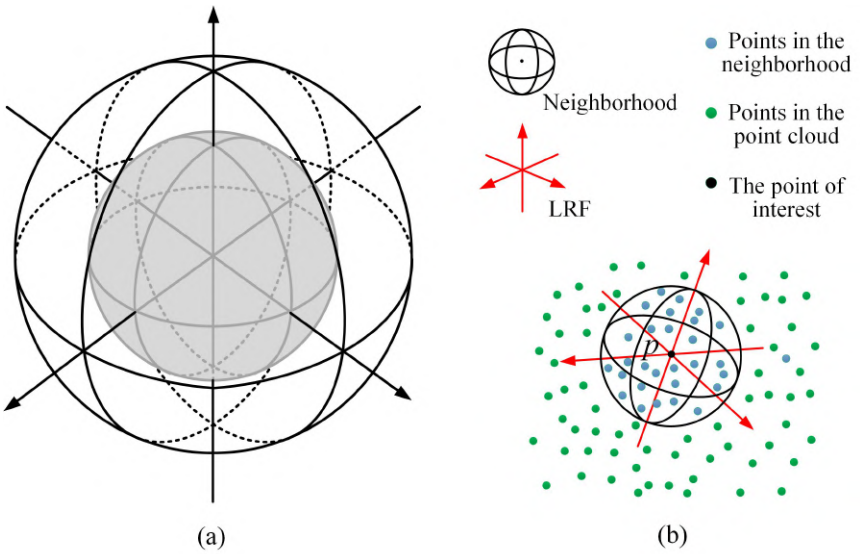


Figure 2.4: (a) Subdivision structure in the neighborhood of SHOT. (b)Neighborhood selection and local reference frame (LRF).

The implementation of SHOT starts from the selection of neighborhood $N(p)$ for the point of interest p. Here, the spherical neighborhood centered at the interest point p with radiusr_s is adopted (see 2.4b). Once the neighborhood is selected, the unique and unambiguous Local Reference Frame (LRF) is defined to the p, through the normal estimation method using total least squares and view-point reorientation. LRF identifies the spatial distribution of points in the neighborhood according to p so that it should be invariant to translations and rotations as well as robust to noise [Tombari et al., 2010].

For the spherical neighborhood with a defined LRF, an isotropic partition with 32 subdivisions is employed along the radial, azimuth and elevation axes (see 2.4a). Each subdivision would be described with a local histogram of 11 bins, and all 32 histograms form up the so-called signature of SHOT with a histogram with 352 bins [Tombari et al., 2010]. With the vector $\boldsymbol{Z}$ of the defined Z-axis in LRF and the normal vector $\boldsymbol{N}$ of each point located in a specific subdivision, the angular feature of each point could be calculated, which is actually a cosine function of the included angle θ_i between $\boldsymbol{Z}$ and $\boldsymbol{N}$:

$$\theta_i = \boldsymbol{Z} \cdot \boldsymbol{N} \tag{2.2}$$

The cosine value $\cos\theta_i$ of all the points in the neighborhood would be assigned and accumulated into bins of the local histogram. In this assignment and accumulation process, a quadri-linear interpolation is applied to avoid boundary effect, namely when accumulating a the angular feature

of a point into a specific bin of a local histogram, the neighboring bin in the same histogram, as well as the bins with the same index, would be added with a weight of $(1-d)$. As aforementioned, finally SHOT will generate a histogram of features with 352 bins. Here, a normalization step is applied on the histogram to guarantee the Euclidean norm of the histogram is equal to one, strengthening the robustness to the variations of point density [Tombari et al., 2010]. The normalized histogram is the output of SHOT.

Comparing to other classic 3D shape descriptors(e.g., FPFH), although the application of SHOT requires longer processing time as well as higher memory consumption, it still has several superiorities: (1) SHOT introduced the spatial distribution of the neighboring points by defining a unique and robust LRF, which would possibly enhance the discriminative power of the shape descriptor; (2) SHOT employed both histogram and signature into the descriptor structure, which makes the SHOT descriptor strongly robust to noise and at the same time highly descriptive; (3) the introducing of the cosine value as the angular outcome feature highlighted the most informative neighboring points and to some extent subdued the effect caused by noise. According to the investigation in [Tombari et al., 2010], SHOT can extract essential features of the underlying shape with the presence of noise and clutters, however, its robustness when coping with point clouds containing varying point densities and contaminated by outliers still need to be investigated.

2.2.2 Eigenvalue-based geometric features

Focusing on the low-level description of the local geometry, eigenvalue based features are introduced in [Chehata et al., 2009; Weinmann et al., 2015]. After choosing a suitable neighborhood $N(p)$, with an e.g., spherical [Lee & Schenk, 2002] or cylindrical [Filin & Pfeifer, 2005] shape, or using voxel-based neighborhood [Xu et al., 2018d], the 3D coordinates of points within a certain neighborhood $N(p)$ can be used to derive the 3D structure tensor M, namely, the covariance matrix of the x-, y-, z- coordinates describing 3D covariance matrix constructed from this neighborhood. The 3D structure tensor $\boldsymbol{M}$ can be calculated with the k points $\{p_1, p_2, ..., p_k\}$ within a neighborhood according to

$$\boldsymbol{M} = \frac{1}{k}\sum_{i=1}^{k}(p_i - \bar{p})(p_i - \bar{p})^T \tag{2.3}$$

where $\bar{p}$ is the geometric center of the used neighborhood, following

$$\bar{p} = \frac{1}{k}\sum_{i=1}^{k} p_i \tag{2.4}$$

The eigenvalues and eigenvectors of M provide important information about the geometry of points inside $N(p)$, which could be used to distinguish between the type of surface which a point p is situated on. For instance, the eigenvector associated with the smallest eigenvalue is the normal vector of the surface at p. As a result, the three eigenvalues existing in the 3D structure tensor are non-negative and present an orthogonal system of eigenvectors [Weinmann et al., 2015]. Thus, the respectively derived eigenvalues λ_i with $i \in \{1, 2, 3\}$ within a certain neighborhood can be used to explore and quantize local 3D shape. The eigenvalues satisfy $\lambda_1 > \lambda_2 > \lambda_3$ and are firstly normalized into $\{e_1, e_2, e_3\}$ by

$$e_i = \frac{\lambda_i}{\lambda_1 + \lambda_2 + \lambda_3}, i = 1, 2, 3 \tag{2.5}$$

With these three eigenvalues, linearity L_λ, planarity P_λ, scattering S_λ, which are dimensionality features, can be further obtained according to

$$L_\lambda = \frac{e_1 - e_2}{e_1} \tag{2.6}$$

$$P_\lambda = \frac{e_2 - e_3}{e_1} \tag{2.7}$$

$$S_\lambda = \frac{e_3}{e_1} \tag{2.8}$$

As described in [Weinmann et al., 2015], L_λ, planarity P_λ, scattering $S_\lambda \in [0, 1]$ represent respectively 1D, 2D, and 3D features with $sum(L_\lambda + P_\lambda + S_\lambda) = 1$. Thus, it is considered that, the dimensionality features react response the probabilities of a set of points to be 1D, 2D or, 3D structures. Futher local 3D shape features consisting omnivariance O_λ, anisotropy A_λ, eigenentropy E_λ, and local curvature C_λ are also eigenvalue-based handcrafted features, refering to

$$O_\lambda = \sqrt[3]{e_1 \cdot e_2 \cdot e_3} \tag{2.9}$$

$$A_\lambda = (e_1 - e_3)/e_1 \tag{2.10}$$

$$E_\lambda = -\sum_{i=1}^{3} e_i \ln(e_i) \tag{2.11}$$

$$C_\lambda = \frac{e_3}{e_1 + e_2 + e_3} \tag{2.12}$$

Consequently, the 3D shape of the points within a neighborhood can be measured using merely geometry. A view of a set of derived feature vectors can be found in Fig. 2.5. As seen from the figure, points located at two different parts of the scene having various local geometry, the feature vectors of which show a low similarity, can probably be well distinguished.

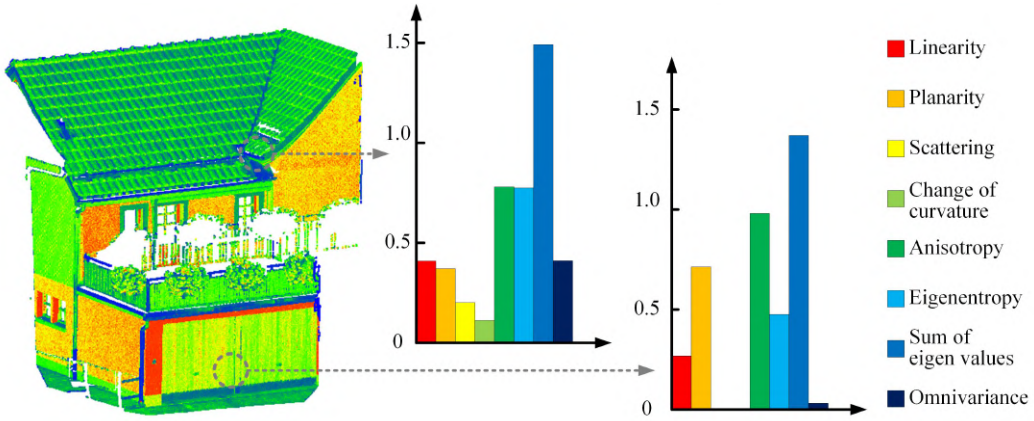


Figure 2.5: Feature vectors of local 3D shapes.

Although the linear, surface and spatial structure in the real scenes can be recognized with these local 3D shape features to some extent, there remain drawbacks. To be specific, there are usually objects of different sizes in the real complex scenes, the local 3D shape features of various kinds of objects with a fixed neighborhood are sometimes similar. Thus, the definition of the neighborhood is crucial. Besides, it is unavoidable, that feature vectors of various objects share similar local 3D shapes, which makes further solutions, for instance, a way of operating available feature vectors, more desirable.

2.3 Geometric modeling of objects

Geometric modeling is to generate simplified representations of the 3D shape of the labeled primitives (i.e., structural components of the building). In this thesis, we utilized the parametric modeling with sample consensus fitting and the surface modeling with boundary representation using cell decomposition.

2.3.1 Parametric modeling with Sample Consensus

Introduced by Fischler Bolles [1981], SAmple Consensus (SAC), including RANSAC and MLESAC, are approaches for fitting sets of observed 2D or 3D points with parameterized mathematical models in the presence of outliers. RANSAC and its extensions are the most popular SAC method that used for shape fitting. Given a set of input points $P = \{p_1, p_2, ..., p_n\}$ and the mathematical model $T(\theta)$ with a set of parameters $\theta \in \Theta$, SAC methods pursuit to estimate the parameter value $\theta^* \in \Theta$ which has the largest number of inliers in P. SAC is an iterative strategy, and in each round of iteration (e.g., the round i), the generic SAC method is executed with the following sequence of steps: Firstly, select a random sample $P_m \in P$ with n_m points, and n_m is the minimal number of points for estimating θ_i. For a 3D plane, at least three points is mandatory to define a plane in 3D space. Then, estimate the value of θ_i using only P_m. Afterward, evaluate the cost $C(\theta_i)$ of fitting entire dataset P to the estimated model $T(\theta_i)$:

$$C(\theta_i) = \sum_{p \in P} f_\rho(T(\theta), p) \tag{2.13}$$

where f_ρ is the function to measure the cost for each p. The iteration are repeated for a given number of times. The parameters θ^* make the model $T(\theta^*)$ earning the minimum cost will be chosen as the optimized parametes for the model, and normally it means the model with the largest number of inliers $P_{in} \in P$. The inliers P_{in} are identified by a give threshold ρ. To be specific, a point has an error e to the model $T(\theta)$, which is small than ρ, the function f_ρ returns zero value and otherwise a positive penalty T:

$$f_\rho(e) = \begin{cases} 0 & \text{if } e^2 < \rho^2 \\ T^2 & \text{else} \end{cases} \tag{2.14}$$

In contrast, MLESAC is an improvement of the original RANSAC, the cost function of which also considers the errors of inliers when fitting the model [Torr & Zisserman, 2000]. In other words, the size of the error e of each point will affect the entire cost $C(\theta_i)$ via the function f_ρ:

$$f_\rho(e) = \begin{cases} e^2 & \text{if } e^2 < \rho^2 \\ T^2 & \text{else} \end{cases} \tag{2.15}$$

For the inliers P_{in} selected by the model $T(\theta^*)$, they can be further used to refine the parameters of the model with weighted least square algorithm. The number of iteration for the SAC methods could be estimated via to the desired probability p_r of chosing an inlier from the data [Fischler & Bolles, 1981].

$$k = \frac{\log(1 - p_r)}{\log(1 - w^{n_m})} \tag{2.16}$$

Here, $p_r = 1 - (1 - w^{n_m})^k$ and $w = n_{in}/n_{all}$. n_{in} is the number of true inliers, while n_{all} denotes the number of all the points in the dataset.

2.3.2 Boundary representation with cell decomposition

Cell decomposition is a bottom-up algorithm to generate the boundary representation of an object, which decomposes a 2D contour into a small set of primitives with simple geometry (e.g., simple convex polygons), namely, cells. Then, these cells are aggregated into a boundary representation of the object with complex geometry. This method was firstly developed for the task of modeling building footprint [Kada, 2007], but now it is also frequently used for the indoor modeling [Oesau et al., 2014; Wang et al., 2017; Previtali et al., 2018]. Although the resulting partitioning only

approximates the original outline, it eases the task of determining and assembling a complex structure from parameterized, standard shapes [Kada, 2007].

This approach assumes that the majority of complex shapes have one main section or many extra connected sections so that a partition, therefore, can be acquired from polygons of the outline [Kada & McKinley, 2009]. Based on this derived partition, a general boundary description of the object can be constructed by assigning parameterized standard shapes to the set of sections. To achieve this, this method will create a set of non-overlapping and quadrilateral shaped polygons (i.e., cells) for each section of the object. By assembling these polygons, the original contours of the section can be approximated by their outlines. In Fig. 2.6, we illustrate the process of using this approach to get the boundary representation of a 2D object.

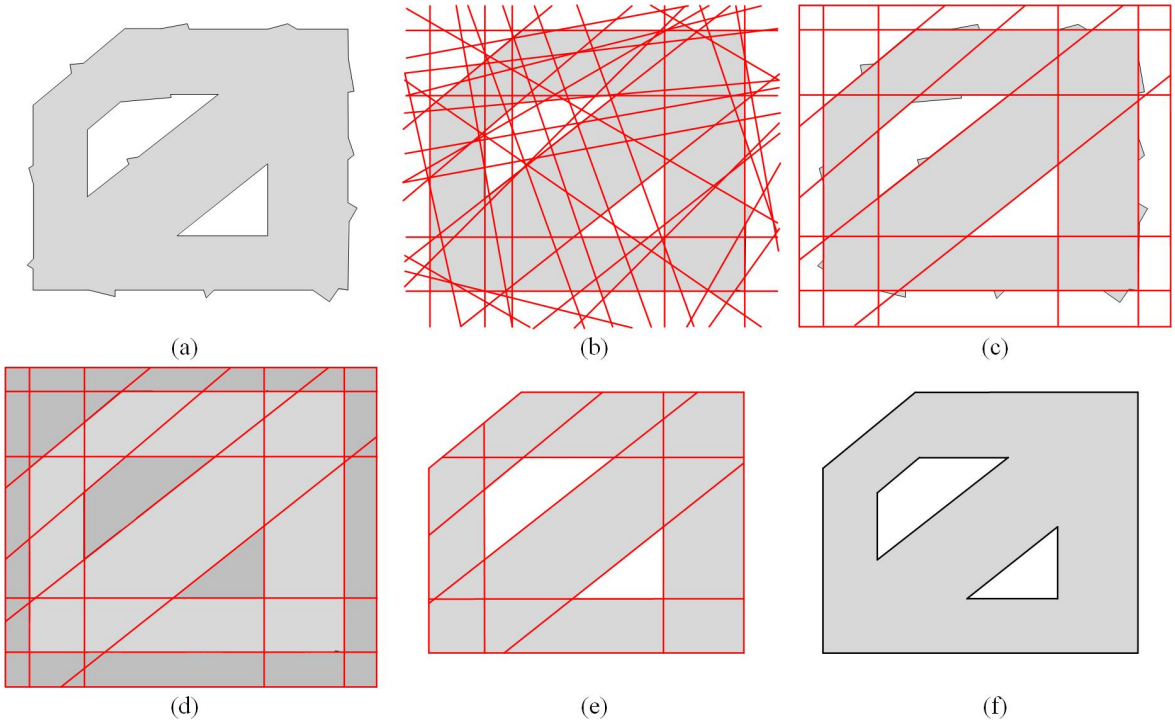


Figure 2.6: Cell decomposition for boundary representation. (a) Contours of the object. (b) Extraction of linear primitives. (c) Refinement of lines. (d) Generation of cells. (e) Selection of cells. (f) Boundary representation of the object.

As shown in the figure, in the cell decomposition process, the first step is to extract the contours of the given object, which is usually achieved by the alpha-shape algorithm [Wang et al., 2017]. Then, linear primitives are detected by the use of SAC methods we discussed in the last section, and these linear primitives should be clustered and merged with outliers removed. Afterward, a 2D arrangement algorithm will be applied to generate cells $C = \{c_1, c_2, ..., c_n\}$ from these intersecting lines. By occupied analysis or classification (e.g., Graph Cuts [Oesau et al., 2014]), the cells will be assigned to two groups, namely the cells belongs to the objects C^* and the cells of the background $C^{'} = C \backslash C^*$. At last, the boundary lines of cells in C^* will be selected. By using the ray-casting method, those outlines of these cells are extracted and connected to form the boundary representation of the object.

2.4 Construction and optimization of graphical models

Graph structure, which is a statistical context model, is commonly used for modeling the geospatial relationship between neighboring 2D/3D points [Landrieu et al., 2017]. Compared with other data structure (e.g., regular 3D girds for structured point clouds), graphical models like the adjacency graph, can encode not only the features of points in the local context but also the interactions between the point and its surrounding neighbors when constructing the weighted edges of the graph. The weights of edges encapsulate the dependency and affinity between connecting nodes. By the optimization (e.g., partition) of the graphical model of the point cloud, the segmentation or classification task can be easily solved [Golovinskiy & Funk, 2009; Landrieu & Simonovsky, 2018].

2.4.1 Graph Cuts

Graph Cuts is an advantageous and popular energy optimization algorithm used in computer vision for image segmentation and stereo vision. Such methods associate a segmentation problem with the Min-cut problem which is a bipartitioning of the graphical model [Boykov & Kolmogorov, 2004]. It is worth clarifying that Graph Cuts is an optimization algorithm based on energy function minimization, which is not equal to "graph cut" method such as Min-cut [Johnson et al., 1993], Normalized-Cut [Shi & Malik, 2000], Ratio-Cut [Hagen & Kahng, 1992]. Here, the later ones should be categorized as spectral clustering methods.

A normal graphical model consists of vertices and edges. To be specific, a graph $G = (V, E)$ is used to represent the data (e.g., image) to be segmented, and V and E are respectively a set of vertex and edge. If the edges have directions, such graphs are called directed graphs. Otherwise, they are undirected graphs. Edges are entitled to values, and the value of edges varies according to different weights representing different physical meanings. In the Graph Cuts algorithm, the graphical model is slightly different from the normal graphical model. The Graph Cuts graph has two more vertices based on the normal graph. These two vertices s and t are represented by the symbols "S" and "T" (see Fig. 2.7b), collectively referred to as terminal vertices. All other vertices must be joined and connected to the two vertices to build part of the edge set. Namely, there are two kinds of vertices and two types of edges defined in Graph Cuts.

Elements of the data structure build the first type of vertices and edges. The vertex corresponds to each pixel in the image, while the connection of every two adjacent vertices is an edge, the set of which is also called n-links. In contrast, the second type of vertices and edges is built between two terminal vertices, called the source point S meaning the source of the flow and the sink point T meaning the convergence of the flow, and vertices of the first type. In other words, we need to construct a connection between normal vertices of the data structure and the two terminal vertices to form a second side. This kind of edge is also called t-links.

In Figs. 2.7a-b, we provide an s-t diagram corresponding to the data structure of an image. Each pixel corresponds to a corresponding vertex in the graph, and two additional vertices are representing s and t. The edge of the solid black line represents the edge n-links of the normal vertex connection of each two neighborhoods, and the edge of the dotted line represents the edge t-links of each normal vertex connected with s and t. In the context of segmentation, s generally represents the foreground target, while t generally represents the background. A "cut" for the graphical model is a subset C of the edge set E, and the cost $|C|$ of the "cut" is the sum of the weights of all edges of the subset C. The disconnection of edges in the set will result in the separation of the graphical model, corresponding to two disjoint subsets "S" and "T", which is term as "cutting" (see Fig. 2.7c). If a "cut" has the smallest sum of the weights for all of its edges, it is regarded as the minimum cut. According to the Ford-Fulkerson theorem, the minimum cut

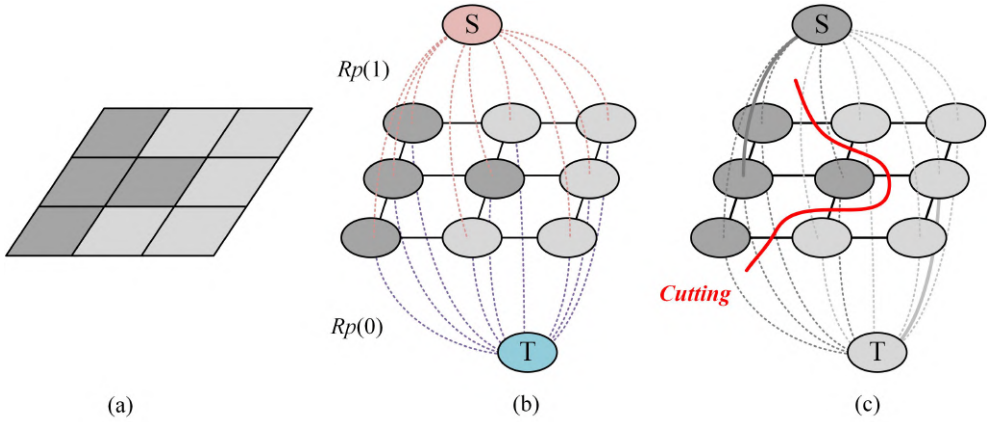


Figure 2.7: Illustration of Graph Cuts for segmentation. (a) Original data structure. (b) Constructed graph. (c) Cut of the graph.

problem of the graphical model is equal to the maximum flow problem of the network so it could be solved by the optimized solution using Goldberg-Tarjan algorithms [Boykov & Kolmogorov, 2004]. To be specific, we set the label of vertices of "S" to one, and the label of vertices of "T" to zero. Then, the "cut" of the graph can be achieved by minimizing the energy function:

$$E(L) = \lambda \cdot R(L) + B(L) \tag{2.17}$$

where $L = \{l_1, l_2, ..., l_p\}$ is the set of labels (0 or 1) given to all the vertices. Here, λ is the important factor determining the balance of influence between $R(L)$ and $B(L)$ on the energy. $R(L) = \sum R_p(l_p)$ is the regional term relating to the weight, and $R_p(l_p)$ represents the penalty for assigning the label l_p to the vertex p. While $B(L) = \sum B_{<p,q>} \cdot \delta(l_p, l_q)$ is the boundary term. $B_{<p,q>}$ can be resolved into a discontinuous penalty between vertices p and q according to following:

$$B_{<p,q>} = e^{-\frac{(I_p - I_q)^2}{2\delta^2}} \tag{2.18}$$

where I_p and I_q stand for the general values of vertices p and q. The term $\delta(l_p, l_q)$ is the key to the boundary term, and on this term there are three constraint:

$$\delta(l_\alpha, l_\beta) = \begin{cases} 0 & \text{if } l_\alpha = l_\beta \\ 1 & \text{else} \end{cases} \tag{2.19}$$

$$\delta(l_\alpha, l_\beta) = \delta(l_\beta, l_\alpha)\delta(l_\alpha, l_\beta) > 0 \tag{2.20}$$

$$\delta(l_\alpha, l_\beta) \leq \delta(l_\alpha, l_\gamma) + \delta(l_\gamma, l_\beta) \tag{2.21}$$

Here, l_γ is the label of a third vertex. The first two constraints tell that an energy between two different labels l_α and l_β should be non-zero. If it is zero, that means the two labels are the same. Generally, if p and q have similar properties, then $B_{<p,q>}$ will be larger, and if they are totally different, then $B_{<p,q>}$ is close to zero. While the last constraint defines the triangle rule, limiting that a shortcut of edges has always lower or similar energy than taking the whole path of edges. Only if the last constraint is satisfied, we can say the boundary term $B_{<p,q>}$ is metric.

The solution of the aforementioned energy function can be achieved by two algorithms: alpha-expansion and alpha-beta-swap [Boykov et al., 2001]. Here, the alpha-expansion algorithm can only be used when the boundary term is metric. Otherwise, the alpha-beta swap algorithm will be used. The basic principle of the alpha-expansion algorithm is to separate all l_α labeled and

non-l_α labeled nodes with "cutting" and the algorithm will change the label of l_α at each iteration. At each iteration, the region $\mathcal{R}_\alpha$ near the node with label l_α is expanded, with the graph weights reset. During the iteration, if two neighboring nodes do not share the same label, an intermediate node is inserted with weighted linking to the distance to the node with label l_α. The algorithm will iterate through each possible label for l_α until it converges. In contrast, the alpha-beta swap algorithm is to successively partition all nodes with label l_α from nodes with label l_β with "cutting" and the algorithm will change the label combination $l_\alpha - l_\beta$ at each iteration. During each iteration, the graph is constructed in a normal way which can segment between the region $\mathcal{R}_\alpha$ and the region $\mathcal{R}_\beta$ efficiently. In other words, for a node, the terminal link weight should be added with the sum of all links to neighbors which are neither within the region $\mathcal{R}_\alpha$ nor in the region $\mathcal{R}_\beta$. Similarly, the algorithm will iterate through each possible combination $l_\alpha - l_\beta$ until it converges.

2.4.2 Normalized Cuts and spectral clustering

The Normalized Cuts algorithm [Shi & Malik, 2000], also termed as N-cut, is a generic method for clustering or segmenting a set of arbitrary objects $P = \{p_1, ..., p_n\}$ organized over an undirected graphical model $G = (V, E)$, where the vertices V correspond to the set P of individual objects, while the edges E define the interaction between neighbors. The clustering or segmentation problem could be modeled as Normalized Cuts, and then solved by the relaxation of affinity matrix using spectral clustering [Von Luxburg, 2007].

As we mentioned before, Normalized Cuts is a kind of spectral clustering methods, utilizing the eigenvalues associated with the similarity matrix W quantifying the similarity between two objects, in order to represent a low-dimensional underlying structure of the input data via dimensionality reduction, so that the feature of which could be more distinctive when conducting the segmentation or classification. The similarity matrix W is constructed under the graph structure. To be specific, for the edge $e(p, q) \in E$ connecting vertices $p \in V$ and $q \in V$, the corresponding entry $W(p, q)$ at row p and column q of W is set to $f_s(I_p, I_q)$, and $W(p, q) = W(q, p)$. If vertices p and q are not connected in G, then we get $W(p, q)$ equal to zero. Here, $f_s(I_p, I_q)$ is the similarity function to estimate the non-negative weight of the edge $e(p, q)$, while I_p and I_q are the general properties of vertices p and q, respectively. For the commonly used Euclidean distance-based clustering, I_p and I_q stand for the two-dimensional coordinates of p and q, and the similarity function can be designed as follows:

$$f_s(I_p, I_q) = \begin{cases} e^{-\frac{||I_p - I_q||_2)^2}{\delta^2}} & \text{if } ||I_p - I_q||_2 < \gamma \\ 0 & \text{else} \end{cases} \tag{2.22}$$

where γ is a given threshold for measuring the minimum distance between two vertices in G.

Given the similarity matrix W, the Normalized Cuts cluster the input vertices into k disjoint subset $\{A_1, ..., A_k\}$ having $\cup_{i=1}^{k} = V$, by minimizing:

$$N_{cut}(A_1, ..., A_k) = \sum_{i=1}^{k} \frac{W(A_i, V \backslash A_i)}{\text{vol}(A_i)} \tag{2.23}$$

where $\text{vol}(A_n) = \sum_{v_i \in A_n} d_i$ where $A_n \subset V$. d_i is the degree of a vertex $v_i \in V$, which is defined as $d_i = \sum_{j=1}^{n} W(i, j)$. When the number of disjoint subsets is equal to two, maximizing the in-

tracluster association of objects and minimizing the inter-cluster disassociation can be obtained simultaneously by the minimization of the following:

$$N_{cut}(A,B) = \sum_{a\in A, b\in B} (\frac{W(a,b)}{\text{vol}(A)} + \frac{W(a,b)}{\text{vol}(B)}) \tag{2.24}$$

where $B = V\backslash A$. However, it is obvious that the above function is a NP-Hard problem, so that we can only find the approximate solution. A relaxed solution for this function is given in [Shi & Malik, 2000] by the spectral decomposition of the weighted adjacency matrix and corresponding Laplacian matrix. Let $\{u_1,...,u_k\}$ be the first k general normalized orthogonal eigenvectors of:

$$\boldsymbol{L}u = \lambda \boldsymbol{D}u \tag{2.25}$$

where $\boldsymbol{D} = diag(d_i), i = 1, ..., n$, and $\boldsymbol{L} = \boldsymbol{D} - W$. Here, considering the Rayleigh-Ritz theorem, $\boldsymbol{U} \in \mathbb{R}^{n\times k}$ is the relaxed solution having $\{u_1, ..., u_k\}$ as columns. Inside the matrix $\boldsymbol{U}$, we denote $y_i \in \mathbb{R}^k$ to the i-th row of U, so that by applying a simple k-means clustering based on y_i, we can get k clusters $\{C_1, ..., C_k\}$. The final solution of the disjoint subset A_i is given by $A_i = \{j|y_i \in C_i\}$.

2.4.3 Efficient graph-based segmentation

To provide a highly efficient and running in linear time (by the number of vertices) graph partitioning solution, an efficient graph-based segmentation method is introduced in [Felzenszwalb & Huttenlocher, 2004].

In the graph-based approach, the segmentation S is to partition vertices V into clusters $C \in S$, and each of them is equal to a connected sub-graph $G_0 = (V, E_0)$. Where $E_0 \subseteq E$, namely, any segmentation S is induced by a subset of the edges in E. By judging the similarity of vertices in a cluster of sub-graph, the quality of segmentation can be evaluated. To be specific, the criterion is that the vertices in a sub-graph should be similar, and vertices in different sub-graph should be dissimilar[Felzenszwalb & Huttenlocher, 2004]. To this end, when weighing the edges, the weights of the edges between two vertices in the same sub-graph should have relatively low, while those edges between vertices in different sub-graph could have higher weights.

Global thresholds are not appropriate for judging the similarity between two vertices, so it is natural to use adaptive thresholds. Here, for the sub-graph C, a Minimum Spanning Tree (MST) $T(C, E)$, having the smallest sum of edge weights, is generated with given vertices to be connected. For two sub-graph (after the generation of MST, they can also be called as two trees), two definitions are given. The first one is the internal difference Int of a sub-graph $C \subseteq V$, which should be the largest weight in $T(C, E)$ of the sub-graph:

$$Int(C) = \max_{e\in T(C,E)} W(e) \tag{2.26}$$

While the second definition is the difference Dif between two sub-graph $C_p, C_q \subseteq V$ should be the minimum weight edge connecting the two sub-graphs:

$$Dif(C_p, C_q) = \min_{e_i\in C_p, e_j\in C_q} W(e_i, e_j) \tag{2.27}$$

By checking whether the difference $Dif(C_p, C_q)$ between the sub-graphs is larger relative to one of the $Int(C_p)$ and $Int(C_q)$, we can predicate if there is evidence for a partition between these two sub-graphs. For controlling the degree to which the difference between components must be larger than minimum internal difference, a threshold funtion is given:

$$D(C_p, C_q) = \begin{cases} \text{true} & \text{if } Dif(C_p, C_q) > Mint(C_p, C_q) \\ \text{false} & \text{else} \end{cases} \tag{2.28}$$

Here, $Mint(C_p, C_q)$ defines the minimum internal difference in either C_p or C_q:

$$Mint(C_p, C_q) = \min(Int(C_p) + \tau(C_p), Int(C_q) + \tau(C_q)) \tag{2.29}$$

where $\tau(C) = k/|C|$ denotes the controlling threshold.

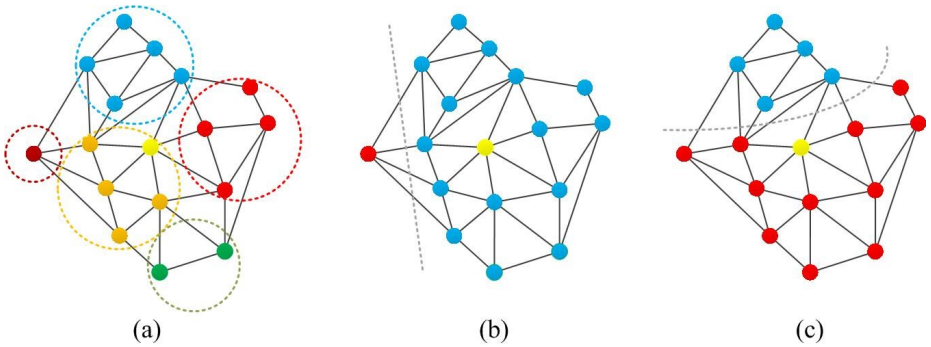


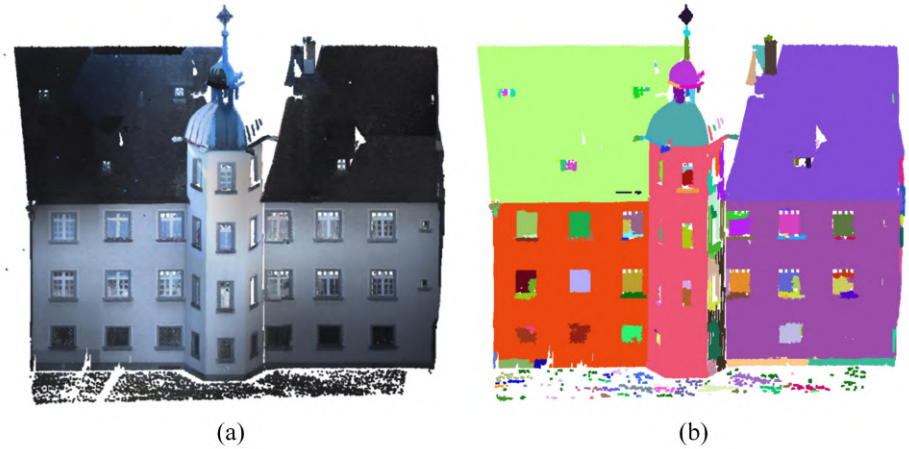
Figure 2.8: Illustration of the graph partitioning with difference algorithm. (a) Efficient graph-based segmentation. (b) Min-cut. (c) Normalized Cuts.

In the clustering process, initially, every vertex v_i is deemed to be one sub-graph C_i. The edges are sorted in ascending order by their weights. Afterward, the graph is partitioned via an iterative process. For vertices $v_i \in C_p$ and $v_j \in C_q$ of an edge $e(i, j)$, C_p and C_q will be merged as one sub-graph, if the minimum internal difference $Mint(C_p, C_q)$ is larger than $Dif(C_p, C_q)$. In the extreme case, if $|C_p| = 1$ and $|C_q| = 1$, then $Mint(C_p, C_q) = 0$,. The merging process is carried out repeatedly by traversing all the possible edges. Based on the output of the partitioned graph, connections of vertices are identified by the group of input entries.

Compared with other graph partitioning methods like Min-cut and Normalized Cuts, one of the advantages of the efficient graph-based segmentation is that it can more suitable for the multi-class clustering problem. In other words, we do not need prior knowledge of the number of classes k. It is true that there is a possibility to automatically identify k with the k-class Normalized Cuts by analyzing the difference of eigenvalues. However, they are robust enough when eigenvalues showing a gradual trend. In Fig. 2.8, we provide an illustration of partitioning a given graph via various "cut" algorithms. As seen from the figure, it is obvious that Normalized Cuts can provide a balanced partition comparing with Min-cut, but failed to conduct a multi-class clustering.

3 Recognition of structural elements in construction sites

In this chapter, we report an application of recognizing structural elements from 3D point clouds [Xu et al., 2017a,b, 2018a]. The methods used in this application involve voxel structure and graph-based clustering, providing a wholly automatic but parametric solution for partitioning and analyzing point clouds of 3D building scenes (see Fig. 3.1). Specifically, Voxel and Graph-based Segmentation (VGS) and SuperVoxel and Graph-based Segmentation (SVGS) using voxel and supervoxel structures are presented. To increase the efficiency and the robustness, the octree-based voxel structure is introduced, which can suppress the adverse effects of noise, outliers, and unevenly distributed point densities as well. The clustering of over-segmented voxels and supervoxels is achieved via graph theory by the local contextual information, which is commonly conducted merely with pairwise information in conventional clustering algorithms. The graph model is constructed according to perceptual grouping laws, considering geometric information associated with points. The adoption of perceptual grouping laws is for estimating geometric cues in order to identify the connections between basic elements (i.e., voxels or supervoxels), enabling a purely geometric and unsupervised solution for segmentation. Our methods conduct the segmentation in a purely geometric way avoiding the use of RGB color and intensity information so that it can be applied to general scenarios. Finally, a sample consensus fitting-based approach is proposed for the recognition of geometric primitives. In Fig. 3.2, we show the core methods corresponding to the ones we have shown in our research frame in Chapter 1.3.



(a) (b)

Figure 3.1: Segmenting the point cloud of 3D buildings. (a) Original unstructured point cloud (textured with RGB colors) and (b) Segmented result rendered with different RGB colors.

Figure 3.2: Methods for recognition of structural elements in construction sites.

3.1 A voxel- and local graph-based strategy for segmentation

The implementation of our proposed segmentation strategy consists of three core steps: the voxelization of the point cloud, the calculation of geometric cues, and the local graph-based clustering. In the first step, the entire point cloud is voxelized into a 3D grid. For the VGS method, voxels are

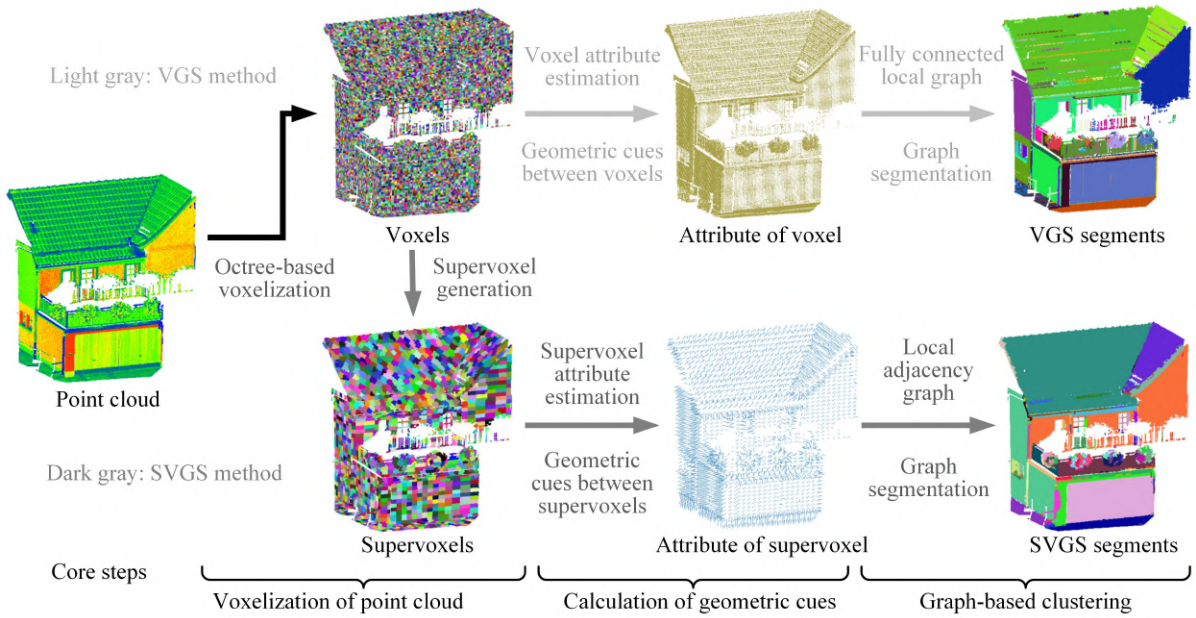


Figure 3.3: Comparison of workflows of two voxel- and graph-based segmentation methods.

basic elements for segmentation. In contrast, for the SVGS method, voxels are further clustered into supervoxels characterized by geometric and spatial consistency, which serve as segmentation primitives. In the second step, for estimating geometric cues of basic elements (i.e., voxels or supervoxels), attributes of each element are calculated according to geometric information of its constituent points. Geometric cues reflect the relation of two adjacent basic elements, namely they indicate whether two structural patches should be connected or not. Three representative principles of the perceptual grouping laws are selected as clustering criteria: proximity, similarity, and continuity. Perceptual grouping laws have a long history of use in the field of computer vision for recognizing objects in a scene, namely for the determination of regions of the visual scene belonging to the same part of higher level perceptual elements [Richtsfeld et al., 2014]. The proximity principle states that elements are likely to be categorized into the same group if they are close to each other, whereas the similarity principle claims that elements tend to be aggregated into a group when they resemble each other. The continuity principle indicates that oriented elements are considered to be integrated into one part in case that they can be aligned with each other. In the last step, based on the geometric cues, the homogeneity of basic elements is assessed and used for weighting edges in graphical models. Graph-based clustering is conducted to merge basic elements according to the edge weights in the graphical model in a greedy process. A separate graph is constructed for each basic element considering only adjacent

elements. This graph encodes the contextual information. By applying the graph segmentation algorithm, the connectivity of each element to its neighbors can be estimated. Then, all the connected elements can be aggregated into complete segments by a simple clustering. A comparison between processing workflows of VGS and SVGS methods is given in Fig. 3.1, with key steps and sample results illustrated. Detailed explanations on VGS and SVGS methods are provided in the following sections.

3.2 VGS: Voxel- and graph-based segmentation

The VGS method is the basic solution implementing the proposed strategy, adopting voxels as basic elements and fully connected local graphs for identifying connections of voxels.

3.2.1 Voxelization of point clouds

In this step, we use the octree-based voxelization to discretize the entire point cloud with 3D voxels. The advantages of using the octree-based voxel structure are as follows [Vo et al., 2015]: (1) It allows indexing the unorganized point cloud with octree structure, (2) it simplifies the dataset and suppresses the outliers and uneven density of point clouds with a grid-based representation, and (3) it defines neighborhood relations of the generated voxels as well as the points within them. It is noteworthy that selecting the size of voxels is a trade-off between the efficiency of processing and the preservation of details. Generally speaking, the smaller the voxels, the more details will be retained. In this work, the size of voxels is determined empirically according to the demands of the application. For example, in our experiments, for the reconstruction of major building elements (e.g., facades and roofs) the size of voxel was set to 0.2 m, but in other cases, it would depend on the requirements of the application.

3.2.2 Calculation of geometric cues

Geometric cues represent the relation between two voxels. The calculation of these cues includes two major steps: the estimation of voxel attributes and the application of perceptual grouping laws.

3.2.3 Estimation of voxel attributes

The attributes of a voxel V describe the geometric characteristics of points inside V, including three groups of features: spatial positions, geometric features, and normal vectors calculated from points. The spatial position stands for the spatial coordinate $\boldsymbol{X}$ of the centroid p of points within a voxel V. The geometric features are eigenvalue based features [Weinmann et al., 2015] related to the 3D distribution of points inside a voxel. In particular, we apply four local shape features, namely the linearity L_e, the planarity P_e, the scattering S_e, and the change of curvature C_e [Weinmann et al., 2015]. These four features are calculated from eigenvalues $e_1 \geq e_2 \geq e_3 \geq 0$ by eigenvalue decomposition (EVD) of the 3D structure tensor (i.e., covariance matrix), which is computed from 3D coordinates of all points inside the voxel.

As described in [Weinmann et al., 2015], L_e, P_e, and S_e relate to 1D, 2D, and 3D structures of points, respectively, while C_e captures the curvature of the surface formed by points. The normal vector $\boldsymbol{N}$ of points within V is obtained from the eigenvectors of the aforementioned tensor. Since noise and outliers are always a problem, the estimation of eigenvalues and eigenvectors is susceptible to errors in the coordinates of points. To alleviate this problem, we use the weighted

covariance matrix proposed in [Salti et al., 2014], assigning smaller weights to points who are distant from the centroid. The covariance matrix $\boldsymbol{M}$ is calculated as follows:

$$\boldsymbol{M} = \frac{1}{\sum\limits_{i:d_i \leq r} (r - d_i)} \sum_{i:d_i \leq r} (r - d_i)(p_i - p)(p_i - p)^T \tag{3.1}$$

where p_i denotes the point in the support region of size r for normal vector calculation. The size r of the support region equals to $\frac{\sqrt{3}d_v}{2}$, where d_v is the size of the voxel. p is the centroid of the points. Here, d_i stands for the distance of p_i from the centroid.

3.2.4 Binary features between voxels

To measure the proximity of the voxels V_i and V_j, the Euclidean distance $D^p_{ij} = ||\boldsymbol{X_i} - \boldsymbol{X_j}||$ between the centroids $\boldsymbol{X_i}$ and $\boldsymbol{X_j}$ of V_i and V_j is used. As far as the second perceptual grouping criterion, similarity, is concerned, it is assumed that the similarity of the spatial distributions of points inside a pair of voxels is reflected by the similarity of their geometric feature values. The similarity measure D^s_{ij} between V_i and V_j in the four dimensional space of the features defined earlier is calculated by the histogram intersection kernel [Papon et al., 2013]. The third perceptual grouping criterion, continuity, is evaluated based on the smoothness [Awrangjeb & Fraser, 2014] and the convexity [Stein et al., 2014] of the surface formed by the points inside adjacent voxels. Here, we make an assumption that the connection types between voxels are mainly as follows: smooth, "stair-like", convex, and concave. Sketches of these four types of connections are shown in Fig. 3.4. The smoothness D^m_{ij} is related to the difference of angles between normal vectors $\boldsymbol{N_i}$ and $\boldsymbol{N_j}$. The convexity D^o_{ij} depends on the local configuration of the surfaces formed by points of two adjacent voxels. A pair of surface patches is considered to be highly connective if the local configuration is convex. Whether the local configuration is defined as convex or concave is related to the relation of $\boldsymbol{N_i}$ and $\boldsymbol{N_j}$ and the vector $\boldsymbol{d_{ij}}$ joining the centroids $\boldsymbol{X_i}$ and $\boldsymbol{X_j}$, where $\boldsymbol{d_{ij}} = (\boldsymbol{X_j} - \boldsymbol{X_i})/||\boldsymbol{X_i} - \boldsymbol{X_j}||$. As illustrated in Fig. 3.4, angles α_i and α_j are calculated. Here, α represents the angle between the normal vector $\boldsymbol{N}$ and the vector $\boldsymbol{d_{ij}}$. If $\alpha_i - \alpha_j > \theta$, the surface connectivity is defined as a convex connection, where θ is the threshold for judging convexity. Otherwise, it is considered a concave connection. Here, θ is calculated by a sigmoid function determined by the difference of α_i and α_j, according to the description in [Stein et al., 2014]. The surface continuity D^c_{ij} is a combination of the smoothness D^m_{ij} and the convexity D^o_{ij}, which is calculated according to Eq. 3.2, signing a higher degree-of-proximity to gradual convex or smooth connected surfaces. For both convex/non-convex situations, the first term of Eq. 3.2 is related to the measure of smoothness $D^m_{ij} = (\alpha_i - \alpha_j)^2$, which is approximated by the difference between angles α_i and α_j instead of using the angle between normal vectors $\boldsymbol{N_i}$ and $\boldsymbol{N_j}$. The second term in Eq. 3.2 is associated with the measure of convexity D^o_{ij}, which is different in the two cases. In the convex case, the measure of convexity is defined as $D^o_{ij} = (\pi - \alpha_i - \alpha_j)^2$. In the concave case, the measure of convexity is defined by a constant penalty, namely $D^o_{ij} = \pi^2$. According to Eq. 3.2, a high degree of continuity (a low value of D^c_{ij}) is assigned to pairs of surfaces that are classified as being convex, "stair-like", or smooth (indicated by $\alpha_i - \alpha_j \leq \theta$), if the corresponding angular difference is small.

$$D^c_{ij} = \begin{cases} (\alpha_i - \alpha_j)^2 + (\pi - \alpha_i - \alpha_j)^2 & \text{if } \alpha_i - \alpha_j \leq \theta \\ (\alpha_i - \alpha_j)^2 + \pi^2 & \text{else} \end{cases} \tag{3.2}$$

For the case of surfaces with a smooth local configuration (see Fig. 3.4a), the angles α_i and α_j are both around $\frac{\pi}{2}$, the value of continuity is almost zero. In contrast, for the case of surfaces with a "stair-like" local configuration (see Fig. 3.4b), the surfaces are highly likely to be separated

parts of different objects and should be split. If the angles α_i and α_j are both around $\frac{\pi}{2}+\phi$, the value of continuity is around $4\phi^2$. Whether the "stair-like" surfaces should be disconnected or not depends on the value ϕ. For the cases of surfaces with a convex or concave local configuration, similar to the work reported in [Stein et al., 2014], for one object, it is assumed that the surfaces with a convex connection should always be preserved while the ones with a concave connection should be split according to the degree-of-convexity criterion. For instance, assuming that the angle $\alpha_i = \frac{\pi}{2}+\epsilon$ and the angle $\alpha_j = \epsilon$ in Fig. 3.4c, the value of continuity is around $\frac{\pi^2}{4}+(\frac{\pi}{2}-2\epsilon)^2$. In contrast, if the angles $\alpha_i = \epsilon$ and $\alpha_j = \frac{\pi}{2}+\epsilon$ in Fig. 3.4d, the value of continuity is around $\frac{5\pi^2}{4}$. It is clear that the continuity measure of the surfaces with a convex local configuration is much smaller than that of the surfaces with a concave local configuration, although the absolute values of the angle differences ϵ are the same.

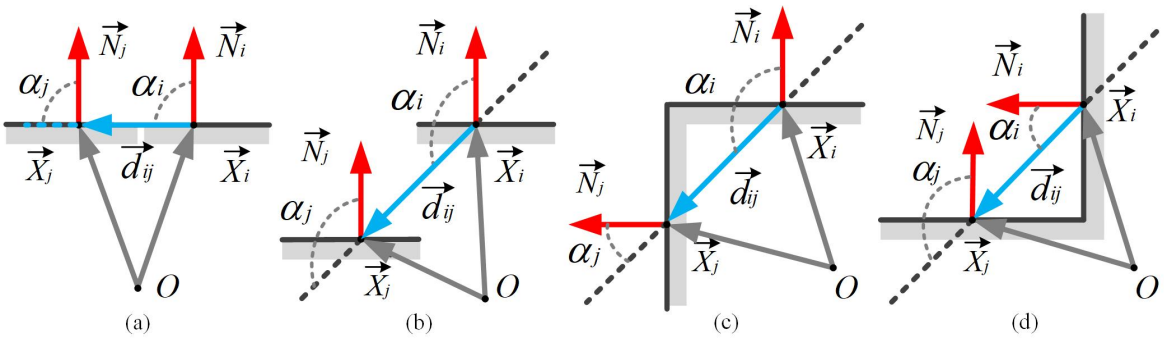


Figure 3.4: Connection types between two neighboring voxels. (a) Smooth, (b) "stair-like", (b) convex, and (c) concave connections.

3.2.5 Local graph-based clustering

In classical methods, the connectivity of voxels is identified by information extracted from merely two adjacent voxels, based on their similarity or on normal vectors [Wang & Tseng, 2011; Papon et al., 2013], but due to the complexity of 3D scenes, the assessment of connections considering only pairwise information seems unreliable. Therefore, we utilize the graph theory to estimate the connections of each voxel, considering information of all the adjacent voxels in a given neighborhood of a central voxel simultaneously. Here, a fully connected local graph $G = (V, E)$ is proposed as shown in Fig. 3.5.

3.2.6 Fully connected local graph of voxels

Graphical models have been widely used in many point cloud segmentation tasks. By using graphical models, the connectivity of two adjacent patches can be assessed in a context-aware way. The majority of approaches using graphical models operate at a global scale, i.e. they construct a graph for the entire scene and each point or element is related to a node (i.e., points or patches) of the graph [Golovinskiy & Funk, 2009; Pham et al., 2016b]. However, a large graph will significantly increase computational costs of the construction and partitioning steps. To avoid this problem, we define a local contextual graph for each voxel considering all the neighboring voxels in a local neighborhood. It means that the size of the constructed graph is limited to the neighborhood of the central voxel. The nodes in the graph correspond to the central voxel and its adjacent neighbors. In VGS, the local contextual graph is fully connected, which ensures optimal extraction of geometric information in the neighborhood. For the fully connected local graph, voxels are regarded as vertices V while the edges E link all the possible pairs of vertices.

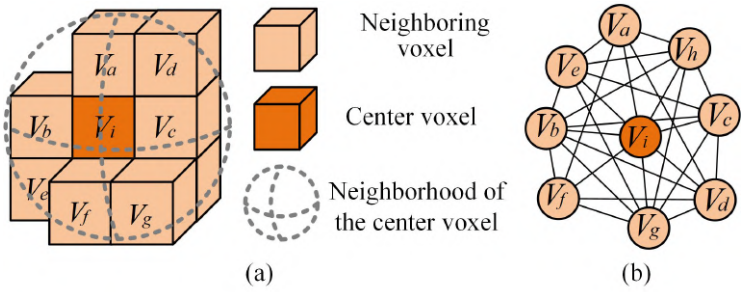


Figure 3.5: Constructing the local graph of a voxel. (a) Adjacent voxels in a neighborhood. (b) Fully connected local graph.

For each voxel, all adjacent voxels in its local graph that are connected to it according to the local graph segmentation procedure are considered to belong to the same segment. The weight $W(i,j) \in [0,1]$ between V_i and V_j is defined by joining all the $D_{ij}^k, k \in [p,s,c]$ between voxels by multiplication, because they are assumed to be independent:

$$W(i,j) = \prod_{k \in [p,s,c]} e^{(-\frac{(D_{ij}^k)^2}{2\lambda_k^2})} \tag{3.3}$$

where λ_p, λ_s, and λ_c are parameters controlling the importance of the spatial distance, the geometric similarity, and the surface continuity, respectively.

3.3 SVGS: Supervoxel- and graph-based segmentation

The SVGS method is an improved solution based on the supervoxel structure and the local affinity graph. There are three significant differences of the SVGS method compared with the VGS method. Firstly, the basic element for segmentation is different. In SVGS, supervoxels are applied for clustering segments instead of voxels. Secondly, the construction of the local graph is different. For the SVGS method, we use the local adjacency graph for each supervoxel rather than the fully connected graph used in VGS. Lastly, for clustering connected basic elements (i.e., supervoxels), the aggregation process is conducted through the merging of adjacency graphs. The purpose of using supervoxels is twofold: first, to improve the efficiency of the proposed strategy, because the use of supervoxels can largely reduce the number of basic elements used for segmentation; furthermore, the construction and segmentation of the local adjacency graph designed for supervoxels are simpler, so that theoretically it also requires less computational resources. The second aspect concerns the ability of supervoxels to preserve edges, as supervoxels have been proven to be quite effective when finding over-segmented boundaries of objects.

3.3.1 Generation of supervoxels

Supervoxels are generated using the VCCS method [Papon et al., 2013], which considers candidate voxels according to their distance to seed points within a feature space comprising centroid positions, normal vectors, geometrical features, and RGB colors. Somewhat different from the approach in [Papon et al., 2013], we calculate the distance using only normal vectors and spatial coordinates of voxels, which is related to the proximity and continuity laws of perceptual grouping. The variant of VCCS we used in SVGS is adopted from the Point Cloud Library (PCL) [Rusu & Cousins, 2011]. One of the most significant advantages of VCCS is its ability to preserve boundaries [Papon et al., 2013], through which we can obtain supervoxels the boundaries of which

coincide with the boundaries of major structures of objects in the scene. It is also notable that the size of voxels (i.e., the resolution of the voxel structure) and the resolution of seeds can greatly affect the performance of VCCS. To be specific, the former determines details preserved in the segments, whereas the latter influences the effectiveness of retaining boundaries. Empirically, we set these factors according to point densities and varying distances from the sensor to objects within the scene.

3.3.2 Local adjacency graph of supervoxels

Unlike fully connected local graphs used in VGS, to apply the graph model to the supervoxel structure, we define a local adjacency graph for each supervoxel encoding all its adjacent supervoxels in a local neighborhood. This is due to the fact that the generation of supervoxels already encapsulated the geometric information of voxels into the supervoxel, so that supervoxels have become independent units. Thus, there is no need to construct a fully connected graph for each supervoxel. Besides, the use of the local adjacency graph can also help to reduce the computational cost. Specifically, for each supervoxel V_i, all its n adjacent neighbors are counted as candidates for constructing the contextual graph $G_i = (V, E)$, which is represented in the form of nodes. Here, V and E represent the sets of all the supervoxels (i.e., nodes) and edges in the graph, respectively. A spherical neighborhood defined by radius R_c is defined as the local context of each supervoxel. Any supervoxel with its centroid located inside this spherical neighborhood will be regarded as a candidate of a direct neighbor of the center supervoxel. Then, the distance between the centroid of the candidate supervoxel and the centroid of the center supervoxel is measured. If this distance is smaller than $\sqrt{3}\iota$, this candidate supervoxel will be regarded as an adjacent supervoxel of the center supervoxel. Here, ι is the seed resolution defining the size of supervoxels when using VCCS. We use a kd-tree to conduct the nearest neighbor search. The weights of the edges E are determined using the same measures based on the laws of perceptual grouping as in the case of VGS. Here, the attributes of supervoxels are calculated using points inside all voxels of the supervoxel. The partition of the local adjacency graph G is also carried out in a similar fashion as for VGS, using graph-based segmentation [Felzenszwalb & Huttenlocher, 2004], by which the segmented graph G^* can be obtained. The supervoxel assigned to the central supervoxel according to G^* are considered to be connected to that supervoxel.

3.3.3 Aggregation of supervoxels

After the partition of local adjacency graphs, to aggregate supervoxels, all the segmented local adjacency graphs are traversed and checked. Fig. 3.6 shows an example, where the node V_k is shared by graphs G_i^* and G_j^*). Segmented graphs sharing such common nodes will be merged into a large graph G representing a segment. At the end of this merging process, each merged graph G will correspond to a segment. Similarly, during the identification of connections for each supervoxel, the cross validation used in VGS is also conducted. In Fig. 3.6, we give an illustration of how the local adjacency graphs are constructed, partitioned, and aggregated.

3.4 Efficient graph-based segmentation

Once the local graphs of all voxels are constructed, we can estimate the connections of each voxel by partitioning the constructed local graph. To this end, an efficient graph-based segmentation method is introduced by adapting the algorithm proposed in Felzenszwalb & Huttenlocher [2004]. Here, the segmentation C partitions vertices V (i.e., voxels) into segments $S \in C$ corresponding to connected components in the graph. Initially, every vertex V_i is deemed to be one segment S_i. The edges are sorted in ascending order by their weights. Afterwards, the graph is partitioned

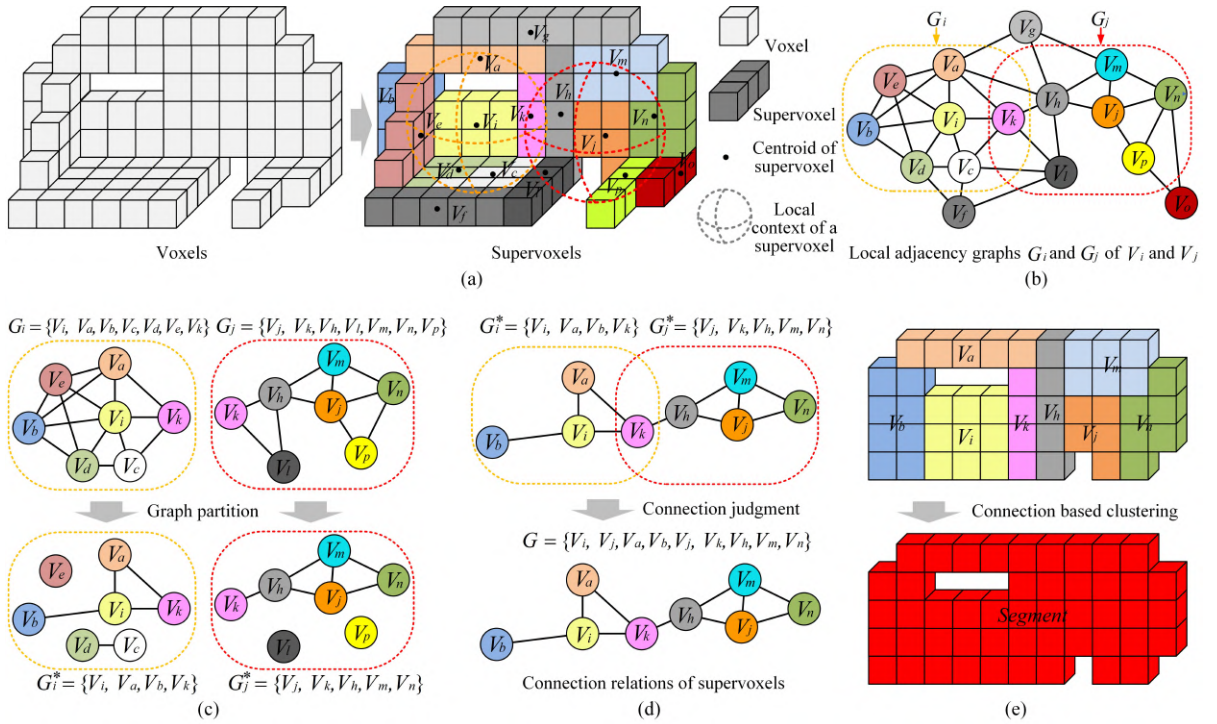


Figure 3.6: Aggregation of supervoxels. (a) Definition of local contextual area of supervoxel V. (b) Local contextual graphs of V_i and V_j. (c) Partition of local adjacency graphs. (d) Determination of connection relations of supervoxels. (e) Connection based clustering.

via an iterative process by comparing the maximum internal difference I_i inside a segment S_i and the external difference between segments S_i and S_j. Here, the maximum internal difference of a segment relates to the largest weight of the edges between vertices included in the segment. In contrast, the external difference is the minimum weight of the edges connecting two voxels of different segments in the graph. If the maximum internal difference of one segment is larger than the external difference between this segment and another segment, we will merge these two segments. After the merging of two segments, the maximum internal difference I of the new segment is updated, and it is increased by a term $\frac{\delta}{|S_{ij}|}$ associated with the number of voxels in the newly merged segment. Specifically, for vertices $V_i \in S_m$ and $V_j \in S_n$ of an edge E_{ij}, S_m and S_n will be merged, if E_{ij} has the smallest weight of all the possible edges connecting vertices of different segments S_m and S_n and the weight w_{ij} of E_{ij} is larger than the threshold τ_{mn}. Here, the weight w_{ij} is related to the external difference between S_m and S_n. The threshold τ_{mn} is estimated as follows:

$$\tau_{mn} = \max(I_m + \frac{\delta}{|S_m|}, I_n + \frac{\delta}{|S_n|}) \tag{3.4}$$

where $|S|$ stands for the number of voxels included in the segment S and δ denotes a constant parameter setting the initial threshold value. In the extreme case, if $|S_m| = 1$ and $|S_n| = 1$, then $\tau_{mn} = \delta$. The merging process is carried out repeatedly by traversing all the possible edges. In Algorithm 1, we provide a detailed description of the graph segmentation process. We assume the central voxel of a graph to belong to the same segment as all neighboring voxels that were connected to it in the graph segmentation procedure.

$G = (V, E)$, Local graph with vertices V and edges E $C = [S_1, S_2, ..., S_n]$: Segments of vertices
0: Sort E in ascending order according to its weight w
1: Initial segmentation $C^0 = [S_1, S_2, ..., S_m]$, $S_i = [V_i]$
2: Initial threshold $\tau_{ij} = \delta$, $I_i = 0$
3: for $\forall E_{ij} \in E$ do
4: If $w_{ij} > \tau_{ij} = \max(I_i + \frac{\delta}{|S_i|}, I_j + \frac{\delta}{|S_j|})$
5: $S_k \Leftarrow S_i \cup S_j$
6: $I_k = w_{ij} + \frac{\delta}{|S_k|}$
7: $C \Leftarrow \{C \backslash \{S_i \cup S_j\}\} \cup S_k$

Algorithmus 1 : Efficient segmentation of the local graph

3.5 Geometric primitive recognition

After the segmentation, the geometric shapes of segments are recognized, and then points of a certain geometric primitive are extracted and refined. Here, a method based on the efficient RANSAC [Schnabel et al., 2007] is adapted to recognize the geometric shapes of segments under the voxel structure. Points belonging to each geometric primitive are extracted via their parametric model.

3.5.1 Constrained candidate set sampling

Since the computational workload of RANSAC is directly linked to the success rate of selecting the good sampling sets [Schnabel et al., 2007]. Here, we use an improved sampling strategy for finding the appropriate sampling sets from the voxel structured point cloud efficiently, which is modified and tailored from the one described in [Schnabel et al., 2007]. We introduce two constraints to limit the sampling process. One is that the points within one voxel provide no more than one sampling point. Another one is that the points sampled should belong to the same branch of the octree structure. For each segment, the sampling of candidate points set is converted to the sampling of voxels sharing the same local octree branch. For the sake of the robustness, the sampled point $p_k \in P$ in a voxel V should be the point on a smooth surface having the smallest distance to the centroid, which is selected as follows:

$$p_k = \arg\min_{p_i \in P} ||p_i - p_0||_2 \tag{3.5}$$

where p_0 represents the centroid of all the points in this voxel.

This sampling strategy is backed by two facts of the natural shape. The first one is that for most points on a shape surface, there always exists a neighborhood located at the smooth area (not the edge, corner or outliers) containing only points belonging to this shape [Schnabel et al., 2007]. Following this, the second one is that finding the points in the smooth area on the basis of the relative residuals of points within one voxel can largely avoid sampling outliers. According to our sampling strategy, the possibility $\hat{P}(n)$ of finding an appropriate set Q for a shape of size n from the point cloud of size N in a single pass is optimized as $\hat{P}(n) = \frac{n}{N \cdot d}$, where d is the depth of the octree.

3.5.2 Identification of shape hypothesis

The aim of this step is to find the optimal geometric model from the points of a segment and estimate its parameters. For the lines and planes, the minimal candidate set includes only the

spatial coordinates $\boldsymbol{X}_1$, $\boldsymbol{X}_2$ and $\boldsymbol{X}_3$ of three points p_1, p_2, and p_3. For the cylinder, two points p_1 and p_2 with normal vectors $\boldsymbol{N}_1$ and $\boldsymbol{N}_2$ are used [Schnabel et al., 2007]. To identify geometric shapes of segment, a cost function summing the residuals of representing points according to the fitted model is calculated. The cost function will assign each model a score by counting the sum of residuals of all the representing points in a segment. The optimal shape of the given segment is identified by the geometric model having the smallest residuals. Once the optimal shape of the given segment is recognized, all the inlier points of this segment will be fitted to the geometric model of the shape with least squares approach and filtered by a threshold of distance from the point to the shape.

4 Segmentation of building objects in built environment

In this chapter, we present an application of segmenting building objects from 3D point clouds[Xu et al., 2016b, 2018d,e]. The overall goal intends to segment the surfaces of buildings into individual objects, referring to logical groups of points pertaining to specific structures, which can be easily used for the further model reconstruction work. In Fig. 4.1, we give an illustration of the segmentation of a scene of 3D buildings. Here, a bottom-up point cloud segmentation method that utilizes Gestalt principles in a hierarchical clustering framework (i.e., Unsupervised Hierarchical Clustering of Gestalt principles (UHCG)), allowing automatic processing for unsupervised segmentation of point clouds, which is developed based on the segmentation using Voxel-based Probabilistic Model (VPM) [Xu et al., 2018d]. For the voxel- and graph-based segmentation method, the weights of edges in the local graph need to be controlled by empirical weight factors. However, for the 3D structures with varying complexities, actually, the weights need to be set adaptively. To solve this problem, in the proposed method, weighted edges of the constructed local graph are estimated by the use of the probabilistic formulation, with the geometric cues estimated according to the Gestalt principles from the information of the local context. The segmentation method we reported is a significantly improved version based on our former work [Xu et al., 2016b]. The major improvements presented in this work include the simplification of the hierarchical clustering framework, the utilization of probabilistic formulation, the optimization of using graph-based segmentation, and the more extensive experiments and evaluations. In Fig. 4.2, we show the core methods corresponding to the ones we have shown in our research frame in Chapter 1.3.

4.1 Hierarchical clustering with Gestalt principles

The entire workflow of our segmentation method is depicted in Fig. 4.3, with involved methods and algorithms as well as illustrations of intermediate results shown.

Our hierarchical clustering framework is developed following the classificatory structure of [Sarkar & Boyer, 1993], which is designed for perceptual organization and consists of four different levels: signal level, primitive level, structural level, and assembly level. Firstly, in the signal level, points are organized by an octree-based structure partitioned into 3D voxels. Then, in the primitive level, all the voxels are clustered into over-segmented supervoxels, with saliency of boundaries found. In our method, supervoxels are the fundamental primitives used in our bottom-up segmentation process. In the structural level, attributes of supervoxels are calculated, in order to extract geometric cues according to Gestalt principles, so that connectivity between supervoxels can be identified. The use of probabilistic formulation estimates weighted edges of the constructed graph, and the efficient graph-based segmentation algorithm is adapted to partition the graph. Finally, at the assembly level, all the supervoxels are aggregated regarding their connectivities under a greedy merging frame, in order to generate complete segments.

4.2 Perceptual connected regions and geometric cues

Gestalt principles, also termed as perceptual grouping laws, have a long history of use in the field of computer vision, which refers to the process of determining regions and parts of the visual scene belonging to the same part of higher-level perceptual units (e.g., objects or patterns) [Wang et al., 2015b; Richtsfeld et al., 2014; Nan et al., 2011; Michaelsen et al., 2010].

The geometric cues reflect the relation of two adjacent patches (i.e., voxel or supervoxel), namely the cues indicating whether two structural patches should be connected or not. As we have discussed in the review part, some criteria like flatness, smoothness, or co-planarity have been widely used as cues for segmentation. However, the commonly used assumptions of smooth or planar surfaces for cues estimation may result in an "over-segmentation" of an individual object, namely partition the surface of the object into independent facets, requiring additional steps (e.g., merging of primitives) to group these facets into a complete surface. To simplify the segmentation process and give a general and unsupervised solution for object segmentation, we introduce Gestalt principles [Richtsfeld et al., 2014] for the estimation of segmentation cues. Gestalt principles delineate the geometric relationship and spatial dependence for assessing the connectivity between two adjacent patches so that we can assign the connections of nodes in the graphical model later [Wang et al., 2015b]. To be specific, four representative Gestalt principles are adopted, namely proximity, similarity, continuation, and closure. The proximity confirms that elements are highly likely to be aggregated into the same group when they are spatially close to each other. The similarity states that elements tend to be summed into one group when they resemble each other. The continuation reveals that oriented elements are disposed to be integrated into one part if they can be aligned with each other. The closure refers to the tendency of mind that prefer to see complete figures or forms even if the original data is incomplete or partially hidden by other objects.

In Fig. 4.4, we illustrate how the Gestalt principles are applied to avoid the over-segmented results when identifying the potential boundaries during the segmentation of the building point cloud. In this illustration, the segmentation of the window and the dome of the building are chosen as examples. When segmenting the window from a given facade, there are six potential boundaries indexed with numbers 1-6 in the figure, which are roughly generated according to the smoothness of the surface. In our method, this step is accomplished via the generation of

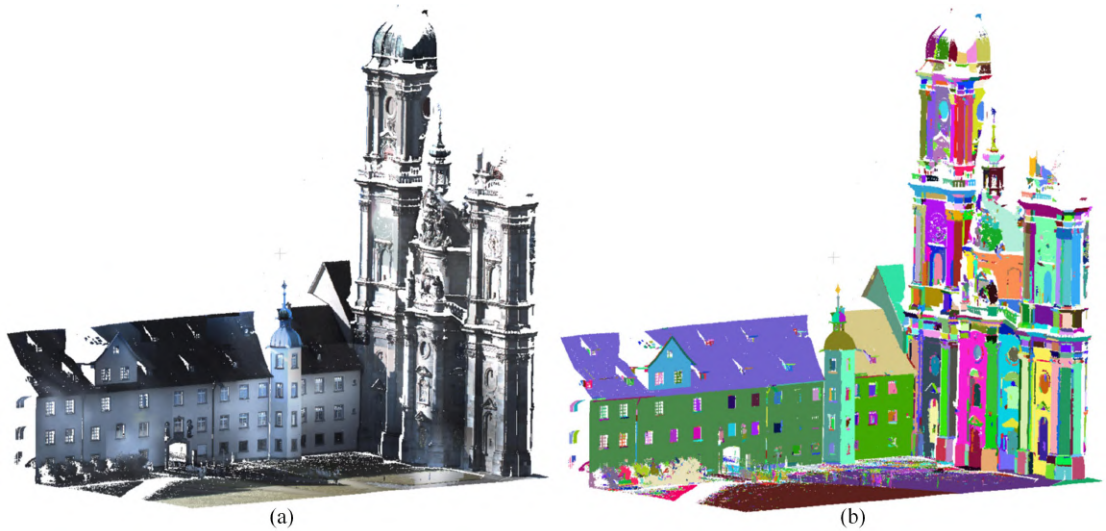


Figure 4.1: Segmenting the unstructured point cloud of 3D buildings. (a) Original point cloud textured with RGB colors and (b) Segmented result.

Figure 4.2: Methods for segmentation of buildingobjects in built environment.

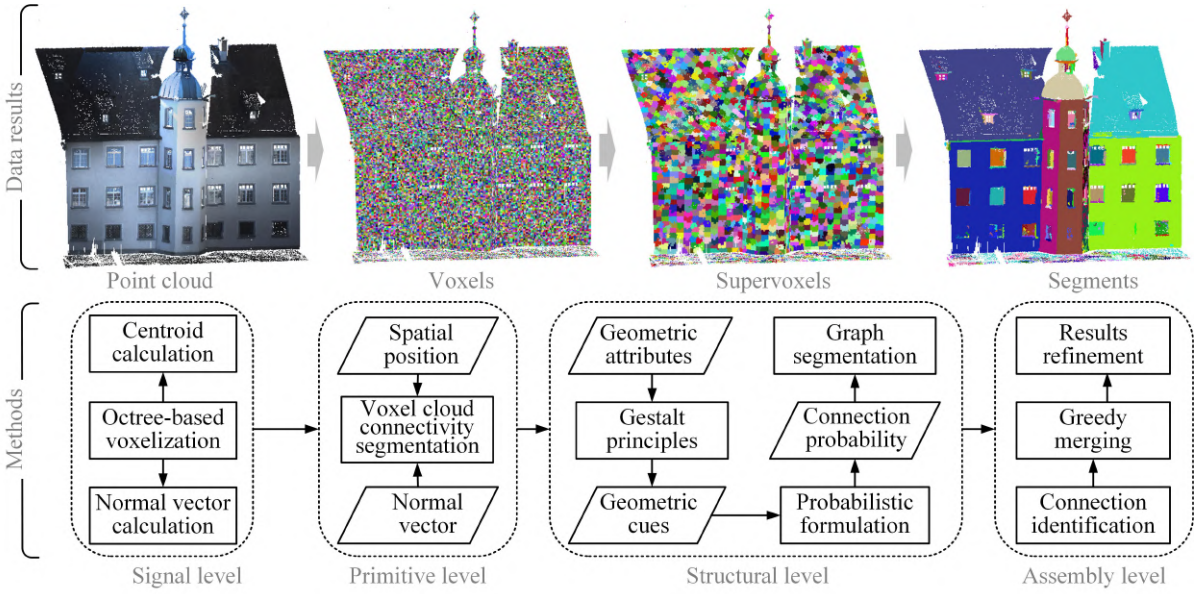


Figure 4.3: Overview of the proposed hierarchical clustering method.

supervoxels. It means that the edges between those supervoxels having no smooth connections will be selected as potential boundaries for segments. Here, the smoothness between supervoxels is estimated by the angle difference of their normal vectors.

For boundaries 1-4, they are the edges between the window frame and the wall surface, having different geometric shapes and no smooth connections, which do not satisfy the requirement of similarity and continuation from Gestalt principles. Thus, these boundaries should be kept in the segments. Similarly, for boundary 6, it is corresponding to a concave corner which is counter to the closure principle, so that it should be kept as well. In contrast, for boundary 5, it satisfies the proximity and similarity principles in light of the symmetric and similar shapes of window sashes, so that this boundary should be canceled and the patches on either side of this boundary should be combined. Similarly, when segmenting the dome of the building, there are five potential boundaries indexed with numbers 7-11. For boundaries 7 and 8, corresponding to concave connections, they should be kept due to the closure rules. While for boundary 9, it is a convex connection between two curved surfaces, which follows the principles of proximity, similarity, and closure, so that it should be canceled in the final segments. As for boundaries 10 and 11, although the surfaces between them meet the principles of proximity and closure, these surfaces have different geometries (i.e., planar or curved surfaces) and not smoothly connected. Thus, they should be kept. For the entire scene, if we can find structural patches with clear potential boundary information, with the help of Gestalt principles, we can quickly identify those boundaries should be kept in the final segments.

In our segmentation method, the implementation of Gestalt principles includes two major steps: calculating the attribute of structural patches and building geometric cues between patches. The Gestalt principles are modeled by the geometric cues built between patches with their attributes, which will be detailedly explained in the following sections.

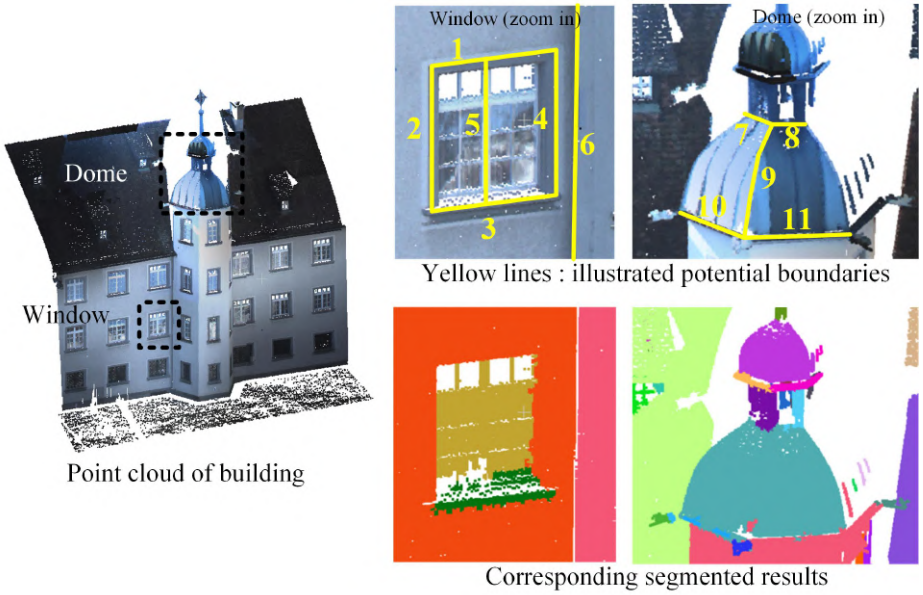


Figure 4.4: Judgment of potential boundaries using Gestalt principles for building segmentation.

4.2.1 Geometric attributes of patches

The attribute A of each structural patch V includes three aspects: spatial position, orientation, and geometric features, which are the unary feature abstracted from the points within it. To obtain the attribute, we firstly adopt the assumption of implicit plane representation [Dutta et al., 2014] (see Eq. 4.1) to represent the structural patch, defining an approximate plane via the normal vector $\boldsymbol{N_i}$ and centroid $\boldsymbol{X_i}$ of the point sets P_i within the patch V_i.

$$< \boldsymbol{N_i}, \boldsymbol{X_i} > -X_i^{\mathrm{c}} = 0 \tag{4.1}$$

where X_i^{c} stands for the distance from the origin to the approximate plane.

The spatial position stands for spatial coordinates of the centroid $\boldsymbol{X} = (x, y, z)$ for the points set $P = \{p_1, p_2, ..., p_n\}$ inside the patch. The orientation represents the normal vector $\boldsymbol{N} = (n_x, n_y, n_z)$ of the approximated surface formed by P. While geometric features refer to the eigenvalue-based covariance features [Weinmann et al., 2015] encapsulating sum, omnivariance, eigenentropy, anisotropy, linearity, planarity, surface variation, and sphericity, which are calculated by the eigenvalue $e_1 \geq e_2 \geq e_3 \geq 0$, via the eigenvalue decomposition (EVD) of the 3D structure tensor, namely the covariance matrix $\boldsymbol{M} \in R^{3\times3}$ of points coordinates of P. Here, a histogram $H = \{h_1, h_2, ..., h_8\}$ with eight bins is used to encode eigenvalue-based features. This histogram is the representation of geometric characteristics of the given patch. The calculation the attribute of the patch is listed in Table 4.1.

4.2.2 Geometric cues of connections

Once the attribute of patches is calculated, the geometric cues mimicking the relation between patches are built the resort to them by using Gestalt principles. Geometric cues can also be regarded as the binary feature between two patches.

The geometric cue of proximity C_{ij}^{p} between V_i and V_j is measured by the spatial distance ΔX_{ij}:

$$\Delta X_{ij} = ||\boldsymbol{X_i} - \boldsymbol{X_j}||_2 \tag{4.2}$$

Table 4.1: Features of a structural patch.

Spatial information	Centroid	(x, y, z)
	Normal vector	(n_x, n_y, n_z)
Covariance features	Sum	$e_1 + e_2 + e_3$
	Omnivariance	$\sqrt[3]{e_1 \cdot e_2 \cdot e_3}$
	Eigenentropy	$-\sum_{i=1}^{3} e_i \ln(e_i)$
	Anisotropy	$(e_1 - e_3)/e_1$
	Linearity	$(e_1 - e_2)/e_1$
	Planarity	$(e_2 - e_3)/e_1$
	Surface variation	$e_3/(e_1+e_2+e_3)$
	Sphericity	e_3/e_1

For measuring the geometric cue of similarity C_{ij}^{s} between V_i and V_j, the difference ΔH_{ij} between H_i and H_j is used:

$$\Delta H_{ij} = \sum_{k=1}^{8} \left(\frac{h_i(k) - h_j(k)}{h_i(k) + h_j(k)}\right)^2 \tag{4.3}$$

Here, the smaller the ΔH_{ij}, the more similar the geometric shapes of two patches are.

In Fig. 4.5, we illustrate five typical connections between approximated planes of adjacent patches. To measure the geometric cue of continuation C_{ij}^{c}, we consider the smoothness C_{ij}^{n} and co-planarity C_{ij}^{d} of the approximated planes of two patches. The smoothness here means that two connected patches in a segment should locally make a smooth surface, whose normals only slightly vary from each other [Rabbani et al., 2006]. Whereas the co-planarity indicates that two connected patches should not form a "stair-like" connection (see Fig. 4.5d). In other words, the distance between these two patches in the direction of the normal vector should be as small as possible. The C_{ij}^{n} and C_{ij}^{d} are quantified by two contributions: the angle ΔA_{ij} and the offset ΔD_{ij} between these planes, respectively:

$$\Delta A_{ij} = \angle(\boldsymbol{N_i}, \boldsymbol{N_i}) \tag{4.4}$$

$$\Delta D_{ij} = (X_i^{\mathrm{c}} - X_{ij}^{\mathrm{s}})^2 + (X_j^{\mathrm{c}} - X_{ji}^{\mathrm{s}})^2 \tag{4.5}$$

where X^{c} is calculated by Eq. 4.1 and $X_{ij}^{\mathrm{s}} =< \boldsymbol{N_i}, \boldsymbol{X_j} >$ [Dutta et al., 2014].

For evaluating the geometric cue of closure C_{ij}^{o}, the convexity criterion, standing for the 3D concave/convex relationship connecting surfaces, is used [Stein et al., 2014]. According to [Stein et al., 2014], the concave / convex relationship between two patches V_i and V_j is inferred from the relation of $\boldsymbol{N_i}$ and $\boldsymbol{N_i}$ as well as the vector $\boldsymbol{d_{ij}}$ joining the $\boldsymbol{X_i}$ and $\boldsymbol{X_j}$. As shown in Figs. 4.5e and 4.5f, the angles $\alpha_i = \angle(\boldsymbol{N_i}, \boldsymbol{d_{ij}})$ and $\alpha_j = \angle(\boldsymbol{N_j}, \boldsymbol{d_{ij}})$ are calculated. If $\alpha_i \geq \alpha_j$, the approximate surface between two patches is defined as a convex connection. In contrast, if $\alpha_i < \alpha_j$, it is defined as a concave connection. Here, we define the measure of closure ΔC_{ij} as follows:

$$\Delta C_{ij} = \begin{cases} (\alpha_i - \alpha_j)^2 & \text{if} \quad \alpha_i - \alpha_j \geq \theta \\ \pi^2 & \text{else} \end{cases} \tag{4.6}$$

where θ is estimated following the description in [Papon et al., 2013] by using a sigmoid function and the difference angle of normal vectors.

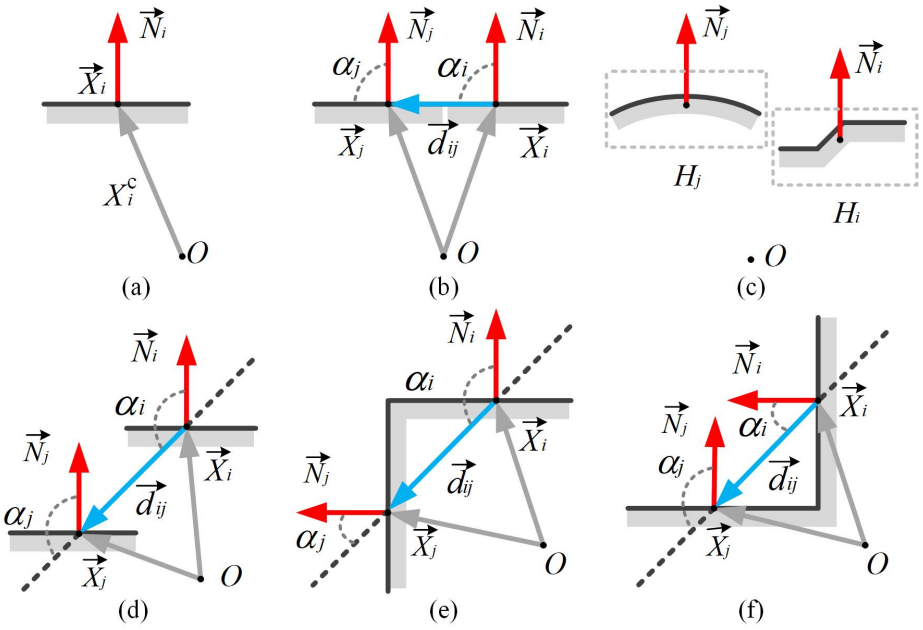


Figure 4.5: Local configurations between patches. (a) Plane representation of the patch. (b) Smooth connection. (c) Connection between patches of different shapes. (d) "Stair-like" connection. (e) Convex connection. (f) Concave connection.

4.3 UHCG: Unsupervised hierarchical clustering using Gestalt principles

The framework of our UHCG method consists of four different levels of data processing: signal level, primitive level, structural level, and assembly level.

4.3.1 Signal level: octree-based voxelization

The octree-based voxelization rasterizes the entire space covering the point cloud into 3D cubic boxes under an octree-based subdivision framework, which split each node in a tree structure into eight child nodes concurrently. Currently, it becomes increasingly popular in the field of point cloud processing [Wang & Tseng, 2011; Vo et al., 2015; Su et al., 2016]. The aim of utilizing the octree-based voxel structure is threefold [Vo et al., 2015]: (1) Organizing points (i.e., indexing the unorganized point cloud); (2) Constructing the rasterized representation of points (i.e., simplifying the dataset and suppressing outliers and noise); (3) Defining neighboring relations of generated voxels as well as the points within them. The entire point cloud is rasterized with a 3D cubic grid, and the points located in the same cubic is clustered as a voxel. The relations between voxels are directly obtained from the octree. In this work, the granularity of the voxel structure is determined according to the demands of the application, namely the subdivision of the octree is stopped according to the divided size of voxels. In each voxel, it should remain at least three points for estimating eigenvectors. It is noteworthy that selecting an appropriate size of the voxel is a trade-off between the efficiency of computations and the preservation of details. The larger the voxel, the more details will be smoothed.

With respect to the octree structure we used, its voxelization process is a top-down process subdividing the bounding box of the entire point cloud into eight children (i.e., the subdivision of space with 3D cubics) recursively, which will generate the voxel structure and create the neighboring relations of voxels simultaneously. This is because, during the subdivision of voxels in

each level, the octree has already encoded the relations of all the nodes in the tree structure. The points are organized by the cubic space of voxels as well. Thus, compared with other nearest neighboring searching (NN) methods mainly designed for point-based data structures like approximate nearest neighbor (ANN) [Arya et al., 1998], by using octree structure, we do not need to firstly create a 3D grid structure to organize and represent all the points, and then identify the neighboring relations by applying the NN based methods, which adds an additional step in the entire workflow.

4.3.2 Primitive level: generation of supervoxels

After the voxelization, we represent the voxelized point cloud by a collection of voxels in order to reduce the computational cost [Pham et al., 2016a]. In our method, the small patch is represented in the form of supervoxel consisting of a set of voxels having geometric consistency according to the normal vector and the centroid of the points within the voxel. To achieve the supervoxels, we still adopted the VCCS method [Papon et al., 2013]. Similar to the work in the previous chapter, we merely use the normal vectors and spatial coordinates of voxels to define the distance, which is related to the geometric cues of proximity C^{p} and continuation C^{c}. The VCCS is implemented and tailored from the PCL [Aldoma et al., 2012]. It is noteworthy that by the use of the supervoxel structure, smaller voxels will be firstly clustered into supervoxels finding the correct boundaries of objects in the scene so that the segmentation results will be not too sensitive to the selection of voxel structures having different granularity.

4.3.3 Structural level: connectivity between supervoxels

After the first two levels, the connectivity of each supervoxel is estimated according to those geometric cues we described in the last Section. In this vital step, a probabilistic framework, which is proposed in [Alpert et al., 2012] and once successfully applied in [Xu et al., 2018d], is formulated to clustering the supervoxels according to their probability of connection. Unlike the probabilistic frame used in [Xu et al., 2018d], in this work, we also encode the similarity cues in the probabilistic formulation and apply the local affinity graph to determine the probability of connecting two supervoxels. Thus, this probabilistic framework involves two crucial steps: the probabilistic formulation, the construction, and partition of the affinity graph.

Formulation of probabilistic model

For aggregating V_i and V_j, we need a measure to assess whether or not they should be merged into a single cluster based on their attributes A_i and A_j. To achieve this goal, the probability $P(S_{ij}^{\pm}|A_i, A_j)$ determining if two supervoxels should be merged is deduced by their attributes as well as those of their surroundings. Here, $S_{ij}^{\pm}$ is defined as a binary random variable assuming that the values S_{ij}^{+} if V_i and V_j should be merged and S_{ij}^{-} if they are not. To evaluate the posterior probability, considering the cues are assumed to be independent, we apply the joint multiplicative model:

$$\begin{aligned} P(S_{ij}^{+}|A_i, A_j) &= \prod_k P(S_{ij}^{+}, C_{ij}^{\mathrm{k}}|A_i, A_j) \\ &= \prod_k P(S_{ij}^{+}|A_i, A_j, C_{ij}^{\mathrm{k}}) \prod_k P(C_{ij}^{\mathrm{k}}|A_i, A_j) \end{aligned} \tag{4.7}$$

Then, Eq. 4.7 can be transformed to Eq. 4.8 with the help of Bayes' theorem, which provides the probability of connecting supervoxels when using each cue. It is noteworthy that according to

the model in Eq. 4.7, the overall probability is estimated by each cue individually, which allows a dynamic adjustment of the influence of using different cues [Alpert et al., 2012].

$$P(S_{ij}^{+}|A_i, A_j, C_{ij}^{\mathrm{k}}) = \frac{L_{ij}^{\mathrm{k}+} P(S_{ij}^{+}|C_{ij}^{\mathrm{k}})}{L_{ij}^{\mathrm{k}+} P(S_{ij}^{+}|C_{ij}^{\mathrm{k}}) + L_{ij}^{\mathrm{k}-} P(S_{ij}^{-}|C_{ij}^{\mathrm{k}})} = \frac{P(A_i, A_j|S_{ij}^{+}, C_{ij}^{\mathrm{k}}) P(S_{ij}^{+}|C_{ij}^{\mathrm{k}})}{P(A_i, A_j|S_{ij}^{+}, C_{ij}^{\mathrm{k}}) P(S_{ij}^{+}|C_{ij}^{\mathrm{k}}) + P(A_i, A_j|S_{ij}^{-}, C_{ij}^{\mathrm{k}}) P(S_{ij}^{-}|C_{ij}^{\mathrm{k}})} \tag{4.8}$$

Here, $L_{ij}^{\mathrm{k}\pm} = L(S_{ij}^{\pm}, C_{ij}^{\mathrm{k}}|A_i, A_j)$ is associated to the likelihood density for $S_{ij}^{\pm}$, calculated via the attributes of supervoxels in the local neighborhood of V_i and V_j. Specifically, the geometric cues we used to estimate the likelihood density involve merely similarity C_{ij}^{s}, smoothness C_{ij}^{n}, and co-planarity C_{ij}^{d} cues. In contrast, the cues used to infer the prior are proximity C_{ij}^{p} and closure C_{ij}^{o} cues. Here, $P(S_{ij}^{\pm}|C_{ij}^{\mathrm{k}})$ stands for the prior probability, assumed to be independent from cues for estimating likelihood density.

Calculation of likelihood density

For calculating likelihood densities, we assume that the likelihood densities are deduced by attributes of surrounding supervoxels of a central supervoxel. Thus, for the likelihood density $L_{ij}^{\mathrm{s}\pm}$ of the similarity cue C_{ij}^{s}, we assume that the value of each bin h_i in H_i is distributed normally, i.e. $h_i(k) \sim \mathcal{N}(u_k, \sigma_k)$. Here, k stands for the number of bins. Therefore, the likelihood density L_{ij}^{s} can be modeled by the χ_k^2 distribution of a degree of freedom of k, with variables ΔH_{ij}. Then, $L_{ij}^{\mathrm{s}\pm}$ is approximated by $L(S_{ij}^{\pm}|\Delta H_{ij})$:

$$L_{ij}^{\mathrm{s}\pm} \approx L(S_{ij}^{\pm}|\Delta H_{ij}) = \chi^2(\alpha_{ij}^{\pm}) \tag{4.9}$$

where $\alpha_{ij}^{+} = |\frac{(k-2)\Delta H_{ij}}{min(\Delta H_i^{+}, \Delta H_j^{+})}|$ and $\alpha_{ij}^{-} = |\frac{(k-2)\Delta H_{ij}}{\frac{1}{2}(\Delta H_i^{-} + \Delta H_j^{-})}|$. Here, $\Delta H_i^{+} = min_n(\Delta H_{in}^{+})$ is the lower bound of ΔH_{in}, while $\Delta H_i^{-} = \frac{1}{n}\sum \Delta H_{in}^{-}$ denotes the average value, calculated according to n adjacent supervoxels.

While for the likelihood density $L_{ij}^{\mathrm{n}\pm}$ of the smoothness cue C_{ij}^{n}, the angles of normal vectors of surfaces are assumed to be independent random variables, which follow Gaussian distributions having same expectations and variances. Thus, we can model the difference of normal angles ΔN_{ij} with a zero mean Gaussian distribution. Then, $L_{ij}^{\mathrm{n}\pm}$ is approximated by $L(S_{ij}^{\pm}|\Delta N_{ij})$:

$$L_{ij}^{\mathrm{n}\pm} \approx L(S_{ij}^{\pm}|\Delta N_{ij}) = \mathcal{N}(0, \beta_{ij}^{\pm}) \tag{4.10}$$

where the standard deviation $\beta_{ij}^{\pm}$ is calculated by using all the difference angles of supervoxel normals in the neighborhood and defined by $\beta_{ij}^{+} = min(\Delta N_i^{+}, \Delta N_j^{+})$ and $\beta_{ij}^{-} = \frac{1}{2}(\Delta N_i^{-} + \Delta N_j^{-})$. $\Delta N_i^{+} = min_n|\Delta N_{in}|$ represents the minimal external difference in the neighborhood, while $\Delta N_i^{-} = \frac{1}{n}\sum \Delta N_{in}^{-}$ denotes the average value of all difference normal angles between adjacent supervoxels.

Similar to $L_{ij}^{\mathrm{s}\pm}$, for the likelihood density $L_{ij}^{\mathrm{d}\pm}$ of the continuation cue C_{ij}^{d}, we model it following the χ_k^2 distribution of a degree of freedom of 2, having variables ΔD_{ij}, as we assume that the term $\Delta X_i^d - \Delta X_{ij}^s$ in Eq. 4.2 follows a zero mean Gaussian distribution. Then, $L_{ij}^{\mathrm{d}\pm}$ is approximated by $L(S_{ij}^{\pm}|\Delta D_{ij})$:

$$L_{ij}^{\mathrm{d}\pm} \approx L(S_{ij}^{\pm}|\Delta D_{ij}) = \chi^2(\gamma_{ij}^{\pm}) \tag{4.11}$$

Here, $\gamma_{ij}^{+} = |\frac{\Delta D_{ij}}{min(\Delta D_i^{+}, \Delta D_j^{+})} - 1|$ and $\gamma_{ij}^{-} = |\frac{2\Delta D_{ij}}{\Delta D_i^{-} + \Delta D_j^{-}} - 1|$. $\Delta D_i^{+} = min_n(\Delta D_{in}^{+})$ relates to the lower bound of ΔD_{in}, while $\Delta D_i^{-} = \frac{1}{n}\sum \Delta D_{in}^{-}$ is the average value calculated via n adjacent supervoxels.

Estimation of the prior

As we stated in the former section, the proximity C_{ij}^{p} and closure C_{ij}^{o} cues are used to estimate the prior $P(S_{ij}^{\pm})$. Here, the prior is defined using these geometric cues:

$$\begin{aligned} P(S_{ij}^{+}) &= e^{-\frac{\Delta {P_{ij}}^2 + \Delta {V_{ij}}^2}{\lambda}} \\ P(S_{ij}^{-}) &= 1 - P(S_{ij}^{+}) \end{aligned} \tag{4.12}$$

where λ is the factor of Gaussian function. Here, the assumption is that two supervoxels are highly likely to be merged into one cluster, on condition that they locate close enough (i.e., proximity cue) and the local configuration of their surfaces meets the convex criterion (i.e., closure cue).

Arbitration of cues

$P(C_{ij}^{\mathrm{k}}|A_i, A_j)$ is calculated for relating the influence of the smoothness cue C_{ij}^{n} and similarity cue C_{ij}^{s} and corresponding attributes. Here, the Gaussian function is applied to smooth $P(C_{ij}^{\mathrm{n}}|A_i, A_j)$ and $P(C_{ij}^{\mathrm{s}}|A_i, A_j)$:

$$P(C_{ij}^{\mathrm{n}}|A_i, A_j) = e^{-\frac{\Delta {N_{ij}}^2}{\lambda_n}} \tag{4.13}$$

$$P(C_{ij}^{\mathrm{s}}|A_i, A_j) = e^{-\frac{\Delta {H_{ij}}^2}{\lambda_s}} \tag{4.14}$$

Whereas for the $P(C_{ij}^{\mathrm{d}}|A_i, A_j)$, since all the local configurations with "stair-like" surface should be disconnected, $P(C_{ij}^{\mathrm{d}}|A_i, A_j)$ should be more sensitive to ΔD_{ij} than the case of $P(C_{ij}^{\mathrm{n}}|A_i, A_j)$ to ΔN_{ij}. Therefore, the Sigmoid function having steeper changing slopes of the curve near the origin is used:

$$P(C_{ij}^{\mathrm{d}}|A_i, A_j) = \frac{2}{1 + e^{\frac{\Delta D_{ij}}{\lambda_d}}} \tag{4.15}$$

where λ_n, λ_s, and λ_d are scale factors.

Construction and partition of local affinity graph

The local affinity graph G is built upon the probability of connections, acting as a crucial role in the aggregation of supervoxels. In the neighborhood of each supervoxel, its adjacency defined by a radius R_a are selected, in order to construct a local affinity graph for this supervoxel. R_a determines the number of adjacent supervoxels that included in one local affinity graph. In this affinity graph, vertices represent all supervoxel within the neighborhood, while the weight w_{ij} of each edge is given by the estimated probability $P(S_{ij}^{\pm}|A_i, A_j)$.

We can achieve connecting relations of supervoxels by the partition of the constructed local affinity graph of each supervoxel. For this purpose, the graph-based segmentation method proposed in [Felzenszwalb & Huttenlocher, 2004] is adapted and used.

Here, the segmentation $\mathcal{C}$ is to cluster supervoxels V (i.e., the vertices in the graph) into segments $S \in \mathcal{C}$ equating with the connected components in the local affinity graph. As the initial step, every vertex V_i is regarded as one segment S_i. The edges are sorted in ascending order according to their weights. Then, the graph is partitioned via a recurrent process by comparing the weight w of an edge (i.e., the probability of the connection). The smallest weight inside a segment S_i is regarded as the maximum internal difference I_i of S_i. For vertices $V_i \in S_a$ and $V_j \in S_b$ of an edge $\mathcal{E}_{ij}$, if the weight w_{ij} of the edge between S_a and S_b is smaller than

$G = (V, \mathcal{E})$, Graph with vertices V and edges $\mathcal{E}$ $\mathcal{C} = [S_1, S_2, ..., S_n]$: Segments of vertices
0: Sort $\mathcal{E}$ in ascending order by its weight w
1: Initialization $\mathcal{C}^0 = [S_1, S_2, ..., S_m]$, $S_i = [V_i]$
2: Setting threshold $\epsilon_{ij} = \delta$, setting initial $I_i = 0$
3: Do {until $\forall \mathcal{E}_{ij} \in \mathcal{E}$ is traversed}
4: If $w_{ij} < \epsilon_{ij} = \min(I_i + \frac{\delta}{|S_i|}, I_j + \frac{\delta}{|S_j|})$
5: $S_k \Leftarrow S_i \cup S_j$
6: $I_k = w_{ij} + \frac{\delta}{|S_k|}$
7: $\mathcal{E}_{ij}$ is traversed
8: $\mathcal{C} \Leftarrow \{\mathcal{C} \backslash \{S_i \cup S_j\}\} \cup S_k$
9: Loop

Algorithmus 2 : Segmentation of the local affinity graph G

the threshold ϵ_{ab}, then the S_a and S_b will be merged as one segment. Here, the threshold ϵ_{ab} is estimated as follows:

$$\epsilon_{ab} = \min(I_a + \frac{\delta}{|S_a|}, I_b + \frac{\delta}{|S_b|}) \tag{4.16}$$

where $|S|$ denotes the size of the segment S and δ is a constant parameter setting the initial threshold. If $|S_a| = 1$ and $|S_b| = 1$, we have $\epsilon_{ab} = \delta$. This merging process is performed repeatedly until all the edges are traversed. In Algorithm. 1, we provide a detailed description of how the segmentation of local affinity graph works. According to the output of the graph partition, in the neighborhood of a center supervoxel, its connections can be identified by the group of connected nodes in its local affinity graph. Compared the conventional segmentation methods only considering the pairwise information between two patches or points, the local information can increase the robustness and reliability when dealing complex structure in the scene. Besides, with graphical model utilized, the threshold for estimating connections can be achieved in an adaptive way.

4.3.4 Assembling level: aggregation of supervoxels

In the assembly-level, connected supervoxels will be aggregated together in a greedy process, in order to form a complete segment. To achieve this process, we do not utilize seeds for a region growing process but instead process each supervoxel sequentially, with the connection relations of all its adjacent supervoxels examined. It means that if two supervoxels are connected, they will be merged into one segment. During the aggregation of connected supervoxels, a refinement process is carried out, consisting of two aspects: the cross-validation between aggregated supervoxels and the nearest grouping of single supervoxels. The cross-validation is conducted to ensure the correctness of the connectivity. For V_i and V_j, after segmenting the local affinity graph of V_i, if V_i is identified as connected to V_j, then in the segmentation of the local affinity graph of V_j, the V_j should also be connected to V_i in turn. Otherwise, these two supervoxels will be regarded as disconnected. The nearest grouping is designed for the isolated supervoxel that grouped not in any clusters. For such supervoxel, its probability $P(S_{ij}^{\pm}|A_i, A_j)$ to all its neighbors V_k are compared, and the neighbor who has the most considerable affinity will be regarded as to be connected to this supervoxel.

5 Reconstruction of linear structural elements in construction sites

In this chapter, we present an application of reconstructing linear objects from construction sites [Xu et al., 2015, 2016a, 2018c]. As an example, a data-driven framework of detecting and reconstructing the scaffolding components, including tubes, toe-boards, and decks, from the photogrammetric point cloud generated by multi-view stereo matching of a construction site with a complex environment (see Fig. 5.1). Our proposed framework consists of two parts: one part concerns the strategy based on projection and methods of grouping and slicing planar surfaces for detecting and extracting points of scaffolds from the construction site. The other part is related to point feature derivation using a new 3D local feature descriptor, Linear Straight Signature of Histograms of Orientations (LSSHOT), designed for extracting points having linear features. In this novel local feature descriptor, we utilize the robust axis of linear shape objects instead of the eigenvector as the principle direction of the objects, which is as the main axis of LRF of the point. The proposed framework can make proper preparation for the further reconstruction work of as-built BIM and provide auxiliary information about the monitoring of the construction process. In Fig. 5.2, we show the core methods corresponding to the ones we have shown in our research frame in Chapter 1.3.

5.1 LSSHOT: linear straight signature histogram of orientations

The proposed LSSHOT descriptor is an extension of SHOT descriptor [Salti et al., 2014], which is intended for the feature extraction of points being part of linear objects. In SHOT descriptor,

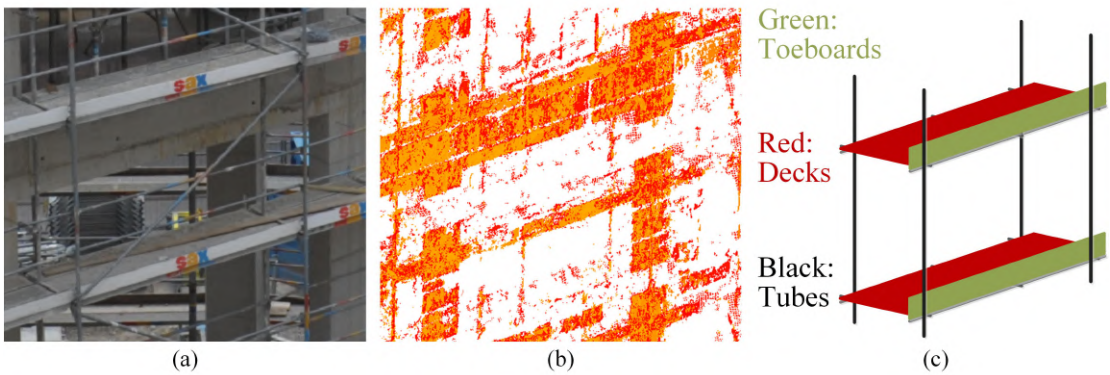


Figure 5.1: Illustration of modeling scaffolds from the point cloud. (a) Image scene of scaffolds. (b) Photogrammetric point cloud generated. (c) Expected model of reconstruction.

Figure 5.2: Methods for reconstruction of linear structural elements in construction sites.

the axes of the LRF are defined by the eigenvectors of the points within the support region. However, in the case of the photogrammetric point cloud, due to the uncertainty of stereo matching, it commonly includes more outliers and noise, notably influencing the accuracy of eigenvector estimation. To solve this problem, we utilize the refined first principal axis of linear shape objects instead of the eigenvector as the main axis of LRF. Besides, we replace the spherical support region with a cylindrical support region along the refined first principal axis, for the sake of better utilization of linear characteristics. The calculation of our descriptor contains four core steps: (1) determination of principal direction; (2) setting of Semi-Local Reference Frame (SLRF); (3) encoding the feature of points and (4) accumulation of histogram. Fig.5.3 gives a flowchart of LSSHOT descriptor with involved methods and algorithms.

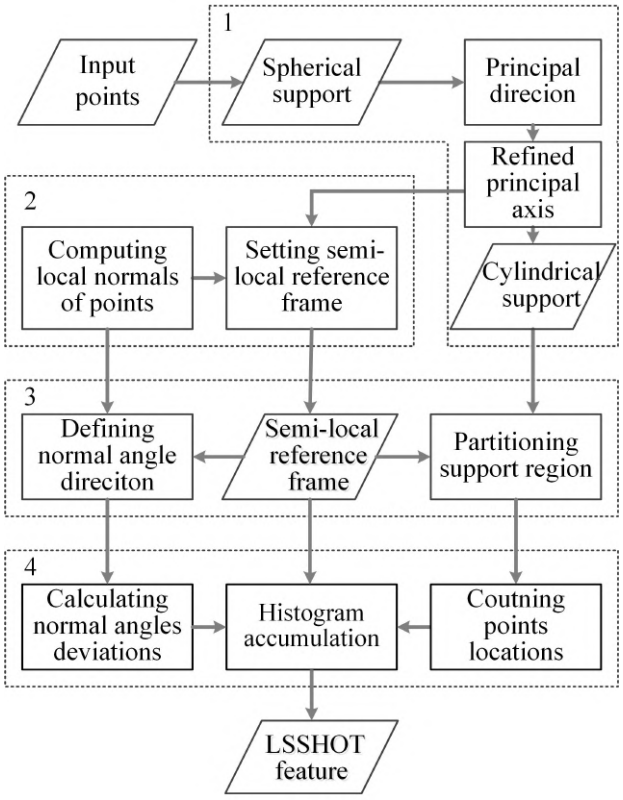


Figure 5.3: Overview of the feature value computation of LSSHOT.

5.1.1 Determination of principal axes

The first step of LSSHOT descriptor is to compute the principal axes of points in the neighborhood using principal component analysis (PCA) and maximum likelihood estimation sample consensus (MLESAC) algorithm [Torr & Zisserman, 2000]. For this task, as shown in Fig. 5.4, the neighborhood of a key point (hereinafter support) is firstly a sphere region centered on the key point and its north pole oriented with the vertical axis Z_0 of the global coordinate frame. Employing PCA, the eigenvector with the largest eigenvalue in the covariance matrix, calculated

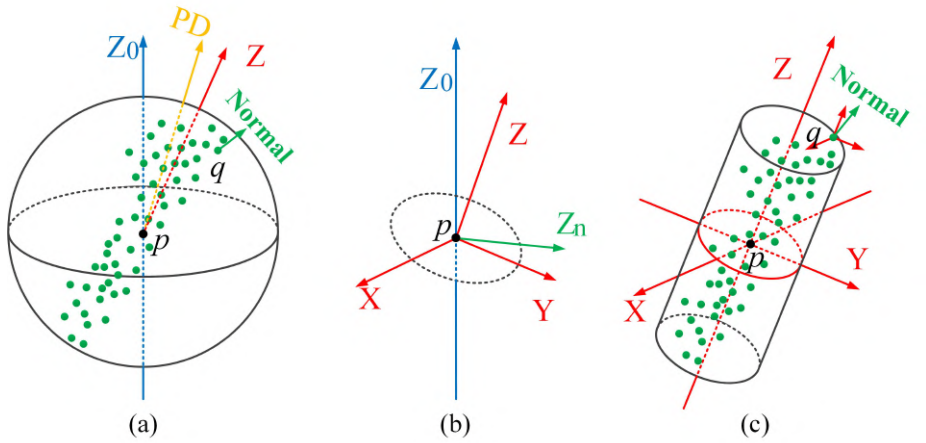


Figure 5.4: Semi-local reference frame of LSSHOT. (a) Calculation of principal direction in the spherical support. (b) Semi-local reference frame. (c) SLRF and points within the cylindrical support.

by coordinates of all the points in the support region, is chosen as the first estimate of the principal direction. Following this, a refinement of the principal direction is conducted via line fitting using MLESAC algorithm in the region around the axis of principal direction. Unlike the random sample consensus (RANSAC) algorithm which selects the solution that maximizes the number of inliers, the MLESAC finds the solution minimizing the residuals when fitting the geometric model [Torr & Zisserman, 2000], to obtain the minimal set of inliers with the most support on the condition that the percentage of inliers is unknown. For fitting the first principal axis with a linear model, the cost function C used for MLESAC is given as follows:

$$C = \sum_i f_\gamma(|x_p - x_i - n_x \cdot t|^2 + |z_p - z_i - n_z \cdot t|^2 + |z_p - z_i - n_z \cdot t|^2) \tag{5.1}$$

$$f_\gamma(e) = \begin{cases} e^2 & \text{if } e^2 < T^2 \\ T^2 & \text{else} \end{cases} \tag{5.2}$$

where (x_p, y_p, z_p) and (x_i, y_i, z_i) are the coordinates of key points and inliers, respectively, and (n_x, n_y, n_z, t) are the parameters of the straight line in 3D space. In Eq. 5.2, T is set to be 1.96 σ, where σ denotes the standard deviation of Gaussian distribution.

In Fig.5.4a, an illustration of the calculation of the principal direction (PD) is given. The green points indicate the candidate points in the initial spherical support. The blue Z_0 axis is the vertical axis of the global reference frame, while the yellow and red axes are the initial estimated principal direction and refined first principal axis respectively.

5.1.2 Semi-local reference frame

As argued in [Salti et al., 2014], the definition of a local reference frame (RF), invariant to translation and rotation and robust to noise and clutter, ensures a 3D descriptor with invariance to the same sources of variations. Since the orientation and linearity of the first principal axis is one of the most distinctive features for the linear objects, and the refinement of first principal axis utilizes points in the support region selectively by excluding outliers and noise points, the use of this axis can greatly increase the reliability of LRA, when compared with the axis of LRA/F defined merely by the eigenvector of points. Thus, in LSSHOT descriptor, a SLRF is defined by introducing the refined principal direction of linear objects as well as the local normal vector of

the key point. As sketched in Fig. 5.4b, the refined first principal axis is set as the $\boldsymbol{Z}$ axis of the SLRF. The $\boldsymbol{X}$ and $\boldsymbol{Y}$ axes are calculated with Eq. 5.3, where $\boldsymbol{Z}_n$ is the local normal vector of the key point p. The key point p is chosen as the origin of SLRF.

$$\begin{aligned} \boldsymbol{X} &= \boldsymbol{Z} \times \boldsymbol{Z}_n \\ \boldsymbol{Y} &= \boldsymbol{Z} \times \boldsymbol{X} \end{aligned} \tag{5.3}$$

Although this SLRF is associated with the directional character of the structure where the key point laid, namely, and it forgoes the ability of being independent to the structure or object that the key point located in, it is still invariant to translations and rotations. To increase the reliability of the calculated local normal vectors, a strategy similar to the one proposed in [Salti et al., 2014] is used, which assigns smaller weights to distant points in order to increase repeatability in presence of clutter. In this method, to improve robustness to noise, the covariance matrix $\boldsymbol{M}$, calculated by all points lying within the surrounding area for local normal vector estimation with a radius of r, is used to compute the normal direction following its eigenvector. Similarly, for the sake of efficiency, the computation of centroid points is replaced by the key point p itself [Tombari et al., 2010]. The $\boldsymbol{M}$ matrix is calculated as follows:

$$\boldsymbol{M} = \frac{1}{\sum\limits_{i:d_i \leq r} (r - d_i)} \sum_{i:d_i \leq r} (r - d_i)(p_i - p)(p_i - p)^T \tag{5.4}$$

where p_i denotes the point in the support region for normal calculation and d_i stands for their distances to the feature point.

5.1.3 Encoding of features

Once the SLRF is defined, a cylindrical support centered at key point p and its north pole oriented with the Z-axis of SLRF is chosen as shown in Fig.5.5c. The points lying in the cylindrical support region are considered as the candidate point dataset for delineating the features. In this work, the length of cylindrical support region is set to 0.2 m, while the radius of cylindrical support region is 0.05 m. Similar to the SHOT descriptor, the proposed descriptor considers both the signature of topological distribution and the histogram of geometric features for points in the cylindrical support region. More specifically, the signature of topological distribution represents the spatial information concerning the location of points within the cylindrical support region, acting as a basic structural feature. The histogram of geometric features is implemented by a set of local histograms of each volume of the support region, relating to the normal directions of points inside the cylindrical support region according to the SLRF. As shown in Fig. 5.5a, the cylindrical support region is segmented into six volumes along the Z-axis. The X-Y plane is subdivided into eight bins with an interval of 45 degrees (Fig. 5.5b), while the angle between the normal and Z-axis is partitioned to six bins (Fig. 5.5c), regardless of the sign of direction. Therefore, as extracted features, the overall histogram of LSSHOT descriptor will consist of 6×6×8=288 bins in total to characterize a key point.

5.1.4 Accumulation of histograms

For each candidate point q within the support region, we first compute its location concerning the SLRF to project its location into one of the six volumes, as shown in Fig. 5.6a. Then, for the local histogram of the normal directions in each volume, the SLRF is translated to the candidate point q, in order to calculate the normal direction angle. The local normal vector of this candidate point q is compared with the axes of SLRF to bin the angle of the normal in the local histogram. Figs.

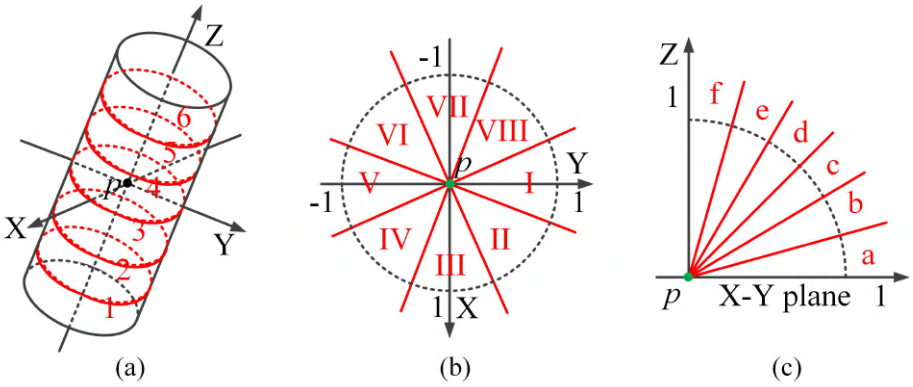


Figure 5.5: Design of volumes and bins for LSSHOT. (a) Volumes of LSSHOT along the Z-axis of cylindrical support. (b) Bins of local histogram in the X-Y plane of cylindrical support. (c) Bins of local histogram between the Z-axis and X-Y plane of cylindrical support.

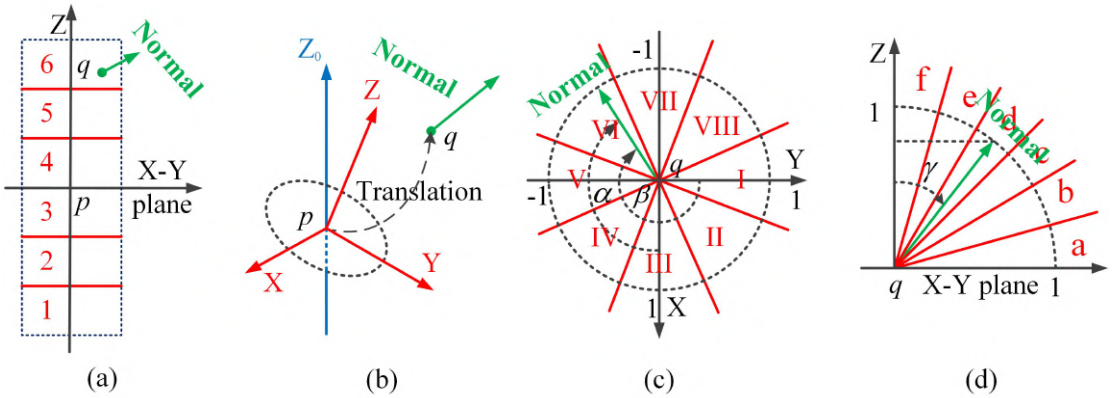


Figure 5.6: Accumulation of bins for LSSHOT. (a) Accumulating location of point in signature structure. (b) Comparing the angle of normal with SLRF. (c) Accumulating deviation of normal in X-Y plane. (d) Accumulating deviation of normal between Z-axis.

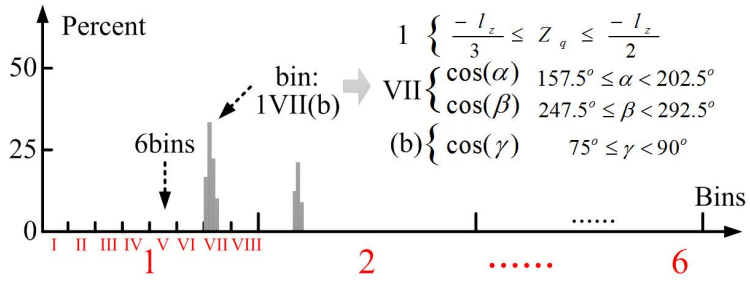


Figure 5.7: Illustration of accumulated histogram of LSSHOT.

5.6b-5.6d give illustrations of how the directional angle of the normal is compared with the defined SLRF. The signature of the topological distribution is acquired by counting the numbers of points falling in each volume. With respect to the local histogram corresponding to each volume, we accumulate the angle of the normal vector for each candidate point q situated in this volume into bins of local histogram according to the cosine of angles α, β, and γ between the normal vectors at q and the axes of SLRF. Figs. 5.6c-5.6d show the detailed accumulation of the angle of the normal of a candidate point. The reason to use the cosine is twofold: it can be computed rapidly, because $\cos(\alpha)$ is directly equal to $\boldsymbol{N} \cdot \boldsymbol{X}$; an equally spaced binning on $\cos(\alpha)$ is equivalent to

a spatially varying binning on α. Moreover, it is noteworthy that for the comparison between normal vector $\boldsymbol{N}$ and $\boldsymbol{Z}$, the sign of direction is not taken into consideration, which means that γ is limited to $[0^o\ 90^o]$. Fig. 5.7 sketches an example bin in the final histogram of a key point p using our descriptor.

5.2 Reconstruction of scaffolding components

The workflow consists of two phases: the detection and the reconstruction of scaffolding components, which is illustrated in Fig.5.8, with the involved methods and sample results illustrated.

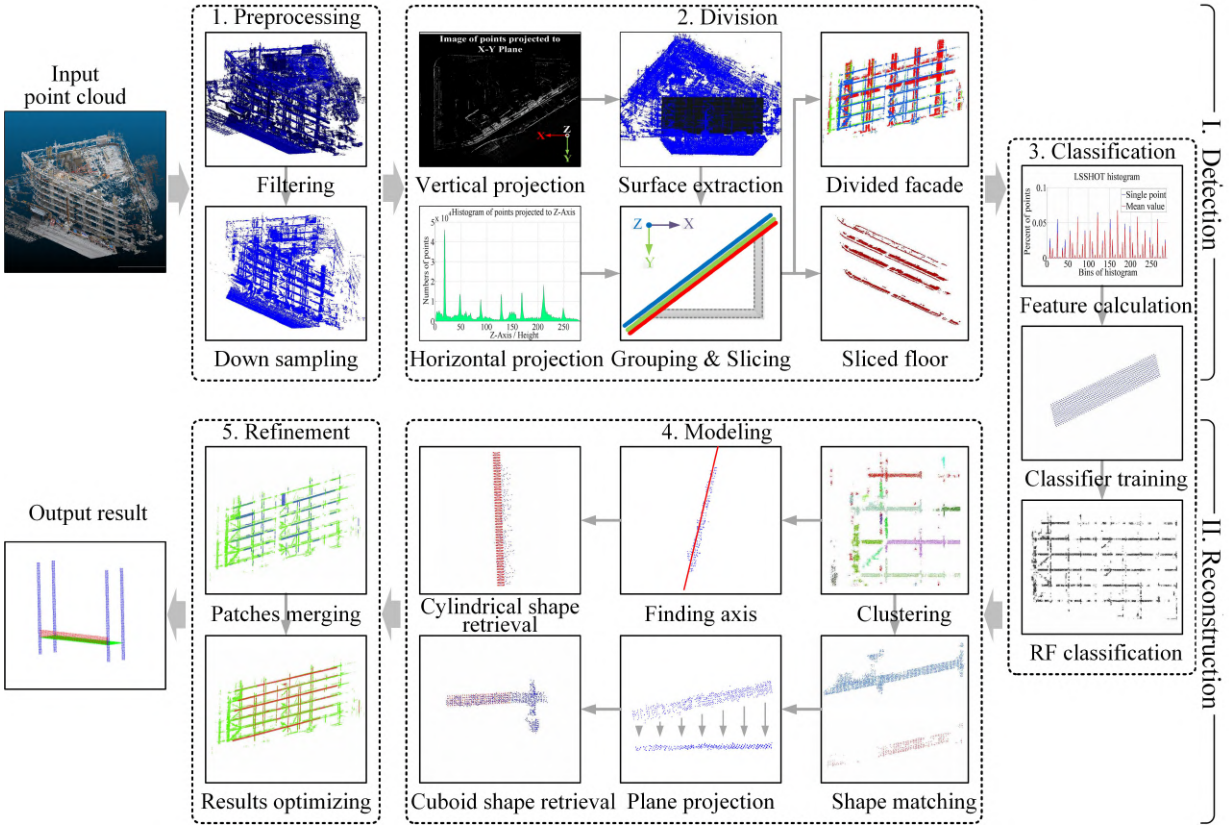


Figure 5.8: Overview of the scaffolds reconstruction procedures.

As for the detection phase, the first step is a preprocessing of the raw point cloud, in which voxel grid-based filtering is applied to sample down the point cloud, following a statistical filtering aiming at removing outliers. In the second step, the aim is twofold: to separate points of facades including building structures and scaffolds from the point cloud of the entire construction site, while the other one is to distinguish points that are associated with scaffolding components from the isolated facades. For the former task, a projection strategy is adopted. Then, planar surfaces acting as primitive elements for further grouping and slicing processing are extracted by model fitting algorithm. In respect of the latter task, a unique feature of the building surface with scaffolds is exploited by considering the parallelism and Euclidean distance between building surfaces and rows of scaffolds. The planar surfaces of building surface and corresponding scaffolds are grouped as an entire building facade via a transformation and clustering of their parameters. Once the division is accomplished, in the third step a supervised classification is adopted, in which points linked to specify scaffolding components are classified. Features of points are extracted

employing the proposed LSSHOT descriptor. With the help of random forest classifier, the points belonging to different types of objects are discerned. For the reconstruction phase, in the fourth step, points belonging to different scaffolding components are segmented into small clusters. Shapes of objects are then retrieved using cylinder and cuboid models based on these clusters. Each reconstructed object is delineated with parameters of its corresponding geometric model. Finally, in the last step, small and single reconstructed patches are merged, in order to obtain a complete parametric representation of objects.

5.2.1 Preprocessing of point cloud

The preprocessing of the point clouds aims at downsampling the row point cloud data, including two steps: statistical filtering and voxel grid based filtering so that the computation cost can be decreased and the outliers in point clouds can be removed.

To remove outliers, a statistical analysis on the neighborhood $N(P)$ of each point p forming by k nearest neighbors (KNN) is conducted [Rusu & Cousins, 2011]. For each point $p_i \in P$, the mean distances $\bar{d}_i$ from it to all its neighbors are computed. Assuming that the points follow a Gaussian distribution with a standard deviation σ_k and mean value u_k, those points whose mean distances larger than the interval defined by the global distance and standard deviation are considered as outliers and filtered. The remaining point cloud P^* is approximated with the remaining points ${p^*}_i$, which is shown as follows [Rusu, 2010]:

$$P^* = \{{p^*}_i \in P | (u_k - \alpha \cdot \sigma_k) \leq \bar{d}_i \leq (u_k + \alpha \cdot \sigma_k)\} \tag{5.5}$$

where α is defined as a given factor of desired density restrictiveness. In our work, α is set to three while the k is equal to ten.

For the voxel grid based filtering of datasets, a voxel grid is created by means of cubic cells of certain size s_c. Voxels in the grid are the 3D analogue of boxes in space, which partition 3D space into uniform 3D cells (e.g., typically cubes) [Rusu & Cousins, 2011]. In the filtering process, for the points within in each voxel V, the set of points will be approximated and represented with the voxel centroid p_c. This voxel grid based filtering can evenly distribute the density of points to avoid the overly dense and sparsely sensed points in specific areas. It is remarkable that the resolution of voxel can affect the preservation of details. Thus, the size r_c of voxel should be determined according to the real demand of reconstruction. In our work, r_c is set to 0.04 m.

5.2.2 Division of building facades

The division of building facades is composed of three critical stages: (1) the projection and selection of points; (2) the extraction of planar surfaces; (3) the grouping or slicing of planar surfaces. In the first stage, vertical structures, including the building walls, rows of scaffolds, are discerned. Following these, as basic elements, planar surfaces are extracted from all the points selected. Parameters of planar surfaces are taken into consideration for identifying the points belong to which planar surface.

Vertical projection

The vertical projection of point cloud is a conversion from 3D point data to 2D image data by projecting points to the ground plane along the vertical direction. As a result, a projection image is obtained, with its gray values standing for the numbers of points mapped to the bin of pixels. The pixel size is around triple diameter of the vertical tube section of scaffolds, ensuring that a single vertical tube can be projected into pixels with limited neighbors. For selecting vertical

structures, we assume that the projected points belonging to the building vertical surfaces and scaffolding components (e.g., walls, tubes, and boards) have a higher assembling density along the vertical direction. Thus, in the projection image, pixels are selected according to their intensities, namely the number of points falling into it. Points corresponding to the pixels with high intensity, showing up as bright piece-wise lines or dots in the image, are expected to be selected as points of vertical structures. To extract these bright pixels, an algorithm using the local maximum threshold is proposed. A sliding window of a given size is applied to the projection image, by which the standard deviation σ_l and mean value u_l of all the pixels P_l covered by sliding window are obtained. In light of the Chebyshevâs inequality, using a local maximum threshold τ_p can be determined by a given global estimation of noise percentage r_p from the whole image, with the help of σ_l and u_l. Thus, the set of pixels P_v representing vertical structure is selected following Eq. 5.6.

$$P_v = \{p_{ii} \in P_l | p_i \geq (u_l + \beta \cdot \sigma_l), \beta = \sqrt{\frac{1}{1 - r_p}}\} \tag{5.6}$$

where the local maximum threshold τ_p for every sliding window is equal to $u_l + \beta \cdot \sigma_l$, while r_p is the noise percentage of the whole image used to approximate the noise percentage in a local window. In this work, the r_p is set to 0.1.

Horizontal slicing

For extracting horizontal structures, we adopt the idea from [Oesau et al., 2014], in which horizontal structures are assumed appearing as peaks in the distribution of points mapped along the vertical axis, as horizontal structures generate a high number of samples owning similar heights. A histogram can be obtained via a projection to the vertical axis, the peaks of which correspond to horizontal structures and then located by the use of the mean shift algorithm [Cheng, 1995; Oesau et al., 2014]. Two thresholds τ_l and τ_h are given to constrain the finding of peaks, limiting the lower bound of the height of Z-axis for slicing and the minimal number of points projected in a valid bin, respectively. Since the accumulated histogram is discrete, the bin size (i.e., the thickness of the sliced point cloud) is consistent with the size of voxel grid for a maximal precision. In our case, the bin size is set three times the voxel size used in the downsampling. By the projection and the selection of local peaks, the points belonging to horizontal structures are sliced from the entire point cloud.

Planar surfaces extraction

In this stage, an assumption is made that the major facades of the building are mainly constructed with a planar shape, for example, the vertical walls, inner and outer rows of the scaffolds. A plane-fitting algorithm based on RANSAC is applied [Schnabel et al., 2007]. Considering the major surfaces of facades consistently having a vertical or horizontal direction, a constraint under the direction of normal vectors is added to the plane fitting process.

Surfaces grouping and slicing

Vertical planar surfaces being part of the same facade including the inner and outer rows of scaffolds as well as the building surface (see Fig. 5.9) are firstly grouped and identified. The horizontal planar surfaces are then segmented confirming the points belong to the decks of scaffolds.

For grouping vertical planar surfaces, we take advantage of the distinctive structure of facades in unfinished buildings showing up as a “sandwich-like” arrangement, with the rows of scaffolds located in parallel with the building surface and having a fixed distance between each other. In Fig. 5.10, the relationship between the building surface and the inner and outer rows of the

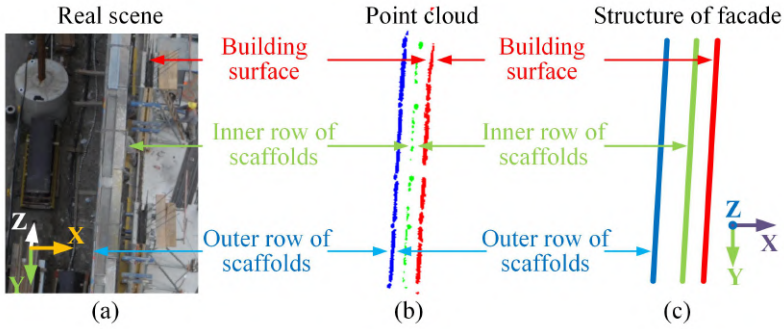


Figure 5.9: Sketch of "sandwich-like" structure for a facade with scaffolds. (a) The actual scene of the facade. (b)Nadir view of real point cloud of facade. (c) Designed structure of facade.

scaffolds is sketched. By utilizing this specific structure, planar surfaces about scaffolds and building wall surfaces is confirmed.

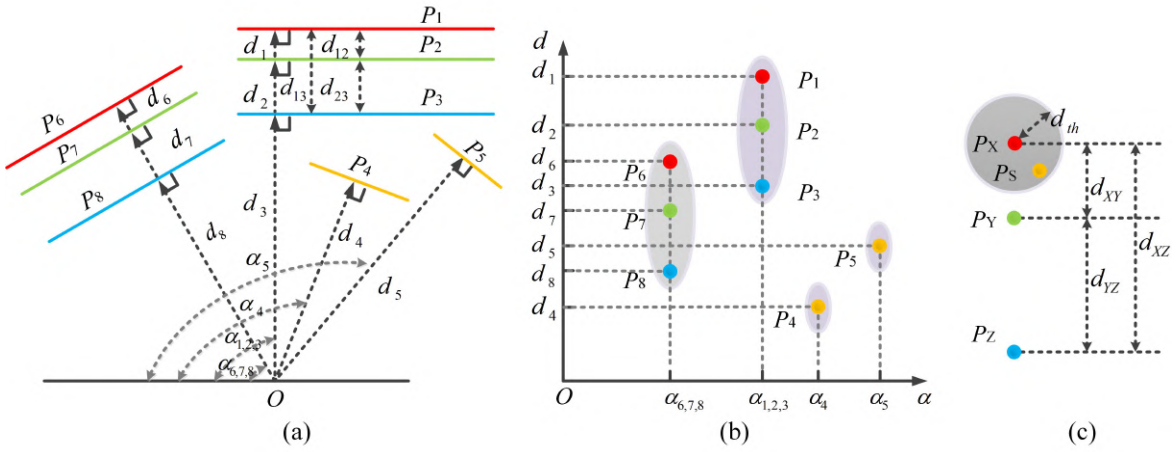


Figure 5.10: Grouping of scaffolds and building wall surfaces. (a) Parameters of planar surfaces in polar coordinate. (b) Statistics of parameters of planes. (c) Relative distances between planes in one group.

To begin this process, all the vertical planar surfaces are set in a polar coordinate frame defined by a reference point and axis, with their distances between planes and reference point as well as the angles of normal vectors calculated. As shown in Fig. 5.10a, d_i and α_i denote the distance and angle of vertical planar surface i. Afterward, a coordinate transformation of all calculated distances and angles is conducted, in which parameters of planar surfaces are projected to a Cartesian coordinate system, regarding the angle and distance as X- and Y- axis, respectively. Theoretically, dots representing parallel and adjacent planar surfaces will be clustered in this coordinate system (see Fig. 5.10b), with an approximately coincident x value and a contiguous y value. If a dot belongs to a cluster of more than three dots with a given buffer threshold, the planar surface represented by this dot will be recognized as a planar surface of one group, and this cluster will also be regarded as a representative of one group. Once a group of planar surfaces is found, the relative distances between planar surfaces in one group are calculated to distinguish the types of planar surfaces (i.e., scaffolds or building wall surfaces) via these relative distances according to the given threshold of distance from the design specification. In our case, the range bounded by the outer and inner rows of scaffolds is designed to be around 0.8 meters regarding the DIN 4420 national standard of Germany about the design of scaffolds.

As for the horizontal planar surfaces, the areas in a horizontal planar surface representing decks of scaffolds are segmented by using the boundaries of two rows of scaffolds in one facade, which are illustrated in Fig. 5.11.

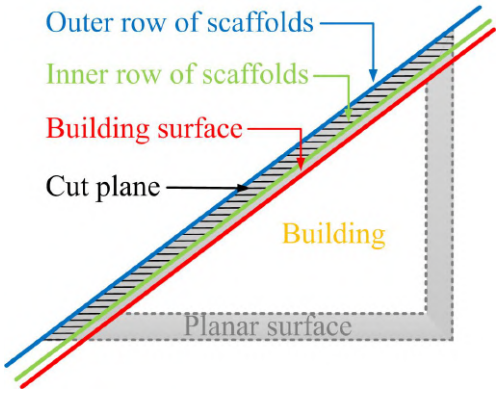


Figure 5.11: Illustration of cutting horizontal planar surface.

5.2.3 Classification of points

The classification of points aims to distinguish points of different scaffolding components, from the separated facades. Points associated with decks of scaffolds are directly obtained from the results of cutting horizontal planar surfaces, therefore in this process, our emphasis is put on the classification of points related to tubes and toeboards of scaffolds. A supervised classification strategy with the random forest (RF) classifier is conducted based on the features extracted by the LSSHOT descriptor.

Feature extraction

Features used in the classification are extracted via the LSSHOT mentioned above descriptor. The principle and operational process of our descriptor has already been depicted in previous Sections. According to the LSSHOT descriptor, for each point, a histogram of 288 bins is calculated using the geometric information of points in the neighborhood. Thus, each point has 288 attributes for the classification. This also indicates that for the training process of RF classifier used, we need at least 288 training samples in order to obtain a reliable classification.

Supervised classification

The supervised classification in our work is conducted with the random forest classifier [Breiman, 2001], which is a combination of tree-structured classifiers created by a randomizing vector sampled independently from the input vectors, and each decision tree voting uniformly for selecting the most popular class to classify the input vectors [Pal, 2005]. The RF classifier employed in this study is based on a combination of geometric features at each node to grow a tree, which makes it less susceptible to over-fitting [Breiman, 1996] due to the strong law of large numbers [Stone & Feller, 1969] as the number of trees increases [Pal, 2005]. In the training, the bagging method is employed for each feature combination to generate a training dataset by randomly drawing with replacement N examples, where N is the size of the original training set [Breiman, 1996].

5.2.4 Modeling of patches

The patch modeling is designed to reconstruct the small patches of each kind of object from the classified points, and then endow them with regular parametric representations. In this step, two processes are carried out: (1) Clustering of classified points. (2) Delineating parametric representations of small patches.

Points clustering

The points clustering aims at cutting the classified points into small clusters in advance of the geometric reconstruction, to make the reconstruction process more accurate and avoid the interference of similar objects. The clustering is based on the seeded region growing algorithm [Adams & Bischof, 1994; Rabbani et al., 2006; Vosselman & Maas, 2010] used to cluster labeled points into small patches. Here, the seeds for region growing methods are set randomly.

Geometric delineation

In the delineation process, cylindrical patches, namely patches of tubes, and âboard-likeâ patches, including patches of decks and toeboards, are modeled separately, with their geometric representations obtained. Two approaches are designed considering different geometric characteristics of these two kinds of elements, in order to delineate objects with different geometric models. The reconstructed objects are parametrically represented by the boundary representation (B-rep), which is a geometric shape described by surface elements and their connection information. In our case, each surface of the reconstructed object is represented and shown by resampled points, with the parameters of its boundary known. In Fig. 5.12a, a parametric representation of a cylindrical object is given. In this example, the parameter of surface boundaries and topological relationships of the surfaces is known, and the representation is expressed in the form of a resample point cloud. In Fig. 5.12, brief schematic diagrams illustrate these different procedures of geometric delineation.

Cylindrical patches are regarded as cylinders with length and radius. In the modeling of tubes, the RANSAC algorithm is adapted to fit the symmetry axis of the cylindrical model, and then the cylinder is modeled with a given radius. After the modeling of cylindrical patches, small fragments nearby are combined to form a complete representation, which can reduce the discontinuity stemming from the clustering and the occlusion of dataset itself.

“Board-like” patches are deemed cuboid with small thickness. Since the candidate point cluster always contains sparse or irregular outliers resulting from the clustering process, a shape matching process is carried out, in which the features of points in both candidate point cluster and training object are calculated by LSSHOT descriptor. The matching takes place between points in the feature space of descriptor with Euclidean distance metric. Afterward, the matched points of one cluster are projected to the principal plane defined by the “board-like” shape, in order to sketch the contours of the shape in a 2D space. Following this, an alpha shape algorithm is used to obtain a polygon representing the projected shape boundary. A rectangular boundary is approximated on the basis of the hull via a rotating calipers algorithm [Toussaint, 1983]. Finally, the cuboid representation is recovered utilizing the rectangular boundary and the thickness. Moreover, since scaffolding components are usually standard objects, prior knowledge including radius of the section of tube and thickness of the board, which is referred from the standard DIN 4420, is utilized to optimize the boundary of objects.

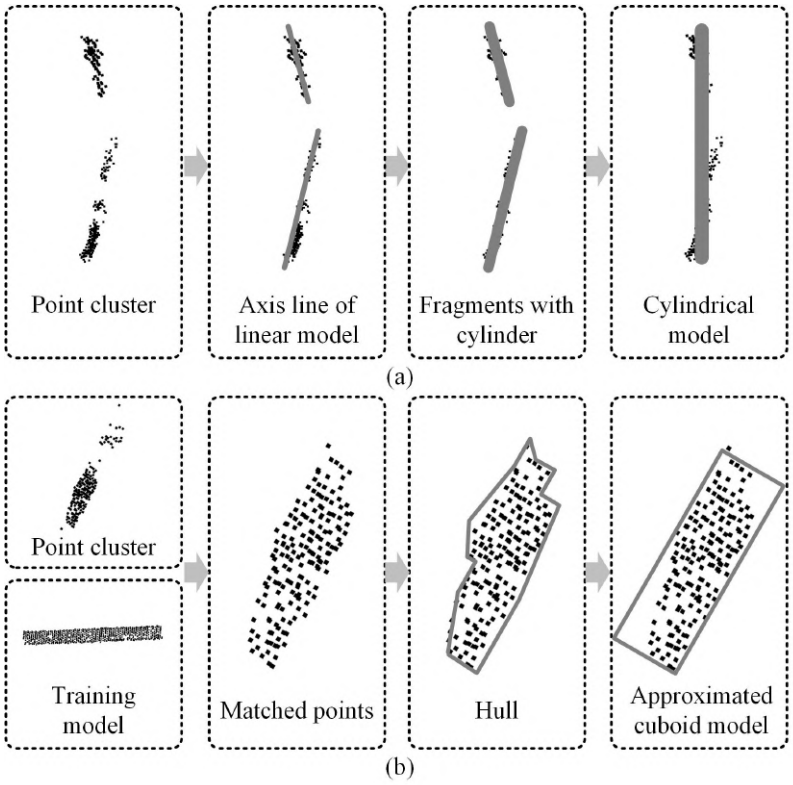


Figure 5.12: Modeling scaffolding components. (a) Modeling cylindrical objects. (b) Modeling "board-like" objects.

5.2.5 Refinement of results

In the final step of the whole workflow, we address two problems: (1) Merging the reconstructed small patches to form complete objects. (2) Optimizing the parameters of geometric representations for merged results.

Patches merging

The merging process aims at fusing fractional patches belonging to the same scaffolding components, in order to achieve complete reconstruction of objects. The similarity measurement weighs differences of patches defined by spatial positions and geometric features. For each modeled patch o_i, the start and end points p_i^s, p_i^e (the centers of sections), as well as the directional vector $\boldsymbol{v}_i$ of the patch, serves as criteria. Concerning the similarity measurement for merging, the normalized cut algorithm [Shi & Malik, 2000] is exploited, which is also serving for the merging of segments in [Polewski et al., 2015]. Here, directional vector $\boldsymbol{v}_i$ is defined as $\frac{p_i^s - p_i^e}{||p_i^s - p_i^e||_2}$. To eliminate the ambiguousness of signs caused by selecting start and end points, a known viewpoint p_v is introduced to orient the directional vector by satisfying Eq. 5.7 with appropriate endpoint.

$$\boldsymbol{v}_i \cdot (p_i^e - p^v) < 0 \tag{5.7}$$

For a set of n patches $O = \{o_1, o_2, ..., o_n\}$ belonging to a same object, the similarity $W(i,j)$ between patches o_i and o_j in similarity matrix W is computed following Eq. 5.8, in which a Gaussian kernel function is involved. In addition, considering the parallelism of directions of

merging patches, a vector constraint is applied, which is related to the inner product of directional vectors $\boldsymbol{V}_i$ and $\boldsymbol{V}_j$ of patches.

$$W(i,j) = \begin{cases} e^{\frac{-m(o_i,o_j)}{2\cdot\sigma^2}} & \text{if } \frac{\langle \boldsymbol{V}_i, \boldsymbol{V}_j \rangle}{|\boldsymbol{V}_i|\cdot|\boldsymbol{V}_j|} > \gamma \\ 0 & \text{else} \end{cases} \tag{5.8}$$

Here, in Eq. 5.8, γ and σ denote the threshold for this constraint and the parameter of the Gaussian kernel and the angle between directional vectors, respectively. In this work, σ is set to 1.0, and γ equals to 0.5. The $m(o_i, o_j)$ denote the measuring distance between patches, which are defined as $\min\{||p_i^s - p_j^e||_2, ||p_j^s - p_i^e||_2\}$. It should be noted that, for the cylindrical model applied to the tube, which is axisymmetric, the constraint of vectors is merely related to the direction of its axis, while for the cuboid model used for toeboards and decks, which is only bilateral, another constraint associated with the second axis of cuboid model is needed. Only if both vectors of the two axes of cuboid model meet the vector constraint, the similarity between patches will be taken into account. As described in [Shi & Malik, 2000], the eigenvector of similarity matrix corresponding to the second smallest eigenvalue f_2 of the normalized Laplacian matrix is obtained via the eigenvalue decomposition and Laplacian matrix obtained from the similarity matrix. The k-means clustering algorithm is performed on f_2 to partition points into two subspace clusters. The bipartition is recursively conducted until k clusters are obtained, with a stopping criterion based on the normalized cut value used. The patches in one cluster are regarded as components being part of one geometric object and refined with optimized parameters.

Results optimization

The optimization of the results includes two sub-steps: the fusion of geometric representations and the refinement of the geometric parameters. The fusion of geometric representations is performed on the merged patches. Once the merging of patches is accomplished, for the small patches in one merged cluster, the resampled points of their representations are taken into consideration for modeling according to the algorithms we state in Section 3.2.4. For the patches of cylindrical object, the representing points stand for the start and end points as well as geometric centroids of each patch, whereas for the patches of "board-like" objects the representing points denote the geometric vertexes and centroid points of each patch. With respect to the refinement of the geometric parameters, the geometric constraints, including the directions of axes as well as the radii and widths of the section, are applied to the modeled geometric representation in order to get the optimized parameters of the models, since the scaffolds are standard components with fixed geometric sizes and angles of installation.

6 Reconstruction of planar building objects in built environment

In this chapter, we propose a framework of building elements reconstruction covering the entire process from the acquired MLS point clouds to the output surface model of planar objects[Sun et al., 2018; Xu et al., 2018b]. The framework consists of two major phases: (I) semantic labeling of point clouds using context-based features and global graph-based optimization and (II) modeling of planar building elements using global graph clustering and cell decomposition. In these two-phase, the global graph optimization plays a vital role in both spatial smoothing and geometric partition.

We firstly present a supervised semantic labeling method designed for classifying MLS point clouds. Here, a novel geometric feature extraction strategy, detrending the local tendency of the geometry, is proposed, which is proved to be effective and efficient for describing local geometry from the 3D scene. Then, instead of using individual points as fundamental elements, the supervoxel-based local context is designed to encapsulate geometric characteristics of points, providing a flexible and robust solution for feature estimation. Conventional segment-based methods heavily depend on the quality of obtained segments, we compromise this issue by using the over-segmented supervoxels. In this step, a boundary refined supervoxel generation algorithm is developed. Finally, a regularization processing using global graph optimization is also applied to improve the quality of the classification result. After the semantic labeling of point clouds, we propose a bottom-up reconstruction method that utilizes global graph-based optimization and boundary representation with cell decomposition, enabling an automatic and unsupervised segmentation of point clouds. Then, A planarity-based selection and model-fitting based refinement for the detection and extraction of planar surfaces is developed. Unlike traditional model fitting based planar extraction method, without iterative process, our plane extraction method is more efficient and adaptive to the real condition of urban scenes. The calculation of smoothness and planarity can provide the estimation of coefficients for the plane model. Afterward, the boundary points of the extracted plane are extracted by the alpha-shape. Line segments are extracted and optimized by the energy minimization. At last, a cell decomposition method is adopted to get the polygon representation of extracted planes.In Fig. 6.1, we show the core methods corresponding to the ones we have shown in our research frame in Chapter 1.3.

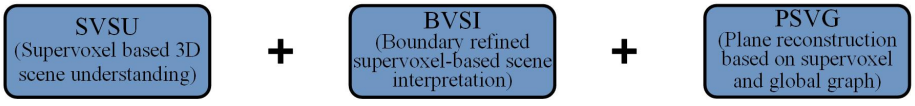


Figure 6.1: Methods for reconstruction of planarbuilding objects in builtenvironment.

6.1 Semantic labeling of point clouds using context-based features and global graph-based optimization

In Fig. 6.2, a general workflow of our proposed method is given, involving five significant steps, namely supervoxelization and selection of local context, segment-based feature extraction, detrending of geometric features, and supervised classification. In the initial step, an over-segmentation process is implemented through the VCCS method [Papon et al., 2013]. Besides, for each supervoxel, a local context is defined, taking all the directly connected neighbors into consideration. In the second step, local geometric features of each supervoxel, as well as its connected neighbors within the local context, are calculated. Afterward, for each supervoxel, a local tendency is estimated in the feature space based on the features of all the neighboring supervoxels in the local context. Then, the geometric features of the center supervoxel are detrended by the use of the local tendency. For the supervised classification, through a training stage, an RF classifier is learned to classify objects by the use of the detrended features in complex urban scenes. Finally, a global graph-based optimization is conducted to refine the initial labels given by RF. The resultant classes have eight different objects, covering man-made terrain, natural terrain, high vegetation, low vegetation, buildings, hardscape, scanning artifacts, and cars.

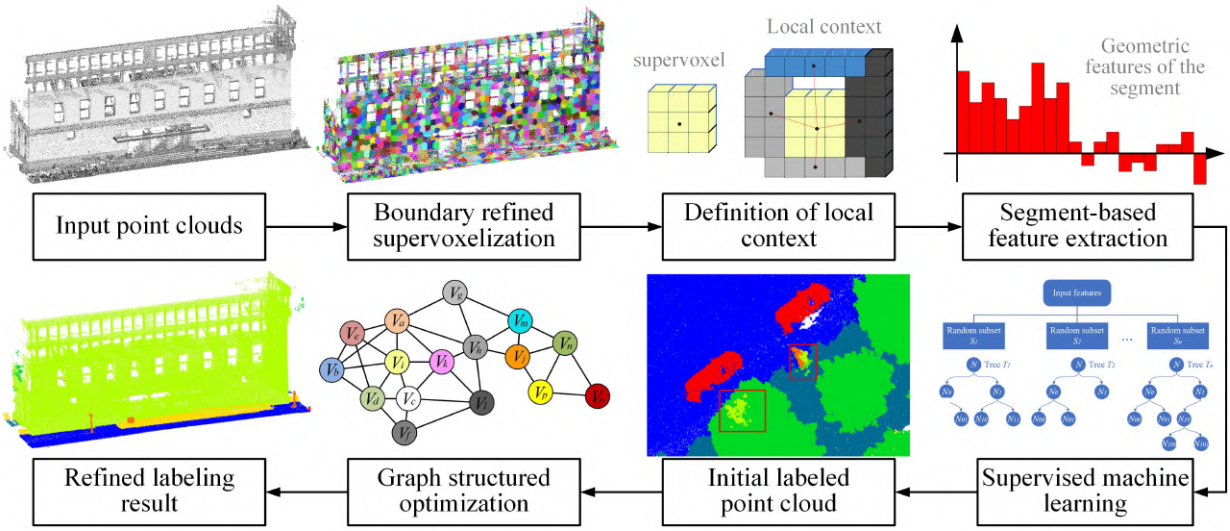


Figure 6.2: Workflow of our point cloud classification.

6.1.1 Boundary refined supervoxel clustering

To organize the entire point cloud into a supervoxel structure, space is firstly divided into a small 3D cubic grid by means of octree partitioning, which splits each node into eight equal child nodes, in order to generate the octree-based voxel structure. Compared with others point-based neighborhoods, for example, kd-tree based points structure, when using voxels as basic processing unit under an octree structure, there is no need to handle problems like uneven density resulting from mobile laser scanning. The octree structure is achieved by the approximate nearest neighbor (ANN) [Muja & Lowe, 2009] searching algorithm, which largely increases the efficiency of the neighboring searching procedure. However, for the supervoxelizatio algorithm, we have used in the former chapter (i.e., VCCS), their results always suffer from the "zig-zag" effect, since the basic element of VCCS is the cubic shape voxel [Sun et al., 2018]. To overcome this problem, we proposed a boundary refined supervoxel clustering algorithm for creating supervoxels with a point-level accuracy of their boundaries.

Our proposed boundary refined supervoxel is based on the original VCCS supervoxel, consisting of two major steps: the detection of boundary points and the refinement of boundary points. In the first step, all the points of one supervoxel will be measured by the distance from the point to the center of the supervoxel considering the local curvature [Li, 2018] exploring the spatial proximity of adjacent supervoxels in geodetic space.

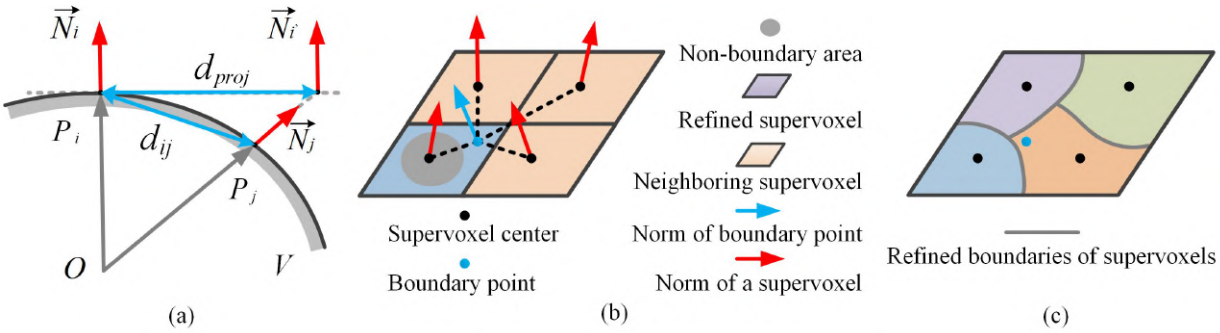


Figure 6.3: Boundary refined VCCS supervoxelization. (a) Projected distance considering local curvature. (b) Local k-means clustering of boundary points. (c) Refined boundaries of supervoxels.

As shown in Fig. 6.3a, in the supervoxel V, from the point P_i to neighboring point P_j, the distance d_{proj} is calculated by its projected point $P_j^{'}$ on the tangent plane of P_i defined by the normal vector N_i. If d_{proj} is larger than a given threshold θ, the point is regarded as a boundary point. Empirically, the θ is set to $r_{seed}/2$, where r_{seed} is the seed resolution of supervoxels. The radius size of spherical neighborhoods for estimating the normal vector is equal to the size of the voxel. Then, in the second step, a local k-mean clustering is conducted between the boundary point and the centers of neighboring supervoxels (see Fig. 6.3b). Here, the clustering is governed by a distance measure calculated in a feature space, considering the normal vectors and spatial distance:

$$D = \sqrt{w_n \cdot ||N_i - N_b||_2 + w_d \cdot ||X_i - X_b||_2} \tag{6.1}$$

where N_i and N_b are the normal vectors of the center of one neighboring supervoxel and the boundary point, while X_i and X_b are their positions, respectively. In our work, only normal vectors and spatial distance are considered during supervoxelization (also for the VCCS step in our work), which shows better performance at preserving real boundary of objects than that implemented at the voxel level. In Fig. 6.4, we illustrate the difference between the original VCCS supervoxelization and our refined boundary VCCS supervoxelization.

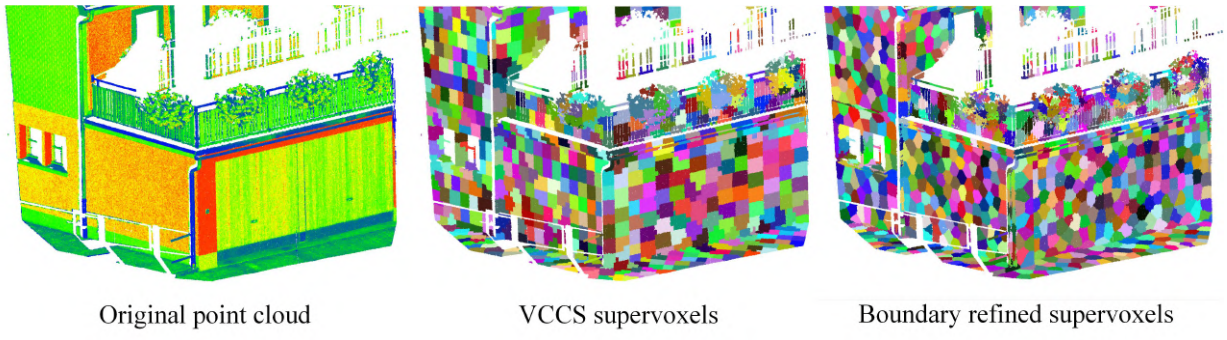


Figure 6.4: Illustration of boundary refined VCCS supervoxelization.

6.1.2 Segment-based feature extraction

Considering a large amount of 3D points containing merely spatial coordinates, we need to extract geometric features from the 3D coordinates for describing the geometry of the object. Since the supervoxel neighborhood and its adjacent graph are already aware, an appropriate representation of the local geometry is necessary. Therefore, eigenvalue-based geometric features [Chehata et al., 2009; Weinmann et al., 2015], as well as additional structural features, are introduced to tackle this problem. The eigenvalue-based geometric features are to represent the local geometry of the object, while additional structural features are introduced to represent the basic structure of the local context.

Eigenvalue-based features, including linearity L_λ, planarity P_λ, scattering S_λ, omnivariance O_λ, anisotropy A_λ, eigenentropy E_λ, local curvature C_λ as well as the sum of eigenvalues $\sum_\lambda$, can be derived according to the method presented in [Weinmann et al., 2015]. The L_λ, P_λ, and S_λ describe the dimensionality of the points, while O_λ, A_λ, E_λ, C_λ, and $\sum_\lambda$ encode statistical features for the shape description. In addition to eigenvalue-based features derived from the 3D structure tensor, height features, orientation features(i.e., normal vectors and verticality), and surface features (i.e., local point density D) are also introduced as additional information for the geometry description. Furthermore, considering the interaction between the supervoxel itself and the local context, we also utilize relative position R_p, relative direction R_d, and spatial distribution pattern R_s advocated in [Yang et al., 2017]. The relative position denotes the averaged distance d_{oi} between the center supervoxel V_o and its first-order neighbor V_i in the local context. For the relative direction, it relates to the averaged angle between the normal vector of the center supervoxel and those of its first-order neighbors in the local context. The spatial distribution pattern stands for the averaged angle of the orientation angle a_{oi} formed by the center supervoxel V_o and the first-order neighboring supervoxel V_i. The angle mentioned here is formed by the connection lines between the centers of the centering supervoxel and those of its neighbors.

To be specific, in Table 6.1, we provide the details about the entire feature vectors we designed. The roughness R is equal to the distance between the center point and the best fitting plane computed on all the points with least square.

Local context of the supervoxel

Although the supervoxel structure has already pre-clustered voxels at a lower level, supervoxels tend to oversegment objects into fragmented pieces, which results in the dissimilarity between features of different patches belonging to identical object. Hence, the decision tree may not be well trained. To tackle this problem, Wang et al. [2015c] utilize the first-order graph around a single supervoxel and generalize this graph into a local reference frame (LRF), which shows an impressive performance at car detection. To be specific, for each supervoxel, we define a local context to capture the contextual information of each supervoxel. In Fig. 6.5, we illustrate the defined local context of the supervoxel.

Detrending local tendency of supervoxel-based context

However, for the complex 3D scene interpretation, there are usually various kinds of objects to be detected and the accurate boundaries between objects are necessary to be identified in the meantime. Besides, according to the analysis conducted in [Guinard & Landrieu, 2017], for local descriptors, even for the same kind of objects, the contribution of each vector in the generated feature histogram are varying. This will result in ambiguities of the generated features for two different kinds of objects, for example, the natural ground surface and man-made ground surface. Both of these two objects have quite similar geometric characteristics (e.g., linearity, planarity, and

Table 6.1: List of totally used features

Features	Definition	Category
Linearity	$L_\lambda = \frac{e_1 - e_2}{e_1}$	
Planarity	$P_\lambda = \frac{e_2 - e_3}{e_1}$	Dimensionality features
Scattering	$S_\lambda = \frac{e_3}{e_1}$	Weinmann et al. [2015]
Omnivariance	$O_\lambda = \sqrt[3]{e_1 \cdot e_2 \cdot e_3}$	
Anisotropy	$A_\lambda = (e_1 - e_3)/e_1$	
Eigenentropy	$E_\lambda = -\sum_{i=1}^{3} e_i \ln(e_i)$	Statistical features
Local curvature	$C_\lambda = \frac{e_3}{e_1 + e_2 + e_3}$	Chehata et al. [2009]
Sum of eigenvalues	$\sum_\lambda = e_1 + e_2 + e_3$	
Height mean	$\frac{1}{n}\sum_{i=1}^{n} Z_i$	Height features
Height difference	$Z_{max} - Z_{min}$	Maas [1999]
Normal vectors	N_x N_y N_z	Orientation features Rabbani et al. [2006]
Verticality	$1 - N_z$	
Local density	$D = \frac{3n}{4\pi {r_{seed}}^3}$	Surface features
Relative position	$R_p = \frac{1}{n}\sum_{i=1,\ldots,n} d_{oi}$	
Relative direction	$R_d = \frac{1}{n}\sum_{i=1,\ldots,n} n_{oi}$	Contextual features
Distribution pattern	$R_p = \frac{1}{n}\sum_{i=1,\ldots,n} a_{oi}$	Yang et al. [2017]

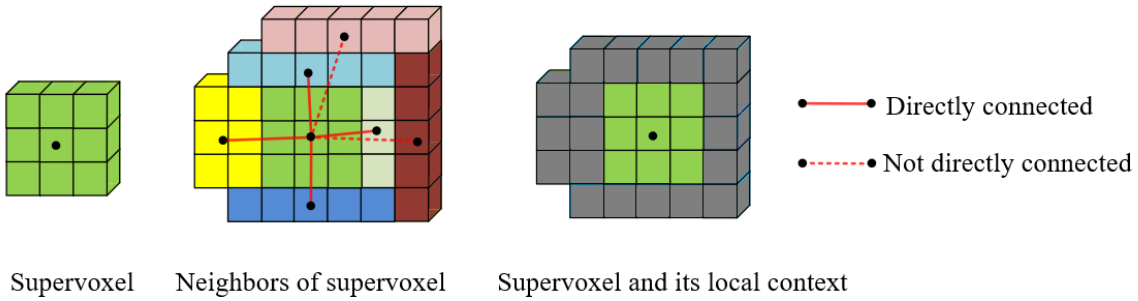


Figure 6.5: Local context of the supervoxel.

normal vectors), and the only obvious difference between them is the smoothness and roughness of their surfaces. For the achieved features histogram, we can conduct a procedure enhancing the useful features vectors with a better saliency and suppressing the trivial feature vectors.

Inspired by the Difference of Gaussian operator for edge detection in the field of image processing, we utilized the strategy given in [Sun et al., 2018] by estimating the local tendency of 3D geometry in a local context for each supervoxel, and then remove the effect of this local tendency, in order to get the salient information of the objects representing distinctive details and structures. The local tendency of the supervoxel context also plays an essential role at precisely assigning supervoxels near real boundaries of objects semantic labels. In Fig. 6.6, we show 1D profiles illustrating the estimation of the local tendency for the geometric surface of an object. It

is clear that after the removal of the local tendency, two geometric shapes with similar structures become more distinguishable.

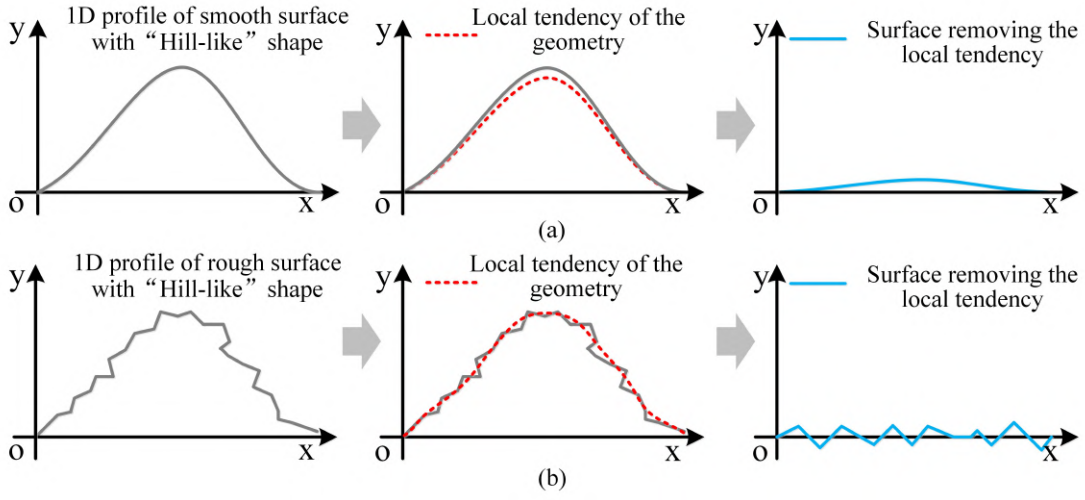


Figure 6.6: Illustration of local tendency of geometric shapes. (a) For objects with smooth surface. (b) For objects with rough surface.

This operation can also be regarded as an "high-pass" filtering which dislodges background geometric information in a local vicinity and preserve only those "high frequency" components. Considering that the eigenvalue based geometric features essentially reflect the geometric structure of the objects, namely those relatively âlow frequencyâ components, better distinctiveness can be achieved if we can combine these two kinds of components together for describing the geometry of the objects.

Detrended geometric features in the supervoxel context

For creating geometric features with local tendency detrended, we calculate dimensionaliy features, statistical features, height features, orientation features, surface features from points of the supervoxel and the local context of the supervoxel. The dimensionality, statistical, and surface features essentially reflect the 3D shape of the object, namely those relatively detailed components. In contrast, the structural, height and orientation features can provide contextual information of the object relating to fundamental components. At the meantime, contextual features encapsulate the interaction in the context. Thus, if we can combine these three kinds of components, better distinctiveness can be achieved for describing the geometry of the objects.

The removal of the local tendency of each supervoxel is achieved in the feature space. Here, the feature histogram of the supervoxel V itself is noted as H_v, while the feature histogram of the local context representing the local tendency is given by H_l, which is estimated by all the points in the local context. At last, the histogram of contextual features is noted by H_r. Thus, the detrended geometric feature histogram H_d is derived by a difference operation:

$$H_d = H_v - H_l \tag{6.2}$$

The final feature histogram H_c is defined by a weighted combination of the H_v, H_d, and H_r:

$$H_c = [H_v^T \ k \cdot {H_d}^T \ {H_r}^T] \tag{6.3}$$

Here, k stands for the weight given to the local tendency, which is estimated by the number of supervoxels in the local context. Finally, a 33 dimensional feature histogram is achieved for

supervised classification. In Fig. 6.7, an illustration of the combination of geometric features is given.

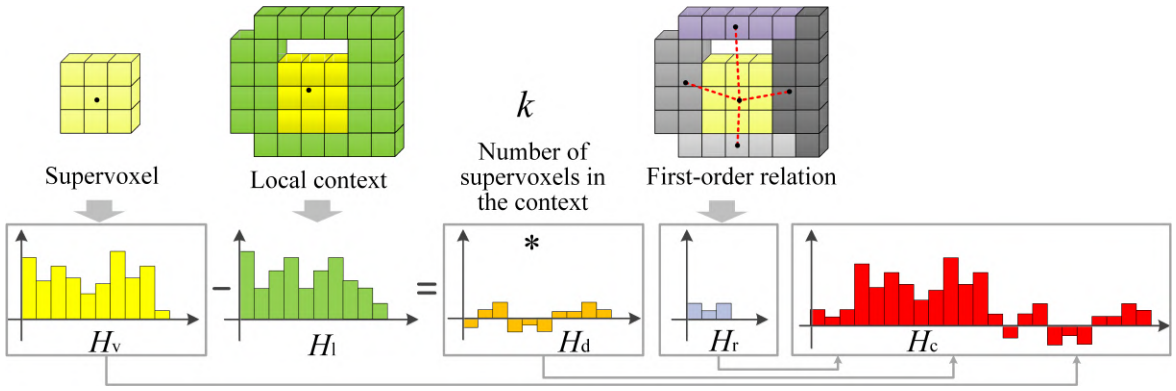


Figure 6.7: Generation of feature histogram.

It is noteworthy that compared with the method given in [Sun et al., 2018], in our method, we do not use radiometric features (e.g., RGB color or intensity), and only 3D coordinates are utilized. To some extent, our proposed detrended geometric feature use a similar strategy like the Difference of Normal (DON) feature presented in [Ioannou et al., 2012], which generate the difference of angles between normal vectors estimated from various sizes of neighborhoods. The difference is that what we used is more than normal vectors, instead, also get the difference of local geometries, height values, verticalities, and densities. Besides, we also considered the contextual features representing the interaction between the supervoxel and its context.

6.1.3 Supervised classification

Once the final geometric features of all the supervoxels in the whole point datasets are calculated, we use a supervised classification strategy with classic RF algorithm [Breiman, 2001] to discriminate supervoxel as well as the points within it with different semantic labels. The RF classifier is a combination of tree-structured classifiers which are created by a randomizing vector sampled independently from input vectors (i.e., feature histogram), and each decision tree will vote for the most likely labels to the sample of input vectors [Breiman, 2001]. Besides, the RF classifier will grow a tree at each node which makes it insensitive to overfitting problems due to the strong law of large numbers as the number of trees increases. In training, the bagging method is used for each feature combination to generate a training dataset by drawing N examples with random replacement, where N is the size of the original training set. After the supervised classification, each supervoxels V_i as well as all the points within it will be given a soft label P_i, where $P \in S$ and $S = \{p \in [0,1]^{\mathcal{K}} | \sum_{k \in \mathcal{K}} p_k = 1\}$. The probability that a supervoxel V_i belongs to the label $k \in \mathcal{K}$ is calculated by:

$$P_{i,k} = \frac{N_k}{N_t} \tag{6.4}$$

where N_k is the number of decision trees voting for class k, while N_t is the total number of decision trees.

6.1.4 Optimization based on graph structure

To reduce the misclassification from the results of random forest, we adopt a global optimization for spatial smoothing based on the adjacency graph, as advocated in [Landrieu et al., 2017]. This

optimization aims to find an improved labeling results $\hat{P}$, and the solution should provide the labeling of supervoxels with enhanced spatial smoothness and remain as close as possible to the input labeling P [Landrieu et al., 2017].

Graph structured optimization

To structure the objective functional of this optimization, we construct the adjacency graph $G = (V, E, W)$, in which the nodes represent supervoxels, and the edges encode their spatial relationship with weights W. More specifically, considering each supervoxel $V_i \in V$ as a node, all supervoxels of its KNN in Euclidean space will be connected. Then, a global graph is composed of all the connected nodes. An illustration of the generated global graph can be found in Fig. 6.8. where $\{P_1, P_2, ..., P_n\}$ is the given initial label of n supervoxels, and the weight $W(i, j) \in [0, 1]$

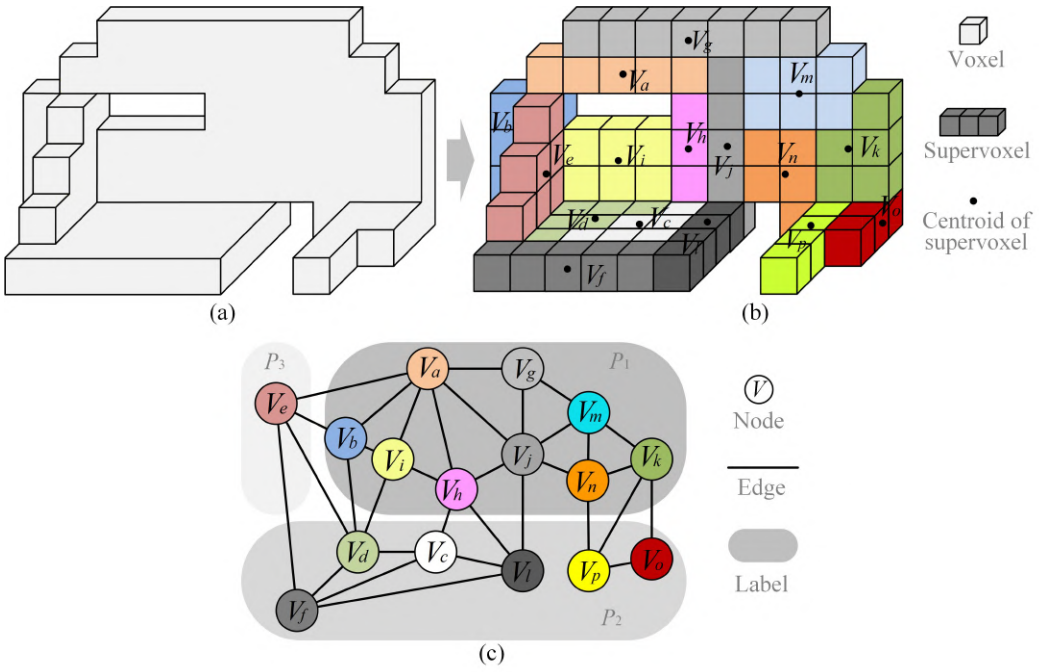


Figure 6.8: Global graph structure. (a) 3D scene. (b) Supervoxelized 3D space. (c) Generated global graph of labeled supervoxels.

of edge $e(i, j) \in E$ between supervoxels V_i and V_j is influenced by their spatial distance ΔX_{ij}, difference of normal vector angles ΔA_{ij}, and similarity ΔH_{ij}, which have been defined in Section 5.2.2:

$$W(i, j) = e^{(-\frac{\delta_x \Delta X_{ij} + \delta_a \Delta A_{ij} + \delta_h \Delta H_{ij}}{2\theta^2})} \tag{6.5}$$

where δ_x, δ_a, and δ_h are weight factors, and θ is the bandwidth of the Gaussian kernel.

With the above mentioned global graph, P^* is the solution of an optimization problem with the following structure:

$$P^* \in \arg\min_{Q \in \Omega} \sum_{i \in V} \phi(P_i, Q_i) + \sum_{(i,j) \in E} \lambda \cdot \psi(Q_i - Q_j) \tag{6.6}$$

where ϕ is the fidelity term, ψ is the regularizer, $\lambda > 0$ the regularization strength, and Ω is the search space. Here, the fidelity term $\phi(P, Q)$ enforcing the influence of the initial labeling P decreases when Q is closer to P. In contrast, the regularizer $\psi(Q_i, Q_j)$ guarantees that optimized labels of V_i and V_j are spatially smooth, which means most adjacent nodes have the same label.

The regularization strength λ balance the influence of the regularization regarding the fidelity term [Landrieu et al., 2017].

Here, the penalizer $\psi(a, b)$ influences the relation between adjacent nodes V_a and V_b, and is thus determined by Potts Model [Potts, 1952]:

$$\psi(a,b) = \begin{cases} 0 & \text{if } P_a = P_b \\ 1 & \text{if } P_a \neq P_b \end{cases} \tag{6.7}$$

where l_a and l_b are the labels of V_a and V_b. The regularization strength λ is estimated as follows:

$$\lambda = e^{-\frac{(d_{ij})^2}{\delta^2}} \tag{6.8}$$

where d_{ij} is the distance between two supervoxels, $g_{ij} \in [0, \pi]$ is the difference between angles of normal vectors N_i and N_j of two supervoxels, and δ is the expectation of all neighboring distances.

While the fidelity term $\phi(p, q)$ is a smooth and convex function, which is calculated as following a linear-logarithmic function of the observed probability, which tends to induce discrete hard labels[Landrieu et al., 2017]:

$$\phi(p,q) = -\sum_{k \in \mathcal{K}} q_k \log(\frac{\alpha}{k} + \alpha p_k) \tag{6.9}$$

where $\alpha \in [0, 1]$ and the entrywise logarithm can make the bserved probability to be smoothed to prevent numerical issues [Landrieu et al., 2017].

Solving the optimization problem

The minimization problem is solved by a Graph-Cut strategy using the alpha-expansion, which can quickly find an approximate solution with a few Graph-Cut iterations. The implementation of the alpha-expansion is achieved by the use of GCO-V3.0 library which is for optimizing multi-label energies via the alpha-expansion and alpha-beta-swap algorithms [Boykov et al., 2001; Kolmogorov & Zabih, 2004; Boykov & Kolmogorov, 2004]. Here, the labeling cost is not considered since we assume that labels of all the objects are independent so that all elements in the labeling cost matrix are set to one, except the diagonal ones seting to zero. The results of this optimization could be automatically adaptive to the underlying scenes without the predefined characteristics for some potential objects.

6.2 Modeling of planar building elements using global graph clustering and cell decomposition

Conceptually, the implementation of our proposed plane reconstruction method consists of two major phases: detection and extraction of planar segments and geometric modeling of planar segments. To be specific, the first phase can be divided into the segmentation of the point cloud and the detection of planar surfaces. For the segmentation, we propose a bottom-up point cloud segmentation method that utilizes supervoxel structure and global graph-based optimization, enabling an automatic and unsupervised segmentation of point clouds. In the subsequent step, a planarity-based extraction is conducted to segments, and only the planar segments, as well as their neighborhoods, are selected as candidates for the plane fitting. The points of the plane can be identified by the parametric model given by the planarity calculation. Afterward, the boundary points of the extracted plane are extracted by the alpha-shape. Line segments are extracted and merged by the mean-shift clustering. For the geometric modeling of planes, a

cell decomposition method is adopted to get the polygon representation of extracted planes. In Fig. 6.9, the processing workflow is sketched, with the core steps of involved methods and sample results illustrated. The detailed explanation of each step will be introduced in the following sections.

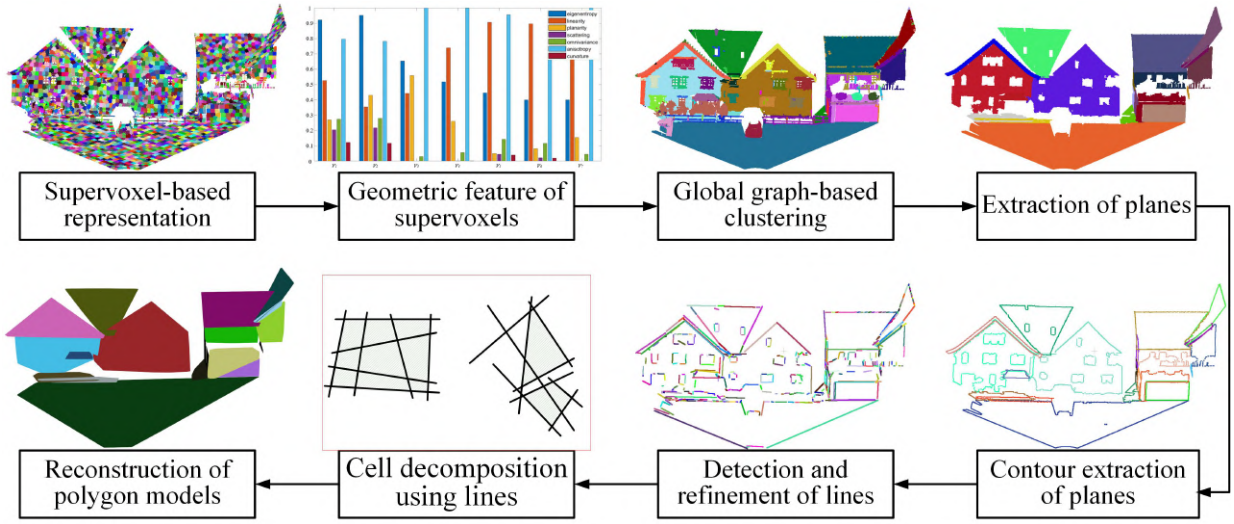


Figure 6.9: Workflow of the proposed reconstruction method.

6.2.1 Detection and extraction of planar segments

Geometric feature of supervoxles

To organize the entire point cloud into a supervoxel structure, the space is firstly divided into a small 3D cubic grid by means of octree partitioning, which splits each node into eight equal child nodes, in order to generate the octree-based voxel structure.Then, the geometric feature of each supervoxel consists of three parts: spatial position, orientation, and local geometry, which are the unary feature abstracted from the points within it. To obtain the attribute, we firstly adopt the assumption of implicit plane representation [Dutta et al., 2014] to represent the structural patch, defining an approximate plane via the normal vector $\boldsymbol{N}_i$ and centroid $\boldsymbol{X}_i$ of the point sets P_i within the patch V_i.

$$< \boldsymbol{N_i}, \boldsymbol{X_i} > -d_i^{\mathrm{c}} = 0 \tag{6.10}$$

where d_i^{c} stands for the distance from the origin to the approximate plane. The spatial position stands for spatial coordinates of the centroid $\boldsymbol{X} = (x, y, z)$ for the points set $P = \{p_1, p_2, ..., p_n\}$ inside the patch. The orientation represents the normal vector $\boldsymbol{N} = (n_x, n_y, n_z)$ of the approximated surface formed by P. While geometric features refer to four of the eigenvalue-based covariance features [Weinmann et al., 2015] encapsulating linearity L_e, planarity P_e, variation of curvature C_e, and sphericity S_e, which are calculated by the eigenvalue $e_1 \geq e_2 \geq e_3 \geq 0$, via the eigenvalue decomposition (EVD) of the 3D structure tensor, namely the covariance matrix $\boldsymbol{M} \in R^{3\times3}$ of points coordinates of P.

Construction of global graphical model

For 3D point analysis, the structure of the global graph is to represent the similarity between nodes connected with edges. In this global graphical model, the node stands for the supervoxel generated from points, while the edge connecting nodes are assigned with the weight of affinity. The structure of the graph maters the representation of the topology of the 3D scene, to simplify the graph

structure, we built the affinity graph based on the spatial connection between supervoxels, which is based on the KNN graph developed in [Funkhouser & Golovinsky, 2009]. Here, the connection between supervoxels is identified by the check of sharing boundaries.

GGBC: Global graph-based clustering

In the field of computer vision, the clustering of points is also formulated as graph construction and partitioning problems. The graphical model can explicitly represent points with a mathematical sound structure [Peng et al., 2013], utilizing context for deducing hidden information from given observations [Yao et al., 2010]. Graph-based clustering aims to divide a dataset into disjoint subsets with members similar to each other from the affinity matrix. In our previous publication [Xu et al., 2018d], we have already tested the use of the local graph structure for the description of the 3D geometry with the supervoxel structure. The use of the local graph model can make the clustering process quite efficient and available for parallel computing when combined with region-growing strategy. However, the local graph structure can merely encode the local geometry information, which can hardly represent the optimal in the global scale, so that over-segmentation frequently occurs when dealing with surfaces with irregular geometric shapes (e.g., points of vegetation). To tackle the drawbacks of local graph model, we developed the Global Graph-Based Clustering (GGBC), which constructs a global graph model to describe the local characteristics of 3D scenes with different complexities, and details of objects are preserved among the clustered nodes. By clustering the nodes V into cliques C, the supervoxels clustered in the same cliques will be merged into a single segment S of points. In Fig. 6.10, we illustrate this global graph-based clustering process.

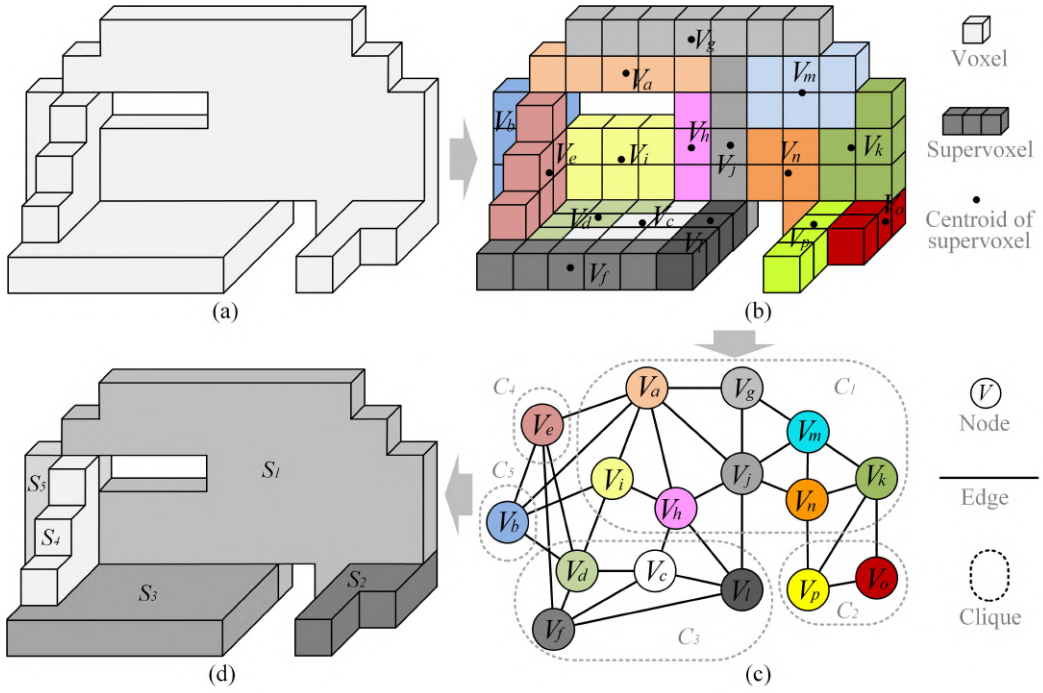


Figure 6.10: Global graph-based clustering. (a) 3D scene. (b) Supervoxelized 3D space. (c) Global graph and the clustering of cliques. (d) Generated segments.

Once the global graph of all the supervoxels is constructed, we can optimize the connection of each supervoxel by clustering nodes of the constructed global graph. To this end, similarly to work in [Xu et al., 2018d], we resolve the graph clustering problem via the adaption of the efficient

graph-based segmentation method proposed in [Felzenszwalb & Huttenlocher, 2004]. After the connections of all the voxels are identified, the connected voxels are clustered into one segment. This clustering process is performed repeatedly by traversing all the voxels with a depth-first strategy. All the connected voxels are aggregated into one segment. Additionally, a cross-validation process is required to examine the correctness of connections. In detail, for adjacent V_i and V_j, after segmenting the graph of V_i, if V_i is identified as connected to V_j, then in the segmentation of graph of V_j, V_j should be connected to V_i in turn. Otherwise, they are identified as disconnected ones.

Extraction of planes

Once the segments are obtained, for each segment, the smoothness and the curvature of the surface will be calculated by the eigenvalue $e_1 \geq e_2 \geq e_3 \geq 0$ from the EVD of the 3D structure tensor of points coordinates.

$$M_e = (e_1 - e_2)/e_1 \tag{6.11}$$

$$C_e = e_3/(e_1 + e_2 + e_3) \tag{6.12}$$

where M_e stands for the smoothness and C_e stands for the curvature. The segment with the smoothness and curvature following given thresholds are extracted as the planar segments.

The supervoxels of the planar segment will be considered as planar supervoxels, and points within these supervoxels are regarded as candidate points of the extracted plane. By the EVD calculation, the centroid and the normal vector of the segment are achieved as well, which will be used as the coefficients of the plane model. Using these coefficients as initial values, all the candidate points are examined by the RANSAC process [Schnabel et al., 2007], for estimating the optimized plane model of the planar segment. Since the initial values are approximately fitted to the plane models, the RANSAC process can find the inliers efficiently. It is noted that, for the planar supervoxels of one planar segment, the points of their neighboring supervoxels located at the outer boundary of the segment are included as the candidate points for the refinement of the extracted plane. This is designed for overcoming the "zig-zag" edges caused by the voxel-based segmentation methods [Sun et al., 2018]. The coefficients of the refined plane model will be calculated by the least square algorithm using the inliers of the RANSAC process. At last, the method of plane grouping given in Section 3.2.2.4 and Fig. 5.10 is applied to merge these neighboring planes having co-planarity.

6.2.2 Geometric modeling of planar segments

Contour extraction of planar segments

For extracted planar segments, 3D points set P_3 will be firstly projected to the 2D plane of this segment with the transformation matrix $\boldsymbol{T}$. With the alpha-shape algorithm, these projected 2D points set P_2 will provide the 2D contour B_2 of the segments. Then, the points of 3D contour B_3 of the segment can be achieved by $\boldsymbol{T}^{-1}B_2$. Here, the alpha shape algorithm [Edelsbrunner et al., 1983] has been used in determining the boundaries from points of a 2D segment especially the boundaries of convex objects. In our case, the alpha shape algorithm can reduce the redundancy of the initial linear structure which can benefit the subsequent linear extraction and refinement. For the alpha shape algorithm, an alpha value ($0 < \alpha < \infty$) is a parameter imposing the precision of the final boundary. A large value ($\alpha \to \infty$) results in the alpha boundary of a convex hull while a small value ($\alpha \to 0$) means that every point can be the boundary points. In Fig. 6.11, we illustrate the boundary points detected by the alpha shape algorithm with different alpha values.

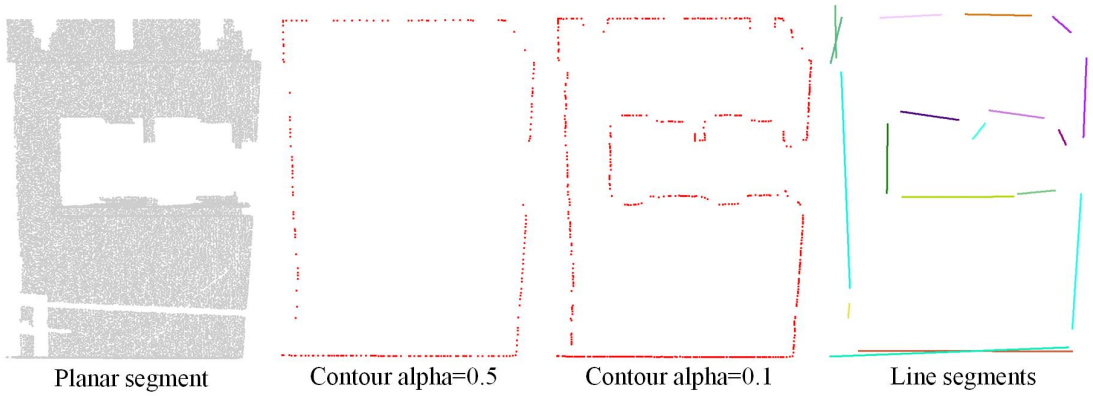


Figure 6.11: (a) Points of a planar segment. (b) and (c) Extracted contours with different alpha values.(d) Fitted line segments.

Detection of line segments

Given the points of segments contours from the previous steps, we perform the RANSAC algorithm [Fischler & Bolles, 1981] to fit the potential line segment candidates. In order to reduce the effects of outliers and fine structures such as irregular bumps and craters in the 2D floor plan, we discard the lines whose supporting points are less than a threshold of n_r. We define $L = \{l_k | k \in 1, ..., m\}$ as the detected line segment candidates from the contour points sets B_3. For each line segment l, all the neighboring line segments having similar orientation angles in a give neighbor region will be regarded as the neighboring set $N(l)$. In Fig. 6.12, we illustration of the selection of neighboring set in the neighborhood.

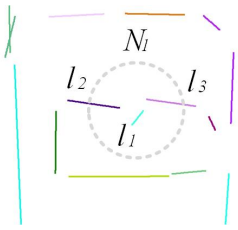


Figure 6.12: Neighboring set $N_1 = \{l_1, l_2, l_3\}$ of a line segment l_1.

Refinement of line segments

To eliminate the redundant line segment candidates and obtain the real and concise line segment representation, we further refine the orientations of fitted line segments, so that the refined line segments can be merged into complete lines with smooth connections. Similar to the approaches [Poullis, 2013], we refine the detected line segment using a regularization of orientations. The first step is to determine the orientation of the fragments of line segments. Then, in the second step, a classification (i.e., a labeling task) of these orientations will be conducted, which can be formulated as MRF problem and solved by Graph Cuts algorithm [Kolmogorov & Zabih, 2004]. To be specific, for each line segment, the orientation θ_p is directly achieved by direction parameters of its line model. Here, we make an assumption that line segments constructing the same polygon may only have a limited number of orientation angles (i.e., labels in the energy function). In other words, edges are encouraged to be parallel or perpendicular with longer ones [Xie et al., 2017]. Moreover, we also assume that the refined angles should not have large deviation from its initial angles.

Based on these two assumptions, we first define an set of orientation angles Φ for all the line segments. Then each line segment l is augmented by forming candidate line segments set L using the center of the origin line and the orientation angles in Φ. After that, an energy function considering both neighboring smoothness and degrees of orientations is built to help line segments to determine proper orientation angles. The energy function is shown as the follows:

$$E = \sum_{p \in P, \theta \in \Phi} \lambda \cdot D(p, \theta) + \sum_{(p,q) \in N} S(\theta_p, \theta_q) \tag{6.13}$$

In the energy function, the data term reflecting the data fitting residuals. $D(p, \theta)$ denotes how well a line segment is fitting to the estimated orientation angle, which is illustrated as following:

$$D(p, \theta) = \sum_{p_i} d_{\perp}(p_i, \theta) \tag{6.14}$$

Where, i is the point in line segment p, while $d_{\perp}(p_i, \theta)$ stands for the perpendicular distance from point i to candidate line segment p with orientation angle θ. In the smooth term, which penalize line segments with similar initial angles being labelled differently, N is line segments set in the orientation neighbour region which includes pairs of line segments that have similar orientation angles. λ is the scale factor that balance these two terms. While θ is the initial orientation angle of a given line segment. δ is the normalization angle value. Here, the initial orientation angle is augmented by including the perpendicular one.

$$S(\theta_p, \theta_q) = e^{\frac{-|\theta_p - \theta_q|}{\delta^2}} \tag{6.15}$$

After the minimization of the energy function with Graph-Cut, the labeling result are translated to the corresponding orientation angles. Those line segments with the same orientation angles are merged to form a new one and corners are the intersection of two unparalleled or perpendicular line segments.

Representation of models

To represent the model of the reconstructed planar element, we use a combination of the boundary representation and the surface representation. The boundary representation is given by the polygon resulting from the cell decomposition which is to approximate the original outline of line segments with closed polygons, and it eases the task of determining and assembling a complex structure from parameterized, standard shapes [Kada, 2007]. In Fig. 2.6, we illustrate the process of using this approach to get the boundary representation of a 2D object. The key step of cell decomposition is the 2D arrangement algorithm applied to generate cells $C = \{c_1, c_2, ..., c_n\}$ from intersecting line segments. Once we get the cells represented with polygons, we will downdsample the points of the planar object, and re-project the downsampled points of the planar object to its corresponding polygon. Then, if a cell contains sufficient number of re-projected points, this cells will be regarded as occupied one belonging to the surface of the planar object. By occupied analysis, cells belong to the objects C^* and the cells of the background $C^{'} = C \backslash C^*$ are distinguished. Here, the boundaries of occupied cells in C^* will be selected. Outlines of these cells are extracted and connected to form the boundary representation of the object. The surface representation is achieved by triangle meshes. The re-projected points and the vertices and edges of the polygons are used as vertices of triangles. The use of the meshes can enrich the details of the reconstructed model, while the use of the boundary representation can constraint the smoothness of the edges for the model.

7 Experiments

In this Chapter, the datasets used in the experiments are introduced, and the quality of the datasets will be analyzed and discussed as well. Besides, we will also dedicate the evaluation metric that will be used in the result and discussion parts.

7.1 Testing data and experiments

Both synthetic datasets, photogrammetric point cloud, and LiDAR point cloud are tested in various experiments. The synthetic dataset aims at evaluating the proposed LSSHOT descriptor in the shape matching tests, while the photogrammetric point cloud is utilized to verify our proposed methods for practical applications of the reconstruction and segmentation. Besides, introduce the LiDAR point cloud for the segmentation, semantic labeling, and object reconstruction experiment.

7.1.1 Synthetic datasets: Shape matching tests

For testing the performance of our proposed LSSHOT shape descriptor, we conducted the shape matching test similar to the work of [Guo et al., 2015]. The synthetic dataset is mainly a point cloud consisting of artificial structures including four kinds of basic geometric shapes and is used as a matching scene for the shape matching tests. In contrast to this matching scene, three additional artificial objects (i.e., the cylinders, boards, and triangular prisms) are used as matching objects in these tests. Fig. 7.1 illustrates the matching objects and matching scene used in this test.

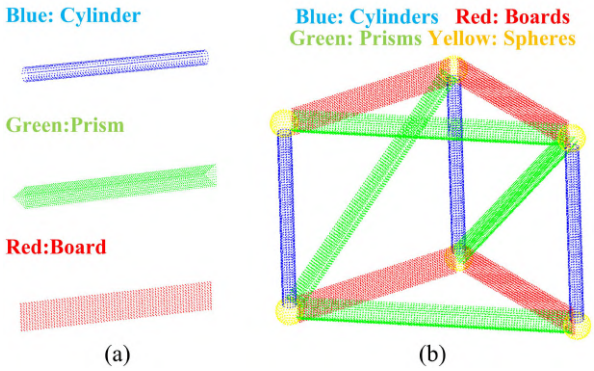


Figure 7.1: Simulated point clouds of (a) matching objects and (b) matching scene.

To assess the robustness of the descriptor, two kinds of noises are added to the synthetic point clouds. The first kind is the matching error noise (ME) mimicking the uncertainties of photogrammetric points obtained by the stereo matching process. The second kind is the background noise (BG) representing outliers in the scene being associated with distractions and mismatches. These noises are created according to different noise models. As the ME noise simulates the matching

errors of points, it is generated by changing spatial positions of points in the original cloud with its amplitude following a Gaussian distribution. This assumption about the noise model has also been used in the aforementioned statistical filtering [Rusu & Cousins, 2011]. The variation of the ME noise amplitude is equal to the voxel size used in the downsampling process, and the mean value is set to be zero. By contrast, the BG noise is assumed as additional points randomly distributed in the range of the whole scene. It means that the position of points representing BG noise is randomly chosen in the simulated scene because we can hardly find an appropriate noise model for the outliers. In the following matching tests, the points of ME noise are regarded as inliers being part of the objects, whereas the points of BG noise are deemed outliers belonging to the background scene rather than objects. In Fig. 7.2, a comparison between the original point cloud (Fig. 7.2a) of a tube, the same point cloud adding ME noise (Fig. 7.2b) and the same point cloud with both ME noise and BG noise (Fig. 7.2c) is shown. In this case, the percentage of noise added to the original point cloud is 30%.

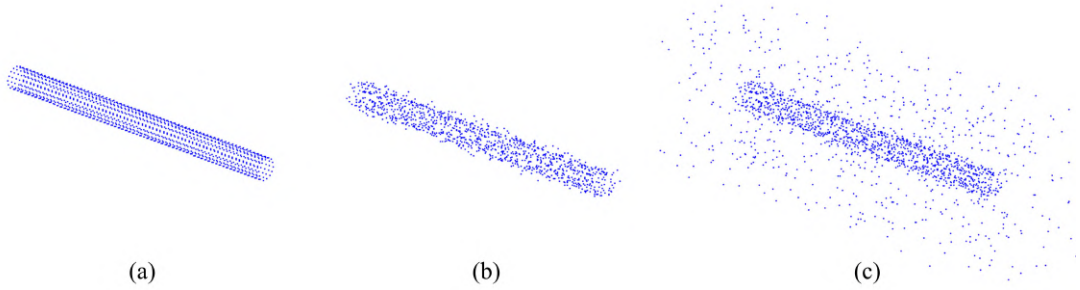


Figure 7.2: (a) Original point cloud. (b) With ME noise.(c) With both ME and BG noise.

7.1.2 Photogrammetric point cloud: Recognition and reconstruction tests

For testing our proposed detection and reconstruction approaches, a construction site in Munich, Germany (Fig. 7.3a) is chosen as the experimental site, with an area on the ground of 2300 m^2. The as-built building in the site consists of three main facades being triangular. The photogrammetric point cloud is generated from a Structure from Motion (SfM) system and multi-view stereo matching method as described in [Tuttas et al., 2014], in which the VSfM Software [Wu, 2013] and LibTSgm [Rothermel et al., 2012; Hirschmuller, 2008] act as SfM and dense matching tools correspondingly. There are in total of 81 images with the size of 4256*2820 pixels used and 33 million points generated. The Z-axis in the coordinate system of the point data is perpendicular to the earth ground plane. It can be seen from Fig. 7.3c that the point cloud contains a lot of noise and clutter, especially the parts near the building surfaces, and the points seem to be sampled with very uneven densities. In this study, the average distance between the inner scaffold row and the building surface is approximately 0.3 m to 0.6 m, while the distance between the outer row and the scaffold is about 1.1m to 1.4 m.

7.1.3 TLS: Recognition, segmentation, and reconstruction tests

The TLS point clouds are from the large-scale point cloud classification benchmark dataset published on www.semantic3d.net by ETH Zurich [Hackel et al., 2017], which covers a wide variety of diverse building scenes like churches, streets, squares, villages, and castles. Specifically, two-point clouds of different scenes are tested (see Fig. 7.4): one is scanned in the area of the cathedral of St. Gallen, whereas the other one is measured in the area of a town square. A clipping process is also conducted to remove the irrelevant and distant parts in the point cloud of the scene. For the original point cloud shown in Fig. 7.4, the color represents the intensity of laser reflections,

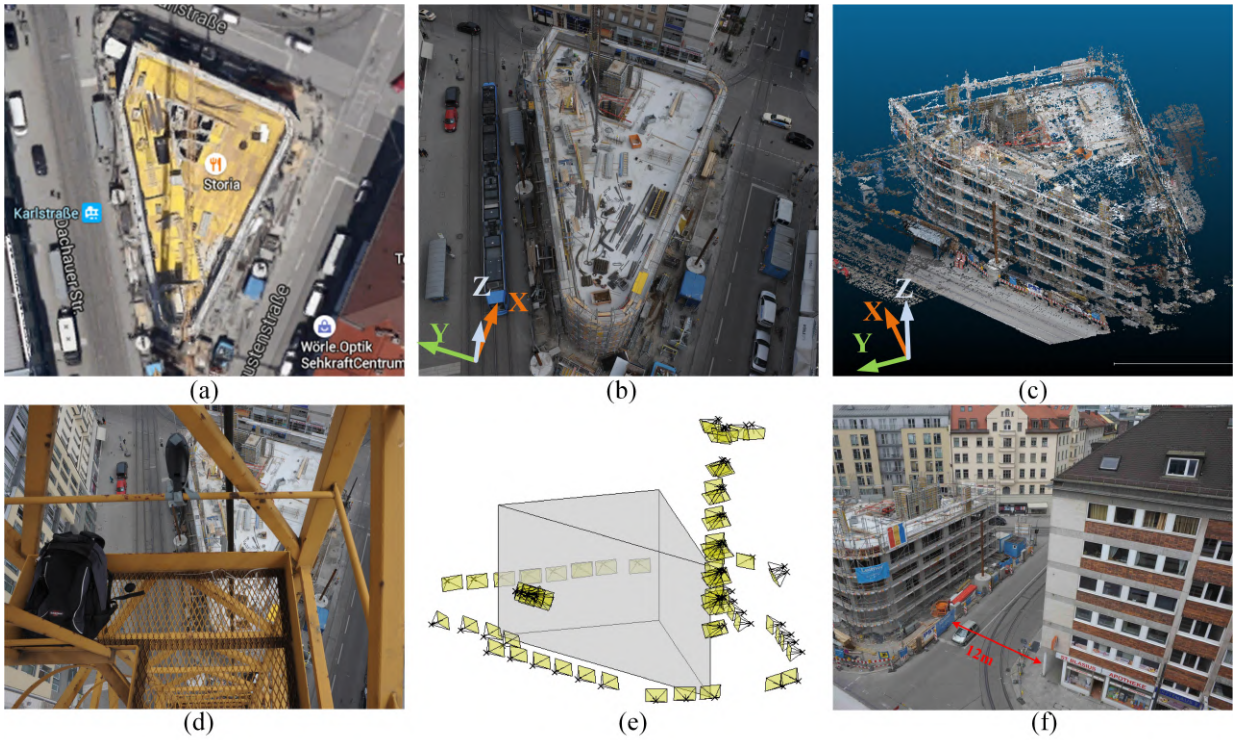


Figure 7.3: Photogrammetric point clouds. (a) Satellite image of the construction site. (b) Image of the construction site taken from the crane. (c) Dense point cloud from images.(d) Situation for taking images from the crane. (e) Image configuration (Figure courtesy of [Tuttas et al., 2017]). (f) Distance between the construction site and neighboring buildings.

with brighter color showing stronger intensities. In our current work, the intensity of the point is not involved. Noise and outliers are kept in the datasets.

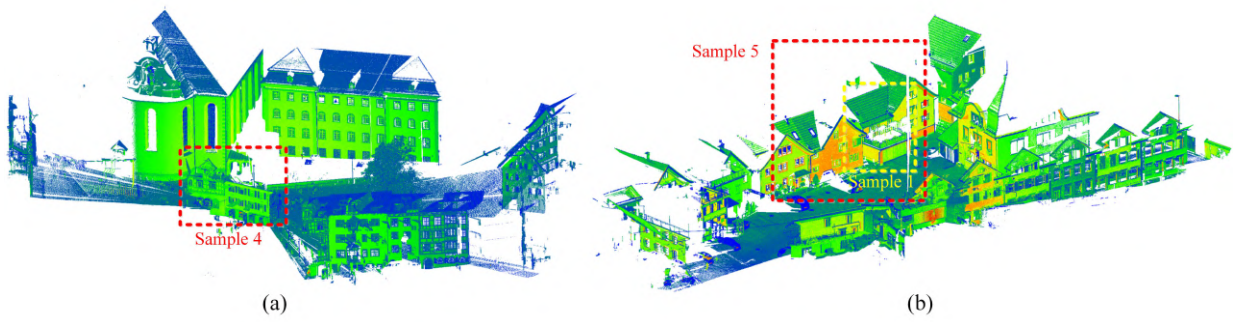


Figure 7.4: Experimental TLS point clouds of building scenes. (a) St.Gallen Cathedral. (b) Town square.

Besides, we also use the point clouds from the scene of a construction site (see Fig. 7.5) located in the downtown area of Munich, Germany, with both laser scanning and photogrammetric point clouds (see Figs. 7.5b and 7.5c) acquired. Its testing area is around 320m^2, including foundation pits, ground objects, wall surface, equipment, etc. The terrestrial LiDAR point cloud is scanned via Leica HDS 7000, whereas the photogrammetric point cloud is created from an SfM system and multi-view stereo matching method [Tuttas et al., 2017], using a Nikon D3 DLSR camera with 105 images. In Fig. 7.6, we provide an illustration of the configuration of images for generating the photogrammetric point cloud of our test scene, involving 43 images. Scanning positions of

the laser scanner are also given in Fig. 7.6. Moreover, before the major processing, statistical outlier removal filtering [Rusu & Cousins, 2011] was applied to these point clouds. The LiDAR and photogrammetric point clouds are both downsampled to around nine million points.

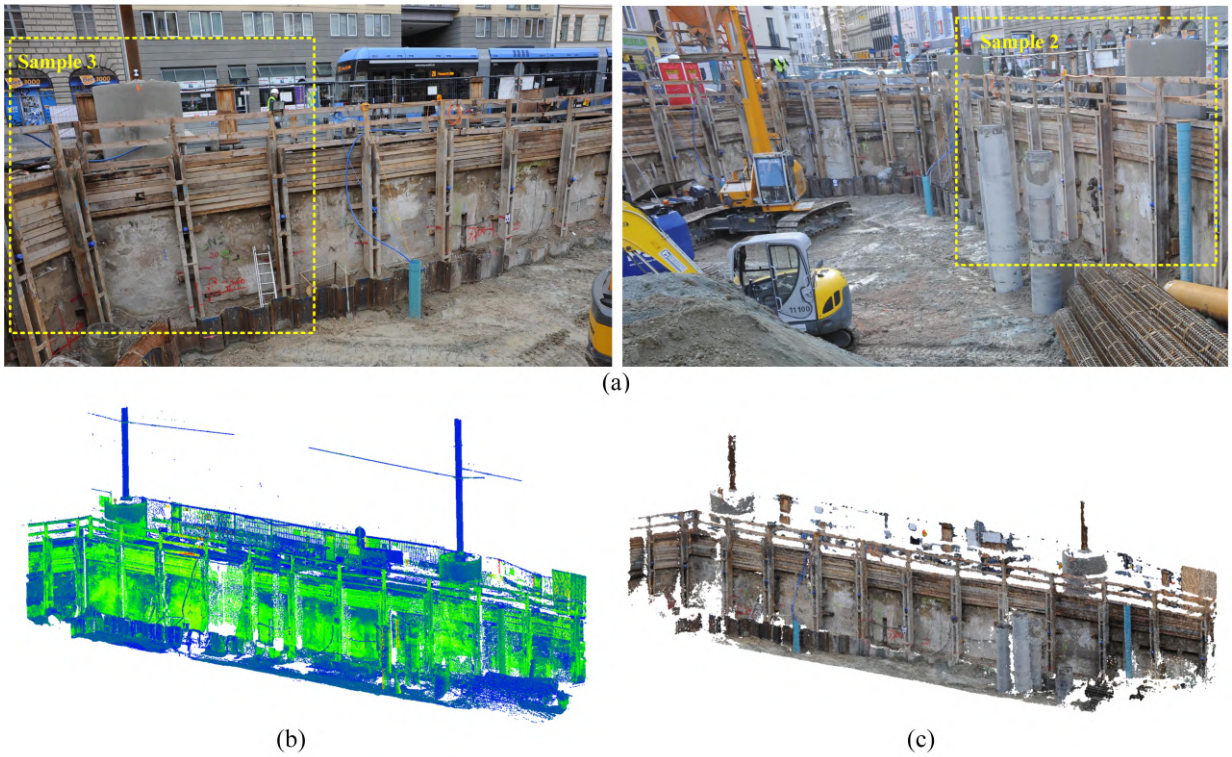


Figure 7.5: Experimental TLS and photogrammetric point clouds of a construction site. (a) Optical images of the construction site scene. (b) Photogrammetric and (c) LiDAR point clouds of the construction site scene.

Table 7.1: Information of testing point clouds.

	St. Gallen	Town square
Number of points	14066783	25183032
Area of scenes	$\approx 4000m^2$	$\approx 2100m^2$

Three sample areas (i.e., Sample 1, Sample 2, and Sample 3) are selected from two scenes and manually segmented as ground truth (see areas in dash boxes of Figs. 7.4b and 7.5a). Note that for the scene of the construction site, ground truth is sampled from both laser scanning and photogrammetric point clouds. The manual segmentation follows the rule that each segment should correspond to a semantic object of building components. For instance, planar segments represent wall or ground surfaces in the scene, while linear segments are related to frames of windows in the facade. The numbers of segments in ground truth datasets are listed in Table 7.2.

Especially for the evaluation of segmentation performance, similar to the work in [Mahmoudabadi et al., 2016], the reference datasets we used as ground truth are produced manually. Two sample areas (i.e., Sample 4 and Sample 5) are selected from the experimental datasets as the reference (see Fig. 7.5). The rules for manual segmentation is fixed. Namely, each segment should correspond to a semantic object of the building component. For example, a planar segment

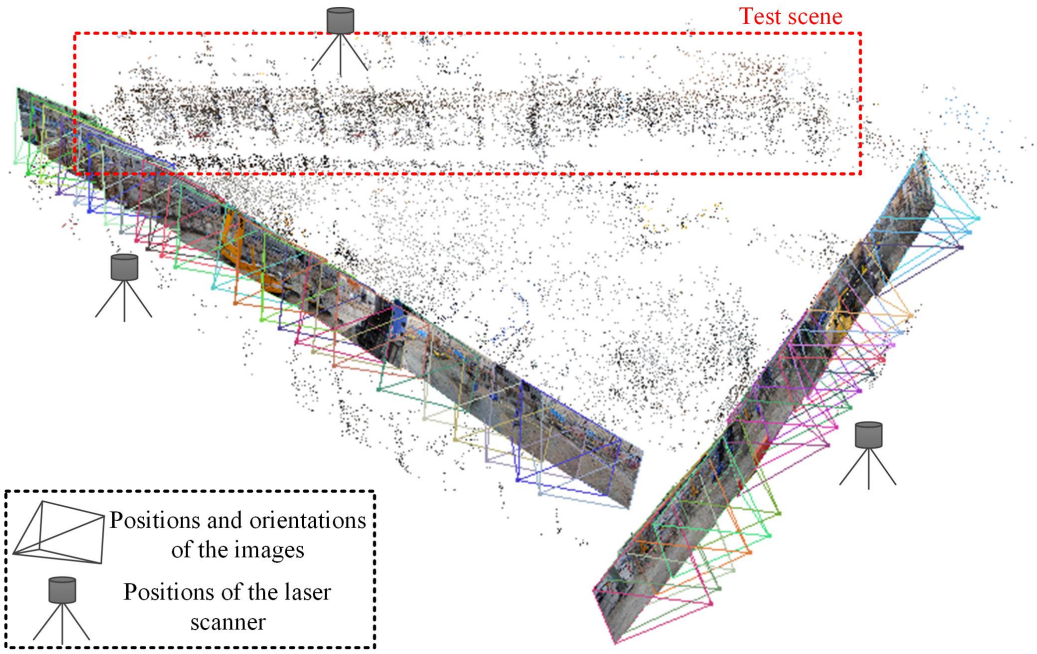


Figure 7.6: Geometric configuration of images for generating the photogrammetric point cloud and positions of the laser scanner for acquiring the LiDAR point cloud.

Table 7.2: Number of segments in ground truth datasets.

Ground truth	Number of segments	
	Laser scanned	Photogrammetric
Sample 1	33	—-
Sample 2	66	37
Sample 3	74	50

representing the northern wall of the building, the frame structure representing to one of the windows in the facade, and the curved surface standing for one of the roof facets of the building. Furthermore, to avoid personal preferences when manually segmenting the dataset, we utilize the strategy propose in [Vo et al., 2015], namely each reference datasets are segmented independently by persons who are familiar with point cloud segmentation work. Then, automatic segmentation results will be compared against two reference datasets individually. Manually segmented reference datasets are shown in Fig.7.8. For the reference datasets 1, there are in total 101 and 66 segments obtained for the scenes of St. Gallen cathedral and Townsquare, respectively. Whereas for the reference datasets 2, there are in total 100 and 84 segments obtained for the scenes of St. Gallen cathedral and Townsquare, respectively.

7.1.4 MLS and TLS: Classification and reconstruction tests

The MLS dataset is adopted in the semantic labeling experiments. Here, the testing area is the Arcisstrasse along the main entrance of Technical University of Munich (TUM) city campus, which covers about an area of around 29000 m^2 and has been already displayed in Fig. 7.10a. Fraunhofer Institute of Optronics originally acquires this dataset, System Technologies and Image Exploitation (IOSB) [Gehrung et al., 2017]. Two Velodyne HDL-64E acquires the used point

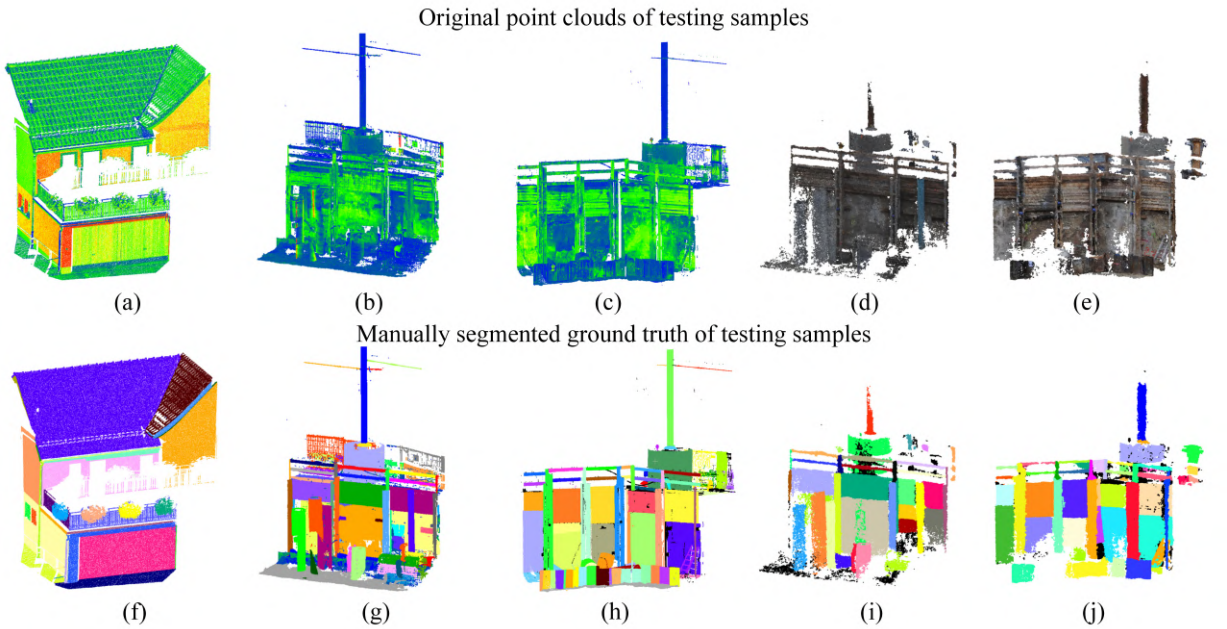


Figure 7.7: Testing samples for segmentation evaluation. (a)-(e) Original point clouds. (f)-(j) Corresponding manually segmented ground truth (rendered with different gray colors).

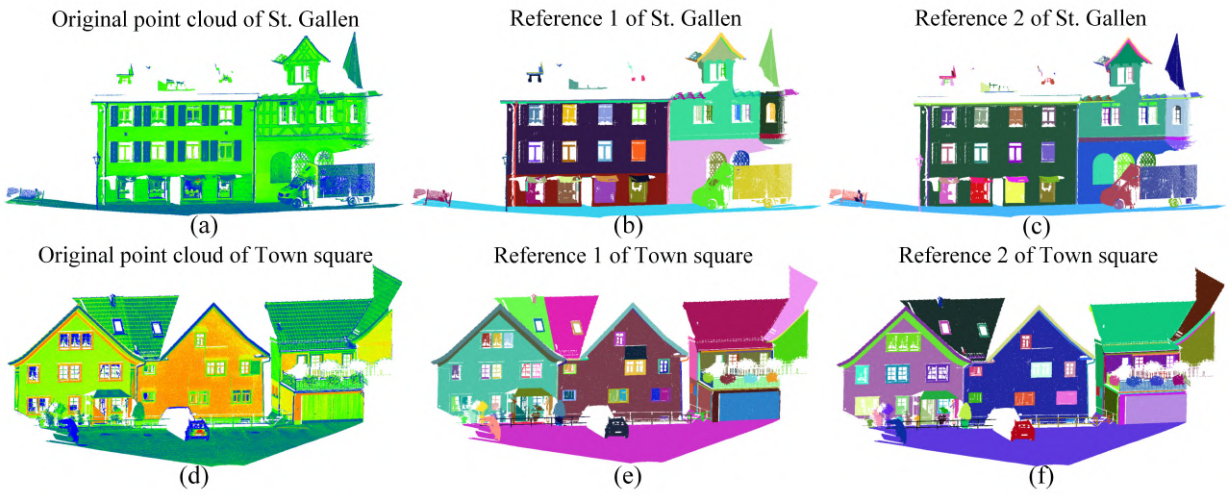
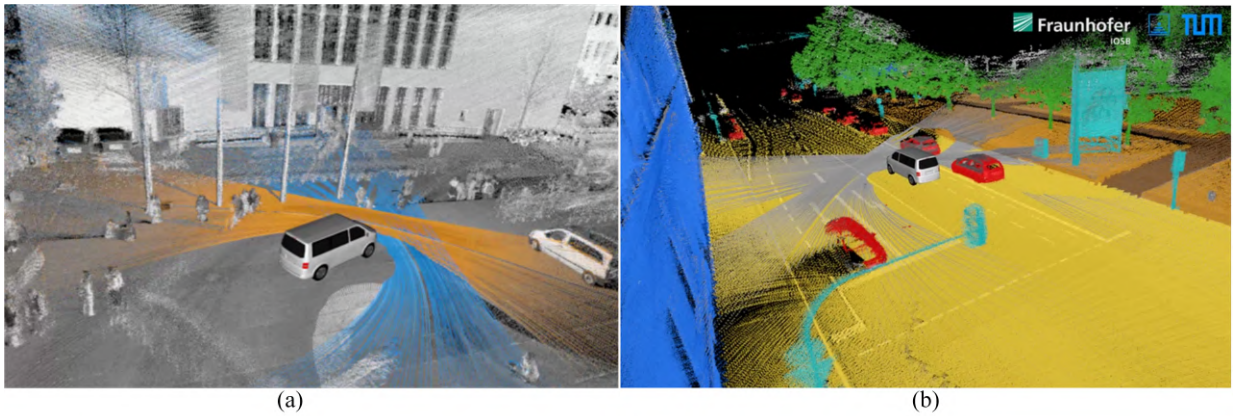


Figure 7.8: Reference set for segmentation evaluation. (a) Original point cloud, (b) Manually segmented reference 1 and (c) Manually segmented reference 2 of Cathedral of St. Gallen. (d) Original point cloud, (e) Manually segmented reference 1 and (f) Manually segmented reference 2 of Town square.

clouds mounted at an angle of 35° on the front roof of the vehicle. Fig. 7.9 provides sketch about how the two scanners are mounted [Gehrung et al., 2017].

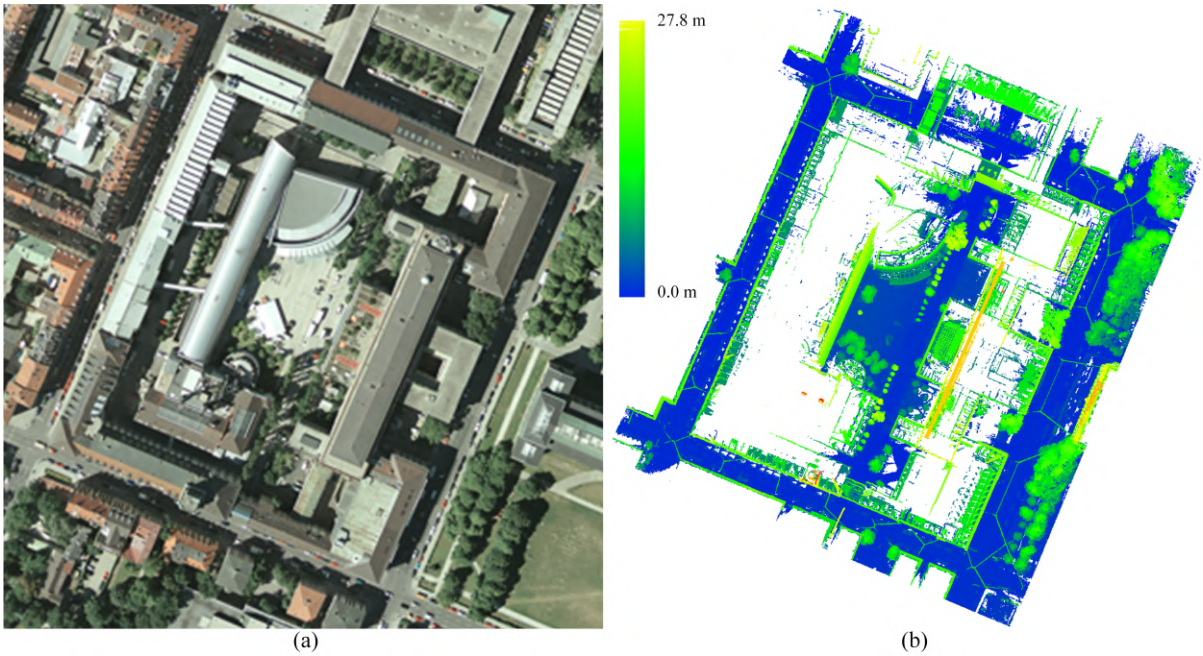
With thousands of scans acquired by the laser scanners along the Arcisstrasse, the scene contains various kinds of objects according to the eight semantic classes [Hackel et al., 2016b]. For the evaluation process, an accurate manually labeled point cloud for the experimental dataset as ground truth is also generated. The manual work is conducted following the ETH standard (Semantic3D Benchmark) and consumed 30 hours. As a consequence, a highly accurate reference of the entire city campus is generated. In Fig. 7.11, the labeled scene is rendered by eight different



(a) (b)

Figure 7.9: Two oblique mounted laser scanners of the MLS system. (a) Figure courtesy of [*Gehrunget al.*, 2017]. (b) One application of using the manually labeled ground truth data.

colors, including building, hight vegetation, low vegetation, vehicles, human-made terrain, natural terrain, hardscape, and scanning artifacts.



(a) (b)

Figure 7.10: Experimental MLS point clouds. (a) Aerial image of TUM city campus and (b) MLS point clouds.

The TLS dataset is used to further test our classification method on the point clouds with varying point density. Here, we used the popular Semantic 3D dataset published by ETH Zürich [Hackel et al., 2016b] for a preliminary validation. This dataset provide manually labeled into eight different classes, namely building, hight vegetation, low vegetation, vehicles, human-made terrain, natural terrain, hardscape, and scanning artifacts. In this experiment, the scans we used are Bildstein and Untermaederbrunnen. Each scene has three scans, we use two of them as training dataset and the rest one as test dataset. In Fig. 7.12, we illustrate the manually labeled reference of these two scenes, with figures courtesy of www.semantic3d.net.

7.2 Quality of point clouds

Before the use of point clouds, the points density and possible outliers of the testing point clouds are analyzed, in order to provide a detailed view of the quality of datasets. For computing the points density, the local points density (LPD) index of the test point cloud is estimated using Eq. 7.1 [Vo et al., 2015]. As for the analysis of possible outliers existing in the point cloud, the mean distances between k-nearest neighbors (MDK) are computed using Eq. 7.2, the origin of which is the statistical outlier removal filter [Rusu & Cousins, 2011]. The equation of calculating LPD and MDK are given as follows.

$$\mathrm{LPD} = \frac{k}{\pi \cdot d_k^2} \tag{7.1}$$

$$\mathrm{MDK} = \frac{1}{k} \cdot \sum_{i=1}^{k} d_i \tag{7.2}$$

where k is the number of neighbors defined by the user, d_i and d_k denotes the distances from the point of interest to the k th neighbor and the furthest neighbor, respectively.

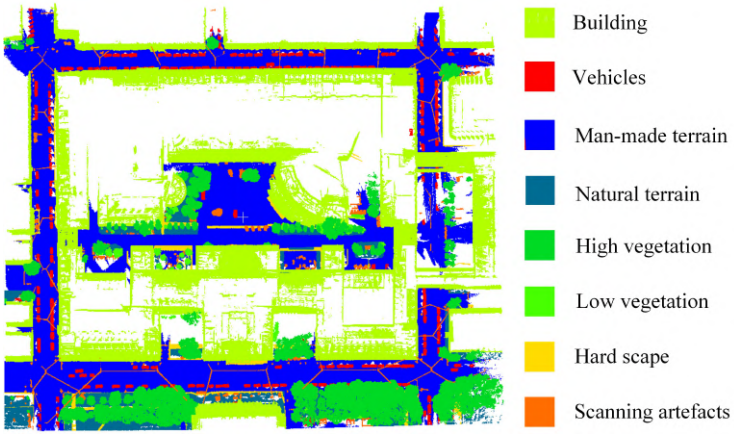


Figure 7.11: Manually labeled dataset of TUM city campus.

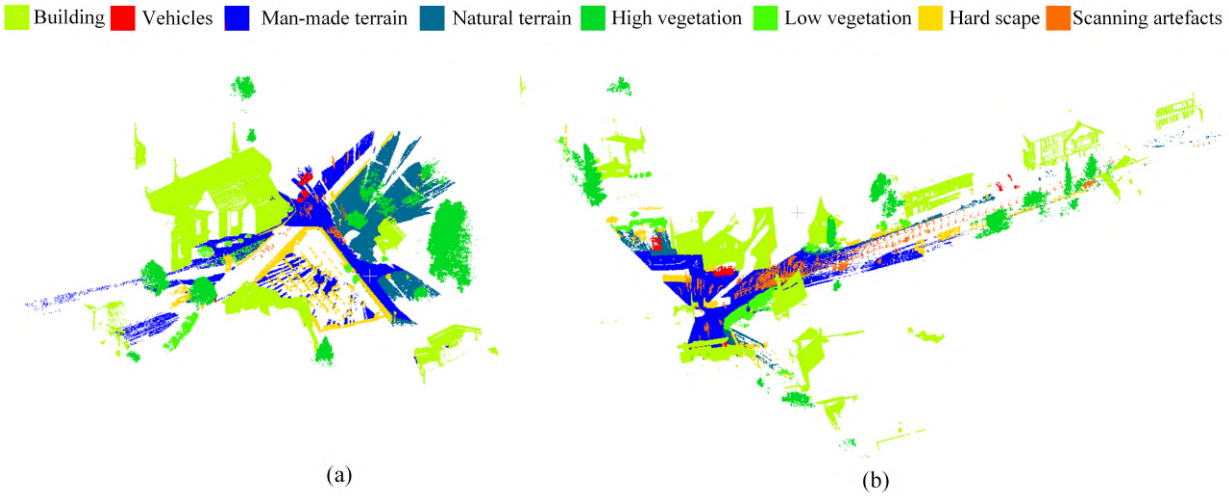


Figure 7.12: Manually labeled ground truth data. (a) Labeled scene of Bildstein and (b) labeled scene of Untermaederbrunnen.

7.3 Evaluation metric

7.3.1 Evaluation metric of the shape descriptor

In the shape matching tests, the artificial point clouds of three geometric objects (i.e., board, cylinder, and prism) are matched to the point cloud of the matching scene, by the use of features calculated via LSSHOT and SHOT descriptor. Here, the renown SHOT descriptor is used as the reference for evaluation. If the distance between the features of two points in feature space is smaller than a given threshold τ_m, they are considered as corresponding points. Features of points from one matching object are compared with those of points from the point cloud of the matching scene. It is worth pointing that in theory, the more significant the τ_m is, the more points will be matched, and at the same time, the possibility of false matches is increased. Both the numbers of correct and false matched points are recorded. By the use of these two numbers, the recall ratio R_r and 1-precision ratio R_{1-p} are calculated as the criterion, as proposed in [Salti et al., 2014]. In addition, to assess the robustness of LSSHOT to noise, the matching tests are conducted with various levels of noises added. The noise percentage of a point cloud represents the percentage of points associated with ME and BG noises in this cloud, respectively. For instance, a noise level of 50% indicates that in the given point cloud the ME noise points occupied half of the points belonging to "clean" objects while the number of BG noise points amounts to half of the points belonging to "clean" objects as well. The noise levels used throughout these tests range from 0% to 90% with an interval of 10%. As shown in Eq. 7.3, the recall ratio is calculated by the number of correctly matched points and the number of points correctly corresponding to the object, whereas the is calculated with the number of false matched points as shown in Eq. 7.4.

$$R_r = \frac{N_m}{N_l} \tag{7.3}$$

$$R_{1-p} = \frac{N_w}{N_m + N_w} \tag{7.4}$$

As described in [Mikolajczyk & Schmid, 2005], a good descriptor should have a high recall ratio for any precision ratio. Thus, the recall versus 1-precision curve and the recall versus noise level curve is presented as the results of matching tests for evaluation.

7.3.2 Evaluation metric of segmentation

The quantitative performance of our proposed method is assessed by the agreement between the segmentation result and the ground-truth dataset (i.e., the reference dataset). The results of the segmentation using our proposed method are compared against the manually segmented ground-truth dataset in a point to point way. The information retrieval measurements, including *precision*, *recall*, and F_1 score, are selected as basic measures for assessing the effectiveness and accuracy of our method. *precision* stands for the percentage of correctly retrieved elements (i.e., correctly segmented points in the segmentation results), whereas *recall* corresponds to the percentage of reference dataset that is correctly retrieved (i.e., correctly segmented points in the reference data).

Indeed, for unsupervised clustering methods, the abovementioned criteria like internal index will be a good choice. Nevertheless, for the segmentation task, what we care more about is not the clustering performance. Instead, we care more about the correctness of the boundaries of each segment. However, according to the ground truth dataset we have, it is tough to get the accurate positions of points belonging to boundaries. Therefore, as an alternative, in the evaluation process, we adopt the strategy utilized by [Awrangjeb & Fraser, 2014; Vo et al., 2015; Nurunnabi et al.,

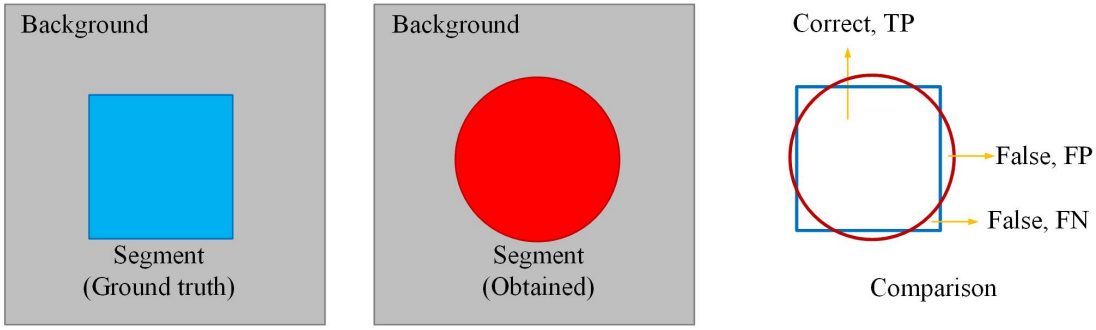


Figure 7.13: Comparison of corresponding segments.

2015], which was originally introduced for the plane detection problem. According to this strategy, a pair of segments (one is from obtained segmentation results, and the other one is from the ground truth dataset) is found and considered as having correspondence. Once a correspondence was found, the number of true positive (TP), false positive (FP), and false negative (FN) point computed from this pair are checked and used to calculate the *precision*, *recall*, and $F1$ measure. The F_1 score is introduced to balance the P_r and R_e, as an overall measurement of effectiveness. These three measures are computed as per Eq. 7.5- Eq. 7.7, using the true positive (TP), true negative (TN), false positive (FP), false negative (FN) from the confusion matrix. In Fig. 7.13, we illustrate how the TP, FP, and FN are defined for comparing a correspondence. With these measures, we can easily determine whether a segment is correctly obtained or not, according to statistical values instead of sophisticatedly comparing geometric boundaries.

$$P_r = \frac{TP}{TP + FP} \tag{7.5}$$

$$R_e = \frac{TP}{TP + FN} \tag{7.6}$$

$$F_1 = \frac{2TP}{2TP + FP + FN} \tag{7.7}$$

7.3.3 Evaluation metric of classification

For the evaluation of the classification, We follow the Pascal VOC challenges [Everingham et al., 2010] and use Intersection over Union (IoU) averaged over all classes. The evaluation measure for class i is defined as

$$IoU_i = \frac{TP_i}{TP_i + FP_i + FN_i} \tag{7.8}$$

The main evaluation measure of our benchmark is thus

$$\overline{IoU} = \frac{1}{N} \sum_{i=1}^{N} IoU_i \tag{7.9}$$

We also report IoU_i for each class i and overall accuracy

$$OA = \sum_{i=1}^{N} \left(\frac{TP_i}{TP_i + TN_i + FP_i + FN_i} \right) \tag{7.10}$$

7.3.4 Evaluation metric of modeling

Reconstruction results are evaluated in terms of the geometric correctness and completeness. The geometric correctness is related to the accuracy of their geometric representations, whereas completeness denotes the number of correctly reconstructed components of one facade. Since the scaffolding elements are not represented as ground truth in the as-planned BIM, the reconstructed objects are back-projected to the original point cloud and compared to the parameters obtained from manually measured points.

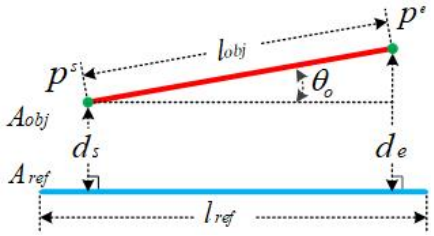


Figure 7.14: Comparison between the main axes of the reconstructed and reference elements.

To be specific, the accuracy of geometric representation considers the differences between the positions, orientations, and sizes of the reference and reconstructed objects, while the numbers of reconstructed elements in each facade are counted in light of the color information of points. The reference positions and orientations are measured from the original point cloud according to the color information, and the reference sizes use the standard sizes of scaffolding elements. It is necessary here to clarify that one reconstructed object in our result often covers several scaffolding elements during the merging refinement. It means that the scaffolding elements covered by one reconstructed object will share a common geometric representation. In one facade, if our reconstructed objects cover a scaffolding element correctly, the element will be regarded a reconstructed one, and its correctness will be evaluated by using the parameters of geometric representation (e.g., position) of its corresponding reconstructed objects. To compare the positions and orientations, we employ the main axes of the linear objects to represent the elements, so that the comparison is simplified to the measurement of distances and angles. As seen from the Fig. 7.14, A_{obj} and A_{ref} represent the main axes of the reconstructed and reference elements, respectively. d_s and d_e denote the distances between both endpoints of l_{obj} and the axis of the reference l_{ref}. Their mean value is used to assess the accuracy of positions. The angles θ_o between the main axes of reconstructed elements and reference ones are calculated for the evaluation of orientations. Since the radius and thickness have already been used as constraints in the modeling process, as for the assessment of sizes, we compare the length of the main axes l_{obj} and l_{ref}, in order to approximate the accuracy of sizes efficiently. The ratio between l_{obj} and l_{ref} is used as the criteria.

8 Results and discussion

This chapter covers the qualitative and quantitative experimental results of the various algorithms and methods listed in Chapters 3-6. Based on the numerical results, the advantages and disadvantages of those algorithms and methods presented in this thesis are discussed and compared with baseline and competition methods.

8.1 Recognition of structural elements in construction sites

In the following experiments, all the methods are implemented via C++ and run on an Intel i7-4710MQ CPU @ 2.5GHz and with 16.0 GB RAM. All the baseline methods are implemented as part of PCL 1.8.0 [Rusu & Cousins, 2011].

To evaluate the performance of our methods, four representative segmentation algorithms, including the Euclidean distance and DON based clustering[Ioannou et al., 2012], the smoothness based region growing [Rabbani et al., 2006], and the Local Convex Connected Patches (LCCP) [Stein et al., 2014] are used as baseline methods. Here, the RG and DON methods are famous and widely used point-based segmentation algorithms, while the LCCP method is a popular supervoxel-based segmentation method, which also adopts the voxel structure and uses the convexity as the segmentation criteria.

8.1.1 Results of the building facade scene

Segmentation results of VGS and SVGS using the LiDAR point cloud from the building facade scene are illustrated in Figs.8.1 a and 8.1b, with segments rendered using different gray values. Here, the voxel resolution used in VGS and SVGS is set to 0.1m, while the seed resolution of supervoxels used in SVGS is 0.2m. The thresholds δ for graph segmentation are optionally set to 0.65 and 0.3, respectively. As seen from the figure, ground and wall surfaces, decks, fences, and window sills are segmented from the building facade as individual objects.

The comparison of results from these two methods shows that VGS tends to over-segment, namely details of a complete structure are segmented as independent parts. In contrast, the result of SVGS tends to be under-segmented retaining multiple objects as one single segment. For example, adjacent planar facets of the same facade are identified as one planar surface. Note that in the result of SVGS, many small independent details such as frames of different neighboring windows are merged as one segment, but in the case of VGS, over-segmented objects consisting of isolated voxels are removed as outliers from the output. Naturally, such a removal step in VGS will be counterproductive to the completeness of the output.

For the quantitative evaluation, we compare the proposed methods with baseline methods based on the manually segmented ground truth. In this test, the voxel resolution used in VGS, SVGS, and LCCP is set to 0.1m, equaling the radius of normal vector estimation in RG and the small radius of normal estimation in DON. The seed resolution of supervoxels in SVGS and LCCP

is 0.2m, equaling the graph size used in VGS and the large radius of normal vector estimation in DON. The threshold δ of graph segmentation used in VGS and SVGS is empirically set to 0.7 and 0.35, respectively. The threshold of normal difference used in RG and DON for smoothness is set to 15°. For the LCCP method, both the convexity tolerance and smoothness are set to 3.0. As displayed in Table 8.1, the proposed methods can outperform other baseline methods, with F_1-measure reaching approximately 0.81. It is also interesting that the RG method shows results comparable with those of VGS and SVGS, but as will be discussed later, VGS and SVGS require less execution time, and they are more computationally efficient.

Table 8.1: Evaluation of segmentation results of Sample 1 in the building facade dataset.

Method	Laser scanned		
	P_r	R_e	F_1
RG	0.8857	0.6957	0.7793
DON	0.5560	0.6160	0.5844
LCCP	0.6523	0.6840	0.6677
VGS	0.8562	0.7559	0.8029
SVGS	0.8403	0.8084	0.8240

8.1.2 Results of the construction site scene

For the tests conducted on the construction site scene, Figs. 8.1c-8.1f depict the segmentation results obtained from the VGS and SVGS methods, on both the LiDAR and photogrammetry point clouds. Similar to the result of building facades, the segments are rendered with varying gray values. The parameters of methods are the same as the ones used for the test using the building facade dataset. It appears that the construction site scene is more complicated than the building facade scene, significantly increasing the difficulty of segmentation. This hypothesis is backed by the obtained result as well, which is inferior to the result of building facade tests. Comparing results of using LiDAR and photogrammetric point clouds based on visual inspections, it seems that for segmenting major structures of the given point cloud, the result of using LiDAR data is generally much better. One possible explanation is that, in contrast to LiDAR points, our photogrammetric point clouds usually have a bit higher percentage of outliers caused by matching errors during the stereo matching process, which may affect the segmentation result. Additionally, with respect to the preservation of concave and "stair-like" connections, the VGS method shows better performance than the SVGS method when using the photogrammetric dataset. One possible reason is that for the SVGS method, the supervoxels are clustered based on normal vectors, which are sensitive to the stronger noise and outliers existing in our photogrammetric point clouds.

Quantitative evaluations are given in Tables 8.2 and 8.3. For both of the two samples, VGS and SVGS methods can outperform the other methods, which exhibit F_1-measures exceeding 0.67 for both the laser scanning and photogrammetric datasets. For the laser scanning datasets, all the methods show similar performance for both samples. However, when it comes to the photogrammetric dataset, the results of Sample 3 are inferior to those of Sample 2 for all methods. One possible reason is that the errors in the photogrammetric point cloud are more pronounced than those in the laser scanning point cloud. This is mainly because of the limited observing positions of acquiring optical images for the construction site, namely, we can only obtain images of different parts of the scene from certain positions so that the distance from the camera to objects in Sample 3 is larger than in Sample 2. A larger object distance decreases the ground resolution of the image (i.e., the size of the footprint of the pixel), generating more sparse point

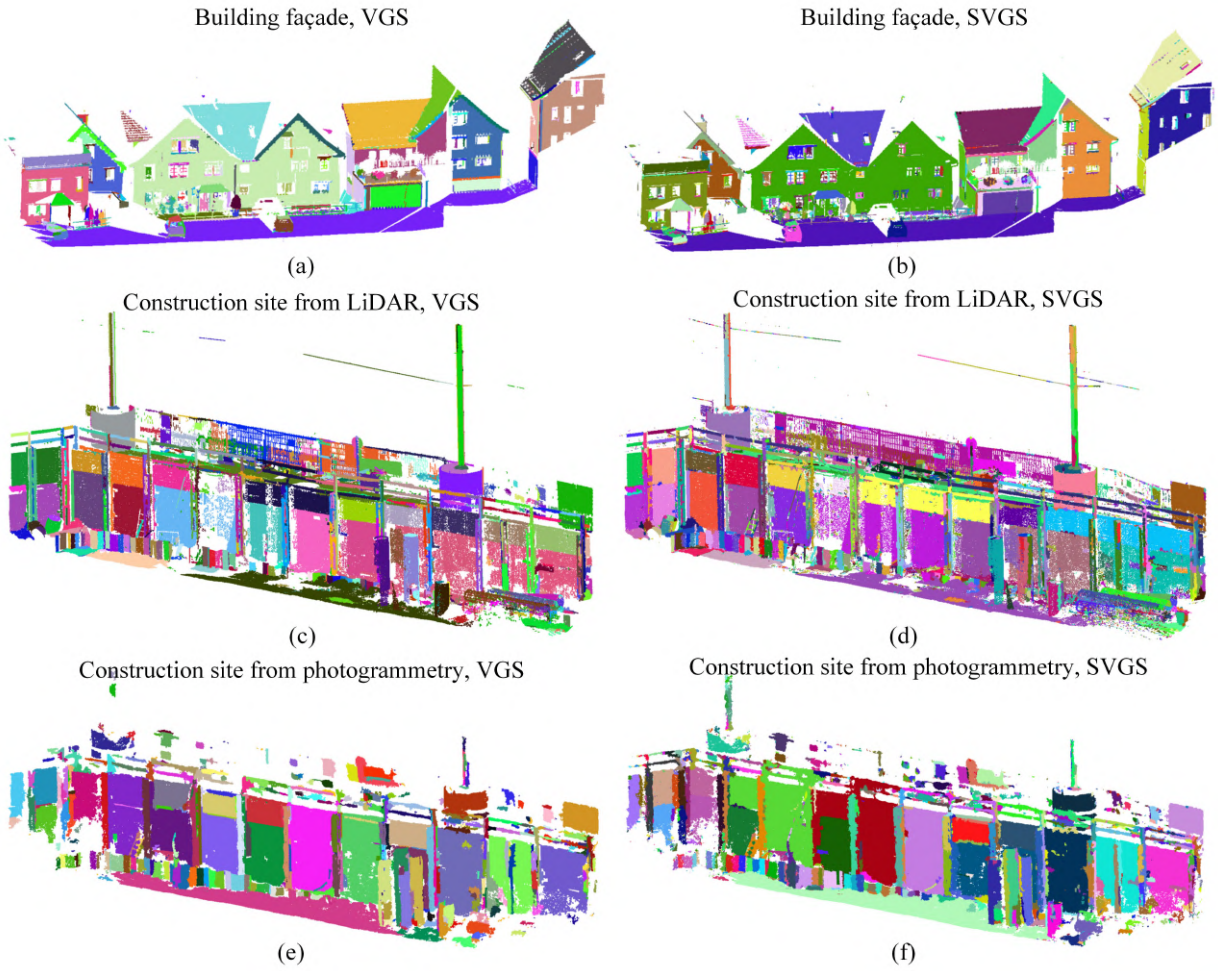


Figure 8.1: Segmentation of the building facade using (a) VGS and (b) SVGS methods. Segmentation results of the construction site using (c) VGS and (d) SVGS methods with LiDAR dataset, and using (e) VGS and (f) SVGS methods with photogrammetric dataset.

clouds, which may decrease the quality of the segmentation result, because to some degree the sparse point density may affect the reliability of the eigenvalues computed from the points.

Table 8.2: Evaluation of segmentation results of Sample 2 in the construction site dataset.

Method	Laser scanned			Photogrammetric		
	P_r	R_e	F_1	P_r	R_e	F_1
RG	0.6098	0.5799	0.5945	0.6371	0.6807	0.6582
DON	0.5875	0.5160	0.5495	0.5649	0.5269	0.5452
LCCP	0.5950	0.5250	0.5578	0.6104	0.5694	0.5892
VGS	0.7105	0.7077	0.7091	0.7655	0.7306	0.7476
SVGS	0.7205	0.7283	0.7244	0.7163	0.7420	0.7289

It is interesting that when using the Sample 2 point cloud, the results achieved for the photogrammetric datasets seem to be even better than those for the LiDAR data for both VGS and SVGS methods in the light of the F_1-measures, but these numbers are inconclusive, because the manually segmented ground truth of the photogrammetric dataset is different (coarser) than the

one of the LiDAR dataset. For the photogrammetric dataset we used, it is difficult to manually segment the point cloud even for a human operator because of its relatively poor quality. Although the quality of the ground truth results in counterintuitive values for F_1-measures, the ranking of the methods achieved for identical input data still shows that the proposed methods perform best.

Table 8.3: Evaluation of segmentation results of Sample 3 in the construction site dataset.

Method	Laser scanned			Photogrammetric		
	P_r	R_e	F_1	P_r	R_e	F_1
RG	0.6687	0.6532	0.6609	0.5629	0.5837	0.5731
DON	0.5745	0.5324	0.5526	0.5113	0.5446	0.5275
LCCP	0.6743	0.5676	0.6164	0.6144	0.6012	0.6077
VGS	0.7960	0.7113	0.7513	0.6843	0.6674	0.6758
SVGS	0.8404	0.7301	0.7814	0.7054	0.6413	0.6718

8.1.3 Geometric primitive recognition

The recognition of primitives is, in fact, a labeling process conducted on the segments. For a quantitative evaluation of the recognition of shapes, the confusion matrices (see Table 8.5) of the extracted primitives are calculated, using the segmented results of the VGS method. In the confusion matrix, the segments are recognized as three categories of primitives (i.e., Cylindrical, linear, and planar objects) and one category of unclassified objects (i.e., the object not belonging to those three types of shapes). The counted numbers of the valid segments (i.e., these segments having largest overlaps with their corresponding manually segmented ground truth) from LiDAR and photogrammetric sample point clouds are 67 and 38 from 81 and 51 obtained segments, respectively.

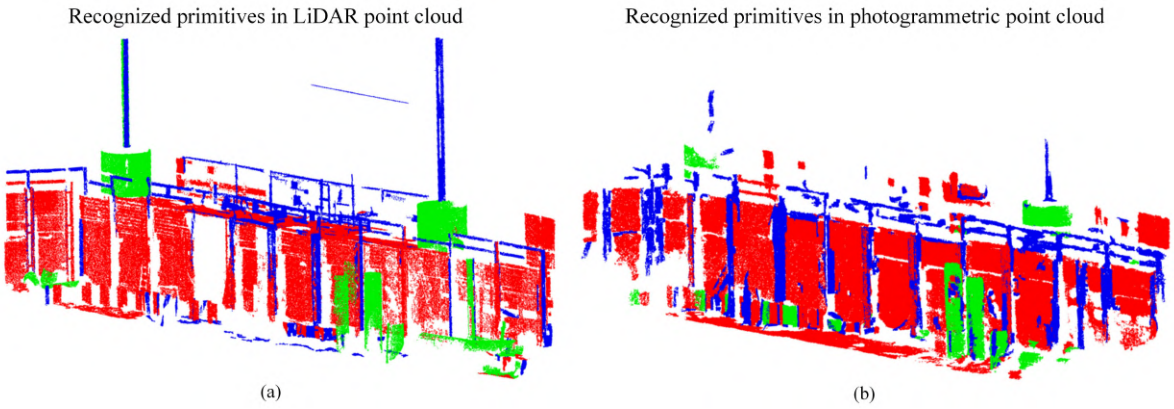


Figure 8.2: Recognized primitives of (a) laser and (b) photogrammetric point clouds.

Seen from the table, the correctness of three kinds of extracted primitives ranges from 0.64 to 0.88. For the results of the LiDAR data, the correctness of all the three kinds of primitives exceeds 0.8. However, if we consider the accuracy of unclassified objects, the overall extraction correctness is dropped to around 0.73. This is because many of the unclassified objects are wrongly recognized as planar or cylindrical ones, since they may consist of small planar and curvature surfaces. This phenomenon reminds us that when dealing with the complicated situations like a construction site in our future work, we need to increase the type of predefined shapes of objects so that

we can delineate the scene more accurately. Similarly, when using the photogrammetric data, planar and cylindrical segments are frequently wrongly recognized as linear objects, significantly decreasing their corresponding extraction correctness, with the values merely about 0.65. One reason for such problem can be linked to the quality of photogrammetric point cloud, which contains lots of the stereo matching errors due to the similar patterns and color of materials in a construction site. The entire extraction results of geometric primitives in the given construction site are demonstrated in Figs. 8.2a and 8.2b. The linear, planar, and cylindrical primitives are marked with red, green, and blue colors, respectively. Apparently, the majority of the geometric primitives we concerned are recognized and extracted. Noting that the sources of point clouds are crucial to the final recognition and extraction performance, and in our cases, the LiDAR data with a better geometric accuracy outperforms the photogrammetric point cloud. In Fig. 8.3, we provide an example of the recognized and extracted cylindrical objects. In this scene, we can find that for for all the cylindrical objects including pipes and steel frames, we got two successfully recognized ones, one partly extracted object, and one failed case. It is noted that the scene is highly occluded with low quality points, so that increase the difficulty of the recognition process revealing the feasibility of our method.

Table 8.4: Confusion matrix of recognition results for primitives

Predict		Laser scanned			Photogrammetric			
True	Cylinder	Linear	Planar	Other	Cylinder	Linear	Planar	Other
Cylinder	0.8	1.0	0	0.1	0.67	0.17	0.17	0
Linear	0.03	0.88	0.03	0.08	0.13	0.78	0.04	0.04
Planar	0	0.06	0.88	0.06	0.09	0.18	0.64	0.09
Other	0.24	0.18	0.35	0.24	0.31	0.13	0.31	0.25
Overall		0.7349				0.5893		

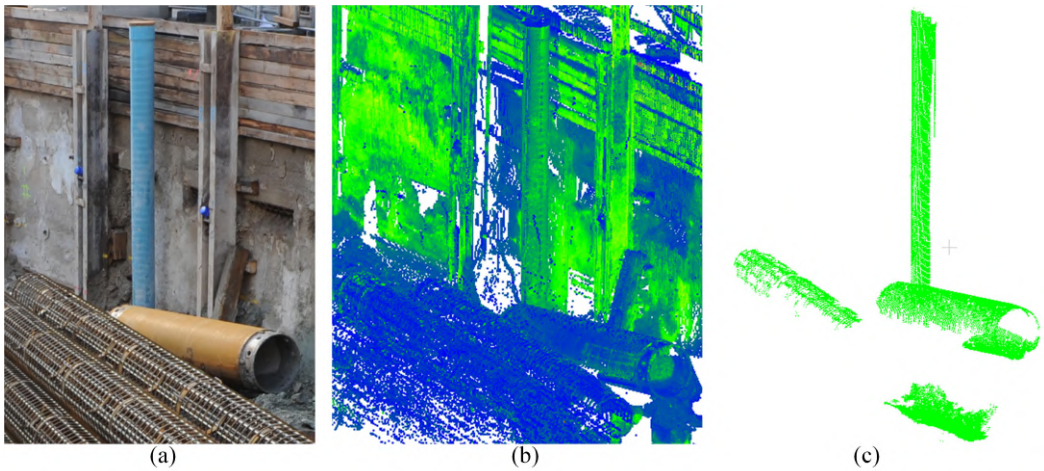


Figure 8.3: Illustraiton of recognized cylindrical primitives. (a) real scene, (b) LiDAR point clouds, and (c) Extracted primitives.

8.1.4 Influence of using different parameters

To thoroughly investigate the performance of the proposed method, we generate precision-recall (PR) curves of segmentation results using manually segmented samples for both the proposed approaches and different baseline methods, by changing thresholds and parameters of the segmentation methods. Here, the voxel resolutions used in LCCP, VGS, and SVGS are fixed to

0.1m, equal to the radius of normal vector estimation in RG and small radius of normal vector estimation in DON. Seed resolutions of supervoxels in LCCP, VGS, and SVGS, as well as the large radius of normal vector estimation in DON, are all set to 0.25m. The parameters that are changed control the granularity of the final segmentation results, having an impact on whether the algorithm will deliver an over-segmentation or an under-segmentation. For RG and DON methods, the changed threshold is the angle difference of normal vectors, ranging from 5° to 90°, while for the proposed VGS and SVGS methods, the changed parameter is the threshold δ for graph segmentation, ranging from 0.1 to 1.0. For LCCP, the changing values are the convexity tolerance and smoothness, both of them ranging from 0.1 to 5.0. The testing datasets are Sample 1 (see Fig. 7.7a) and Sample 3 (see Fig. 7.7b), in the latter case involving laser scanning and photogrammetric point clouds.

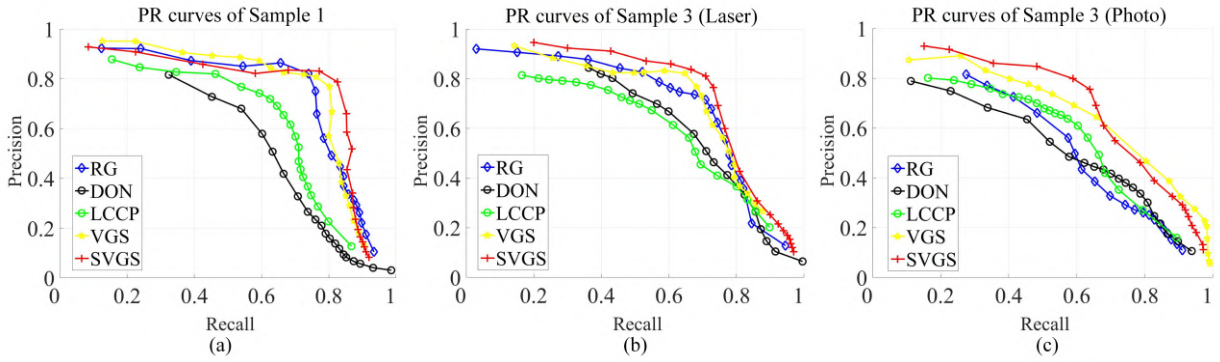


Figure 8.4: PR curves of (a) Sample 1, (b) Sample 3 (Laser scanning) and (c) Sample 3 (Photogrammetric).

As shown in the PR curves of Fig. 8.4, the proposed methods have better performance than the baseline methods for all the three testing datasets when the recall value is larger than 0.75. Besides, the shapes of these PR curves also indicate that the proposed methods can obtain better segments with a good trade-off between precision and recall. Specifically, for both laser scanning datasets, the classical RG method can obtain approximate or even better precision values than those of VGS and SVGS with small recall values. This phenomenon indicates that the RG method tends to create over-segmented results for the test datasets. Such a phenomenon can also be observed in the result of the LCCP method. This is because the smoothness and convexity criteria used in LCCP can segment planar surfaces and box shape objects well, but when dealing with complex surfaces or structures, for example, objects of linear shape or rough surfaces with patterns, they are likely to generate over-segmented fragments, breaking the entire structure into small facets. In contrast, as for the result using our photogrammetric dataset, the inferior geometric accuracy of our photogrammetric points degrades the segmentation performance of all the methods, especially for point-based methods (i.e., RG and DON), with their precision and recall values lower than 0.6 at best. This may be explained by the fact that the stereo matching errors in our dataset, which may result from our image configuration and distances from objects, influence the reliability of estimated normal vectors of points. In other words, these normal vectors are wrongly estimated, leading to blurred details of objects and in consequence to wrong segmentation results.

For the proposed methods, as expected, the threshold of graph segmentation plays a crucial role and is responsible for the over-segmentation or under-segmentation of obtained results. To give a more detailed view, we illustrate segmentation results of using VGS and SVGS with three varying thresholds of graph segmentation in Fig. 8.5. As seen from the figure, it seems that too large threshold will cause will cause under-segmentation while too small thresholds will lead

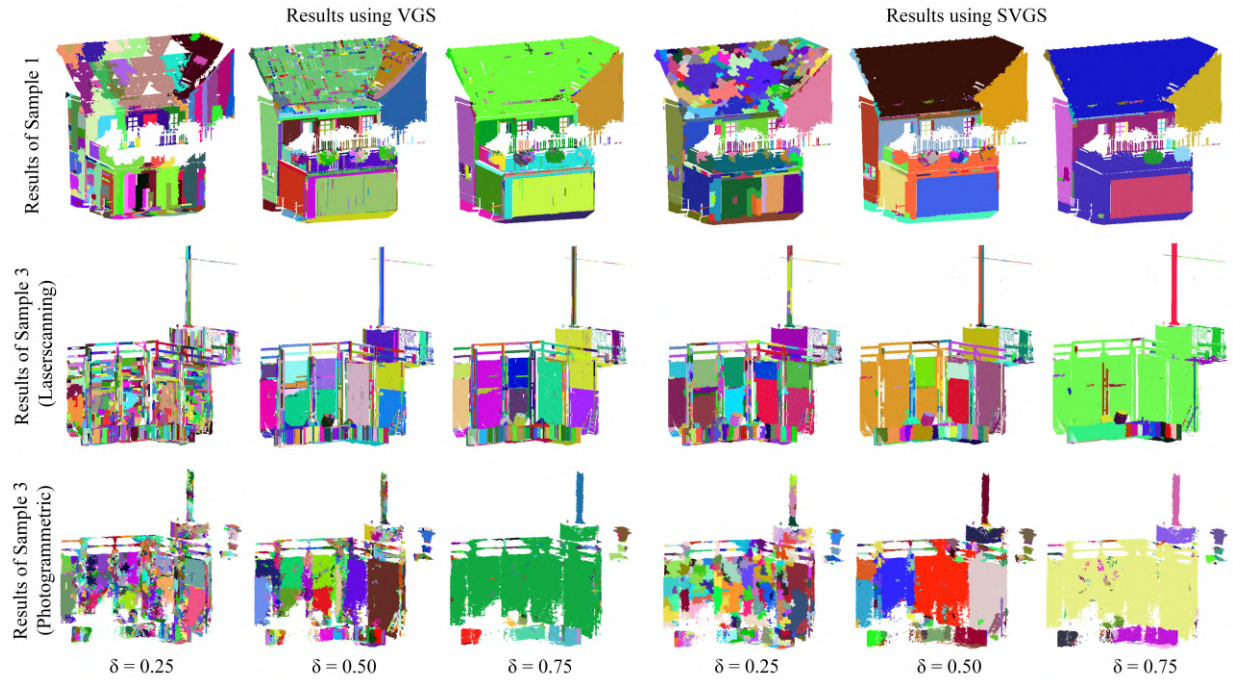


Figure 8.5: Segmentation results of using VGS and SVGS with different thresholds of graph segmentation.

to over-segmentation. Such results suggest that the use of inappropriate thresholds will result in a wrong partition of the local contextual graph so that connections of voxels are wrongly estimated. Besides, comparing the results of these two methods, it can be observed that the VGS method is more sensitive to the change of thresholds. To be specific, for the VGS method, it is evident that the smaller the threshold, the finer the granularity of obtained segments, whereas for the SVGS method, the point cloud has already been appropriately segmented with a small threshold. Then, with the increasing thresholds, the quality of the segmentation results does not change significantly. Furthermore, another interesting finding is that despite using the same threshold and parameters, segmentation results of different data types (i.e., laser scanning or photogrammetric points) are entirely different. Specifically, laser scanning datasets are more sensitive to changes of thresholds than photogrammetric ones. This phenomenon can be easily observed by comparing shapes of PR curves from Figs. 8.4b and 8.4c, namely we can hardly find obvious turning points of these PR curves for all the tested methods.

8.1.5 Granularity of voxelization

As mentioned earlier, the granularity of the voxel structure is also an essential factor influencing the quality of obtained segments using the proposed methods. For analyzing the underlying relationship between the granularity of voxelization and the accuracy of segments, we conduct experiments by using different sizes of voxels and supervoxels, with certain thresholds δ for graph segmentation. Here, for a given threshold δ of graph segmentation, voxel resolution used in both VGS and SVGS ranges from 0.05m to 0.25m, with an incremental interval of 0.025 m. The seed resolution of supervoxels in SVGS is set to three times the voxel resolution used. In these tests, the threshold δ of graph segmentation is set to 0.5. The testing datasets are Sample 1 (see Fig. 7.7a) and Sample 3 (see Fig. 7.7b), involving both laser scanning and photogrammetric point clouds. In Fig. 8.6, the F_1-measures of the segmentation results are displayed.

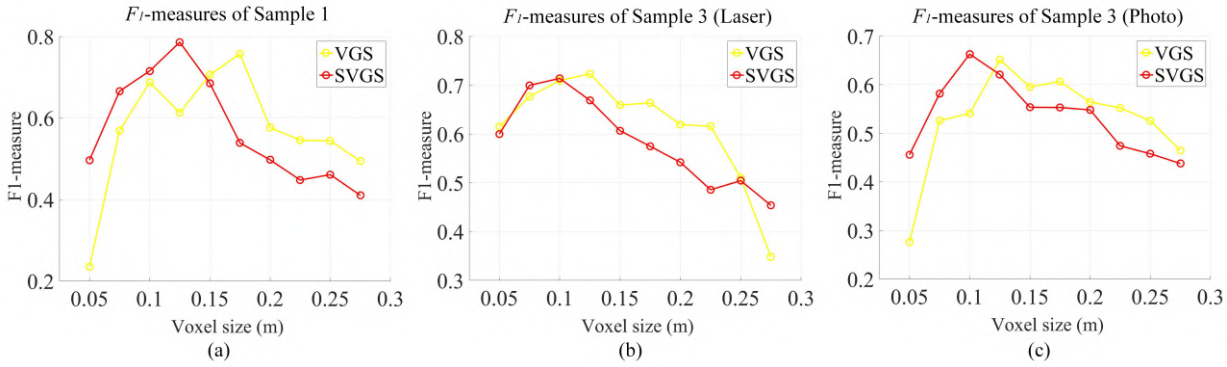


Figure 8.6: F_1-measures of (a) Sample 1, (b) Sample 3 (Laser scanning), and (c) Sample 3 (Photogrammetric), using different voxel granularities.

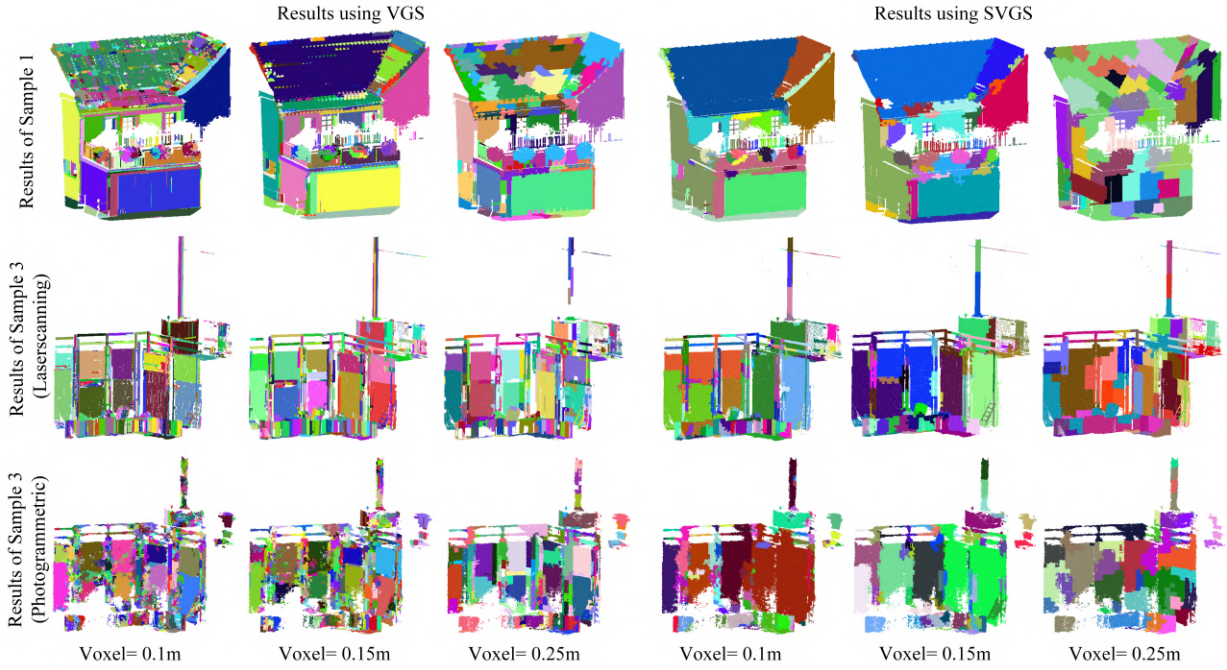


Figure 8.7: Segmentation results of using VGS and SVGS with different voxel granularities.

As seen from the figure, we can find that only with an appropriate resolution of voxels, our proposed methods can achieve optimal segmentation results, namely sizes corresponding to the peaks of these plots. For results of VGS, the highest F_1-measures of these three datasets are around 0.8, 0.70, and 0.66, respectively. While for the SVGS method, the highest F_1-measures of these three datasets are around 0.75, 0.71, and 0.65, respectively. It is an interesting finding that compared with SVGS, the VGS method requires a relatively smaller voxel resolution for the same dataset under the same threshold of graph segmentation. To give a more straightforward view, in Fig. 8.6, we illustrate segmentation results of using VGS and SVGS methods with different voxel granularities. Seen from the figure, we can deduce that both too large or too small sizes of voxels will result in over-segmentation of the entire scene. However, there are some slight differences. To be specific, if the voxel resolution is too small, over-segmented objects appear as individually isolated voxels. This is because that points within some voxels are too few so that they cannot show specific geometric shapes for judging the connection between voxels. Thus,

they are customarily regarded as outliers and finally become segments consisting of merely one voxel. On the contrary, a too large voxel resolution will cause errors when segmenting the local graph, because each voxel contains too many points related to more than one structure or object so that points within it generate relative general geometric information, namely all the voxels have similar attributes with no distinctiveness. Therefore, as nodes in the local graph, such large voxels will naturally cause wrongly partition results of the graph model.

In Fig. 8.6, it can be seen that only with an appropriate resolution of voxels, the proposed methods can achieve optimal segmentation results. These optimal results correspond to the maxima of these curves. For results of VGS, the highest F_1-measures of these three datasets are around 0.8, 0.70, and 0.66, respectively, whereas for the SVGS method, the highest F_1-measures of these three datasets are around 0.75, 0.71, and 0.65, respectively. It is an interesting finding that compared to SVGS, the VGS method requires a relatively smaller voxel resolution for the same dataset under the same threshold of graph segmentation. For providing a more direct view, in Fig. 8.7, we illustrate segmentation results of using VGS and SVGS methods with different voxel granularities. Based on the figure, we can deduce that both too large or too small sizes of voxels will result in over-segmentation of the entire scene. However, there are some slight differences. Specifically, for a too small voxel size, over-segmented objects appear as individually isolated voxels. This is because, within some voxels, there are too few points, so that the geometric features cannot be estimated reliably enough to judge about the connectivity of voxels. Thus, they are typically regarded as outliers and finally become segments consisting of merely one voxel. On the contrary, a too large voxel size will cause errors when segmenting the local graph, because each voxel contains too many points related to more than one structure or object, so that the points inside carry relatively general geometric information, namely all the voxels have similar attributes with no distinction. Therefore, as nodes in the local graph, such large voxels will naturally lead to an incorrect partition of the graph.

8.1.6 Comparison of execution time

Efficiency is also an important criterion for assessing the performance of an algorithm. To thoroughly explore the efficiency of the proposed strategy, we carry out a set of execution time tests with different sizes of datasets. As the granularity of elements (e.g., voxels, supervoxels, and neighborhoods of normal vectors) can significantly influence the computational cost, in these experiments, we also compare results of using different granularities of elements. In detail, there are three groups of tests using a different granularity of voxels and supervoxels. Here, the voxel sizes used in LCCP, VGS, and SVGS are set to 0.05 m, 0.1 m, and 0.2 m for different groups of tests, equaling the radius of normal estimation in RG and the radius of small normal estimation in DON. The seed resolution of supervoxel in LCCP and SVGS, as well as the radius of large normal estimation in DON, are set to 0.125 m, 0.25 m, and 0.5 m for different groups of tests, respectively. Comparisons of the execution time are given in Fig. **??**. Here, the execution time contains all processing steps, including the time required for creating the octree structure.

As demonstrated in Fig. 8.8, point-based methods (i.e., RG and DON) require a much longer execution time than the voxel-based methods, regardless of the size of the neighborhood for normal vector estimation, voxels and supervoxels. As for the proposed methods, for small datasets, the execution time of VGS is approximately the same as that of SVGS, but with increasing size of datasets, the execution time of SVGS becomes longer than that of VGS. This is mainly due to the calculation of element attributes. Although the use of supervoxels reduces the number of elements, larger elements (i.e., supervoxel) include more points, requiring a longer computation time for calculating the attributes. Moreover, the generation of supervoxels also consumes additional time. Besides, the proposed methods are inferior to the LCCP method with respect to efficiency. This

is because of the time consumed by the graph-based segmentation step, used in the proposed methods.

Comparing the results of using a different granularity of elements, we can also find that voxel-based methods have an advantage over the point-based baseline methods. Namely, our methods show stable execution time when using varying granularity. Specifically, the proposed methods and LCCP are not sensitive to increasing voxel and supervoxel size. In contrast, the execution time of RG and DON methods grows drastically when utilizing a larger neighborhood for estimating normal vectors. This is because, for the point-based methods, the estimation of normal vectors is applied to every point in the dataset. The larger the neighborhood selected, the longer time is needed, whereas for voxel-based methods, the estimation of normal vectors is applied to elements (i.e., voxels and supervoxels), and all the points inside the element share the same normal vector. Larger elements will decrease the number of calculated normal vectors. However, using larger voxel or supervoxel sizes may also blur the details of segments, and result in "zig-zag" edges of segments. Thus, finding the appropriate granularity of voxel structures for voxel-based segmentation methods requires a trade-off between accuracy and speed.

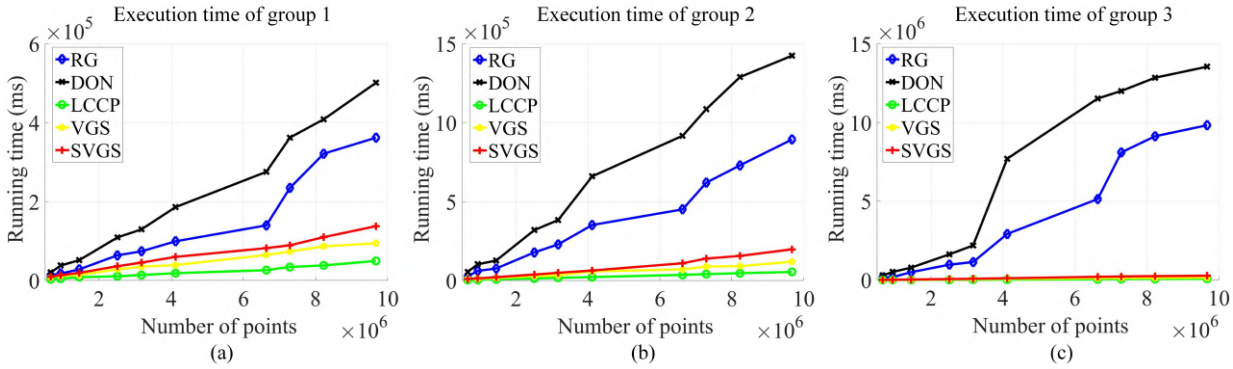


Figure 8.8: Comparison of execution time. Tests of (a) group 1, (b) group 2, and (c) group 3.

8.2 Segmentation of buildingobjects in built environment

In the following experiments, all the algorithms used are implemented via C++ and run on an Intel i7-4710MQ CPU @ 2.5GHz and with 16.0 GB RAM. Reference methods (i.e., RG, VIS, and LCCP) are implemented with PCL 1.8.0 [Rusu & Cousins, 2011].

The results of the segmentation using our proposed method are compared against the manually segmented ground-truth dataset in a point to point way. Four representative point cloud segmentation algorithms, namely the smoothness based Region Growing (RG) [Rabbani et al., 2006], Voxel-based Incremental Segmentation (VIS) [Wang & Tseng, 2011], Locally Convex Connected Patches (LCCP) [Stein et al., 2014], and Supervoxel and Graph-based Segmentation (SVGS)[Xu et al., 2016b] are used as baseline methods. Here, the RG method is a renowned point-based segmentation algorithm, which has been widely used in plentiful applications. While the VIS method is a representative voxel-based segmentation method using proximity and co-planarity as cues for segmentation. In contrast, the LCCP method is a popular supervoxel-based segmentation method, adopting the convexity as the segmentation criteria. The SVGS method is our recently published work, which initially combines the supervoxel structure and perceptual grouping laws in segmentation but it uses Markov clustering for identifying the connection.

8.2.1 Quality of point clouds

In these experiments, voxel sizes used in our UHCG, VIS, LCCP, and SVGS are 0.1m, equaling to the radius of normal estimation in RG. That means that the normal vectors used in the different methods are calculated all by using a neighborhood of same size, namely a sphere or cubic of 0.1 m size. Seed resolutions of supervoxels in our proposed method, LCCP, and SVGS are both set to 0.25 m. The size of the neighborhood for aggregating the supervoxels is set to 0.5m. The threshold for angle difference of normal vectors is set to 15°. For our proposed method and SVGS, thresholds for graph segmentation are set to 0.5. Scale factors for estimating the arbitration of cues are set to 0.25 empirically.

In Figs. 8.9a and 8.9b, the points density distribution is given, while in Figs. 8.9c and 8.9d, the statistic of the mean distance between KNN is displayed. As seen from Figs. 8.9a and 8.9b, points densities of these two testing datasets are various. The density of the point cloud of St. Gallen is lower than that of the town square. Varying points densities may result in inferior results of segmentation because the calculation of geometric cues largely depends on points inside the supervoxel. Theoretically, the more points the supervoxel have (i.e., higher points density), the more reliable the calculated geometric cues are. As for the mean distances between k-nearest neighbors, the point cloud of town square has lower mean distance values, when compared with that of St. Gallen. The possible outliers, referring to the points having a large mean distance, appear more frequently in the point cloud of Town square. It is noted that points having mean distances larger than 5 m are not illustrated in Figs. 8.9c and 8.9d . To cope with sparse outliers existing in the datasets, we add a constraint for voxels, removing those voxels having a number of points less than three, which is the minimum number of points for estimating the normal vector.

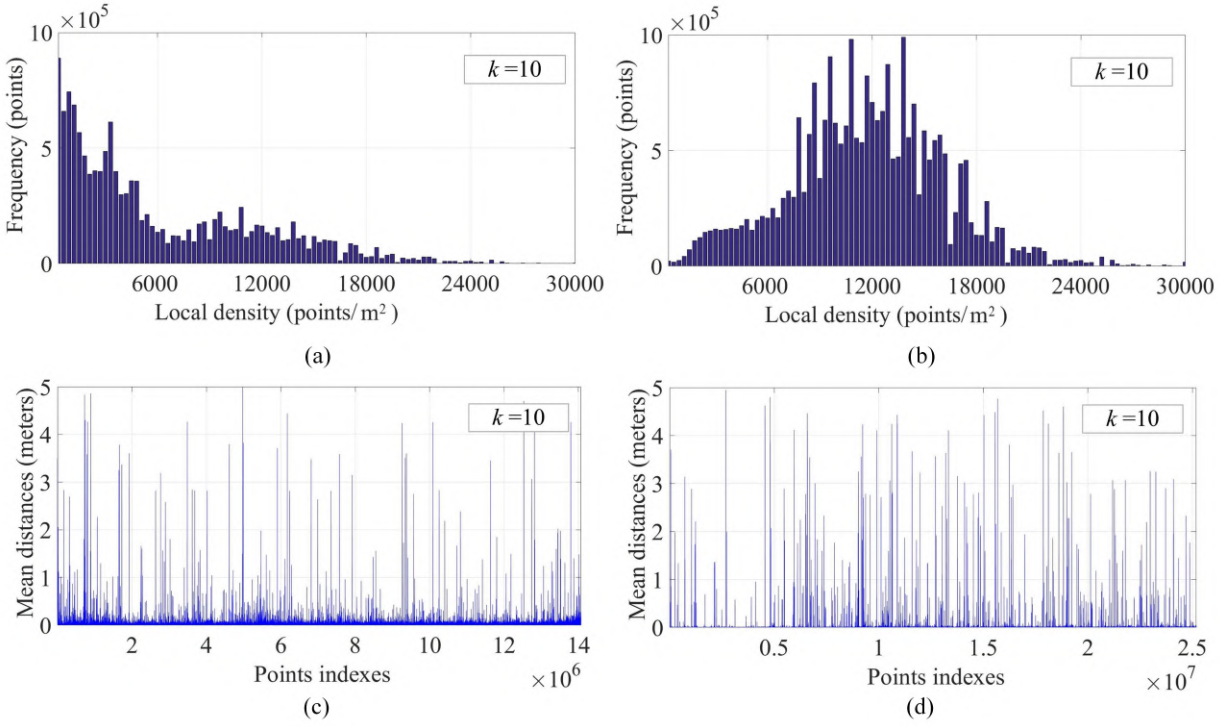


Figure 8.9: Points density distributions. (a) St. Gallen. (b) Town square. Mean distances between k-nearest neighbors. (c) St. Gallen. (d) Town square.

8.2.2 Qualitative results

In Fig. 8.10, segmentation results of two experimental point clouds are illustrated, with different segments rendered with varying colors. The execution time of segmenting these two scenes is 689.847s and 408.303s, respectively. It can be seen from the figure that point clouds of the urban scenes are partitioned into segments having certain geometric shapes. For example, the ground and wall surfaces, roofs, eaves, and window sills are segmented as individual objects. However, for adjacent facades owning co-planar arrangements, many of them are not distinguished, namely under-segmented. As a common problem for point cloud segmentation, this phenomenon has reported and discussed in many works of building roof segmentation from ALS point clouds [Aljumaily et al., 2017], because such roofs and facades have no distinctive differences from the geometric aspect. Besides, for vegetations (e.g., bushes and tree branches) which have few homogeneous geometric features, the over-segmentation frequently occurs, displayed as random clusters of points. One of the reasons for such a phenomenon is due to the generation of supervoxels. During the supervoxel generation, we only consider the proximity and continuation cues, which is represented in the form of the distance between voxels and seeds and the consistency of normal angles. For voxels composing vegetation, their normal vectors can not show clear consistency, so that supervoxels generated can hardly find correct boundaries of bushes or tree branches. Notably, for the area with too sparse point density (e.g., the ground surface in Fig. 8.10a), the over-segmentation can also happen. This is because such sparsely distributed points in the supervoxel may be judged as scatters rather than planar surfaces when considering the similarity cue. So that their probability of connectivity will be largely decreased when aggregating these supervoxels. This problem can be easily solved if we enlarge the resolution of seeds for supervoxels.

8.2.3 Quantitative results

In Fig. 8.11, segmentation results of selected samples of using different methods are illustrated. In Tables 8.5 and 8.6, quantitative evaluation are given. As seen from the figure and tables, for using both two reference datasets, our proposed method outperforms other baseline methods, with more objects completely segmented and F_1 measures larger than 0.66 achieved. For the segmentation of planar surfaces (e.g., ground surface and walls), all the used methods show good performance. Especially for RG and VIS, which utilize smoothness as the segmentation criterion, the flat area in the scene is well segmented. However, when it comes to connection parts between surfaces, over- and under-segmentation occur. For results of using LCCP, over-segmentation frequently happens when segmenting linear shape objects, for example, eaves located at the edge of roofs. This is because the judgment of extended convexity criterion used in LCCP requires singular connections formed by adjacent surfaces of supervoxels, but for such linear shape objects, there are no enough neighboring supervoxels for conducting the judgment of singular connections, so that incorrect segmentations may occur. The SVGS method shows similar results to those of our proposed method, with the F_1 measure better than around 0.63, but for the area with similar geometric characteristics (e.g., flatness and normal vectors), SVGS may over segment the area. For instance, the area of the ground in Fig. 8.11k separated by fences are segmented as individual objects using SVGS. Besides, in SVGS, the closure cue is not taken into consideration, so that for convex connections in the scene, they are always segmented into two planar surfaces, for example, the pinnacle consisting of multiple planar surfaces in Fig. 8.11e are separated as two parts. However, as we have discussed in the qualitative evaluation, both of the two methods cannot perform well when segmenting vegetations.

Besides, the comparison of execution time is given in Table 8.7, demonstrating that the efficiency of our proposed method is almost equal to our former SVGS method but much better

Figure 8.10: Segmentation of building scenes using UHCG. (a)St. Gallen. (b)Town square.

than the traditional point-based region growing method. However, our method is inferior to the LCCP and VIS methods with respect to efficiency. This is because the graph-based segmentation algorithm used in our proposed UHCG method and SVGS has an iterative process for optimally partitioning the graph, which is relatively time-consuming. In theory, the larger the neighborhood for aggregating the supervoxel is, the more complex the graph will be constructed, requiring a longer execution time of the entire segmentation process. However, using a larger voxel or supervoxel size may also blur details of segments. Thus, it is a tradeoff to find the appropriate granularity of voxel structures for voxel-based segmentation methods.

To thoroughly investigate the performance of our proposed UHCG, we generate the precision-recall (PR) curves of segmented results using a sample reference dataset (see Fig. 7.8) with different baseline algorithms, by changing thresholds of segmentation methods. Here, voxel sizes used in our method, VIS, LCCP, and SVGS are 0.1m, equaling to the radius of normal estimation in RG. Seed resolution of supervoxels in our proposed UHCG, LCCP, and SVGS are both set to 0.25m. The size of the neighborhood for aggregating supervoxels are set to 0.5m. All the changing

Figure 8.11: Original point clouds of (a) Sample 4 and (g) Sample 5. (b) and (h) results using RG. (c) and (f) results using LCCP. (d) and (j) results using VIS. (e) and (k) results using SVGS. (f) and (l) results using our proposed UHCG.

Table 8.5: Evaluation of segmentation results of Sample 4.

Method	Use reference 1			Use reference 2		
	P_r	R_e	F_1	P_r	R_e	F_1
RG	0.5883	0.6855	0.6332	0.5596	0.7213	0.6302
VIS	0.7958	0.5832	0.6302	0.6516	0.5781	0.6126
LCCP	0.5453	0.6867	0.6079	0.4640	0.7096	0.5611
SVGS	0.7288	0.6456	0.6847	0.6441	0.6798	0.6614
UHCG	0.8769	0.6352	0.7367	0.7381	0.5988	0.6612

Table 8.6: Evaluation of segmentation results of Sample 5.

Method	Use reference 1			Use reference 2		
	P_r	R_e	F_1	P_r	R_e	F_1
RG	0.7419	0.6009	0.6640	0.7678	0.5781	0.6596
VIS	0.5955	0.5751	0.5877	0.5736	0.5529	0.5631
LCCP	0.5541	0.5646	0.5593	0.5327	0.5717	0.5515
SVGS	0.5979	0.6589	0.6269	0.6008	0.6663	0.6319
UHCG	0.6840	0.6847	0.6843	0.6904	0.6510	0.6701

Table 8.7: Comparison of execution time for different methods.

Time(s)	RG	VIS	LCCP	SVGS	UHCG
Sample 4	2047.073	32.639	68.074	226.750	284.128
Sample 5	1787.611	22.136	22.136	195.067	247.784

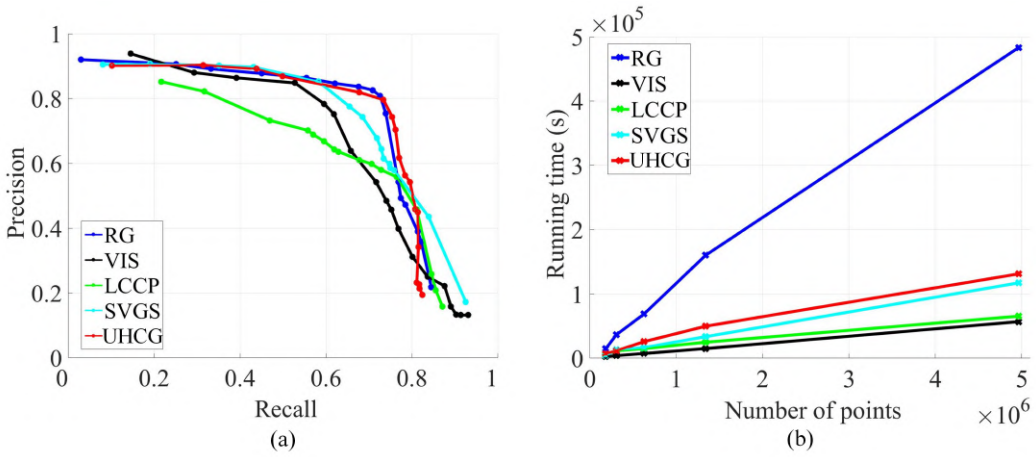


Figure 8.12: PR curves and time comparison. (a) Precision-recall curves. (b) Execution time comparison.

thresholds aim at controlling the granularity of the final segmentation results. For VIS and RG, changing thresholds are the angle difference of normal vectors, ranging from 5° to 90°. While for our UHCG and SVGS, changing values are the threshold for graph segmentation, ranging from 0.1 to 1.0. For LCCP, the changing value is the convexity tolerance and smoothness, both of which range from 0.1 to 5.0. As shown in PR curves (see Fig. 8.12a), the UHCG method has better performance than the others when the recall value is larger than 0.725. In contrast, the conventional RG method can obtain approximate or even better precision values than that of our method with a small recall value. This reveals that, for the used testing sample data, RG tends to create over-segmented results. A similar phenomenon can be observed from the curve of using LCCP. This is because, for this segmentation method, the smoothness and convexity criteria can segment planar surfaces and box shape objects well, but when it comes to more complex surfaces or objects, such as linear shape objects or rough surfaces with patterns, they may generate over-segmented surfaces, breaking the entire object into small fragments. The shape of the PR curve indicates that our method can obtain better segments with a good trade-off between the precision and recall. Besides, we compare the execution time when dealing with different sizes of datasets. The execution time contains all processing steps, including the time for creating voxel and supervoxel structures. We can find that VIS requires the least computation time, while LCCP ranks second in execution time. Two of our methods (i.e., our UHCG method and the former SVGS method) require longer computation time than that of LCCP slightly. In contrast, the classic RG method spends the longest execution time, especially when dealing the large-scale point cloud, the execution time of RG increases rapidly. By contrast, the execution time of the rest of methods growths gently when the size of testing dataset increased.

8.3 Reconstruction of linear structural elements inconstruction sites

In the following experiments, all the methods are implemented via C++ and run on an Intel Core 2 Duo @ 2.2GHz and with 8.0 GB RAM. The baseline method for evaluating the performance of proporsed descripter is SHOT, which is implemented through PCL 1.7.1 [Rusu & Cousins, 2011].

8.3.1 Performance of the proposed descriptor

LSSHOT histograms for typical objects

Four typical linear shape objects of different types are generated synthetically for checking the consistency of features extracted by the proposed descriptor, including board, cylinder, triangular prism, and quadrangular. In Fig. 8.13, an illustration of LSSHOT feature histograms of points belonging to these objects is given. In these figures, the blue plot represents the histogram of one key point arbitrarily selected from the point cloud of the object, while the red plot exhibits the mean value of the histograms of all the points from the point cloud of the object. It can be seen from the figures that different kinds of linear straight shape objects exhibit distinctive feature histograms, with significant differences in both distribution and values of accumulated bins. In contrast, for points belonging to the same objects, the histograms agree with the mean value of the whole histogram, which means that for an object with certain geometric shape, the feature histograms are highly consistent, especially for the board and tube.

Shape matching results

Fig. 8.14 shows the recall versus noise level curves of shape matching tests using the proposed LSSHOT descriptor and SHOT descriptor. In this test, the 1-P_r value is fixed to 0.1. It can be seen from the figures that, for the results of the board, the recall values of both LSSHOT and SHOT descriptors gradually decline with the increase of noise level, but their recall values show similar values for all noise levels with a downward trend. When there is no noise added, the recall value can reach around 0.9, and then decreases to about 0.6 with the noise level increasing to 90%. In contrast, when it comes to tubes and prisms, the LSSHOT descriptor shows better results than those of SHOT, with a higher recall value when dealing with the dataset of the same noise levels. To be specific, in the case of the tube, the recall values of LSSHOT experience a gentle decline from more than 0.9 to approximately 0.7, while the recall values of SHOT suffer a dramatic decrease when the noise level is greater than 20%. It is interesting that for the prism both two descriptors display a stable tendency but the recall value of LSSHOT is higher than that of SHOT for all noise levels, with their values kept around 0.5 and 0.3 respectively.

The recall versus 1-P_r curves are shown in Fig. 8.15, in which the results obtained under three incremental noise levels using various objects are given, respectively. As we mentioned before, an ideal descriptor should have a recall equal to 1 for any precision [Mikolajczyk & Schmid, 2005]. It can be seen from the figures that in the case of the board, the recall versus 1-P_r curves of LSSHOT and SHOT reveal almost the same values and tendency. At the 1-P_r value of approximately 0.1, the recall value can achieve 0.7, and the recall value can be further improved to more than 0.9 by trading off some precision. It is noteworthy that the LSSHOT shows even inferior performance when the noise level is 90%. However, when using tube and prism as test objects, the LSSHOT exhibits explicit advantages over SHOT. For example, in the case of the tube, the recall value of LSSHOT can be higher than 0.7 with the 1-P_r value being limited to lower than 0.1. On the contrary, the precision of SHOT experiences an apparent decline when the recall value is increasing. One of the possible explanations for this difference is that the board

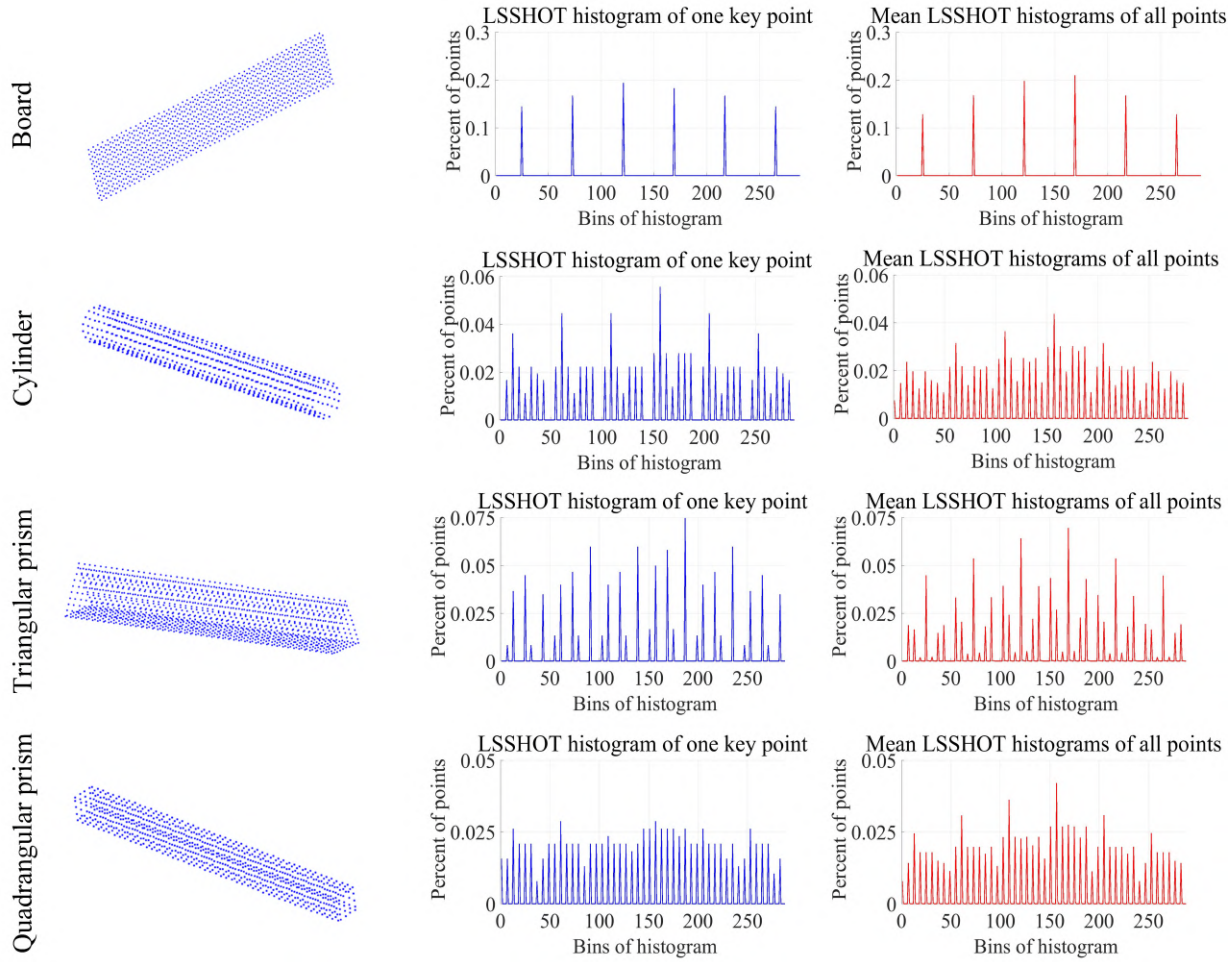


Figure 8.13: Typical linear shape objects and their LSSHOT histograms.

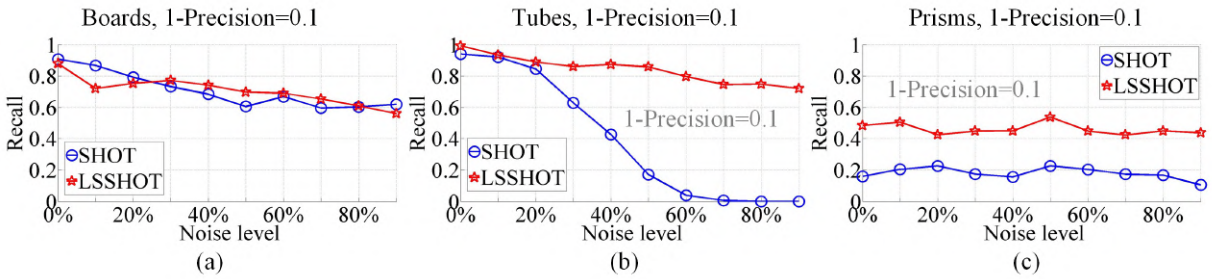


Figure 8.14: Recall versus noise level curves. (a) Boards. (b) Tubes. (c) Prisms.

has simple planar geometric surfaces, which is beneficial to normal vector estimation so that both descriptors can perform well. Nonetheless, when using objects with more complex geometric surfaces such as a tube with a cylindrical shape and curving surface, the estimation of the normal vector is susceptible to the noise level. As a consequence, in such case, the LSSHOT benefiting the utilization of linear characters shows better robustness than the SHOT, with the recall value reaching higher than 0.7 while the 1-P_r value is remaining smaller than 0.3.

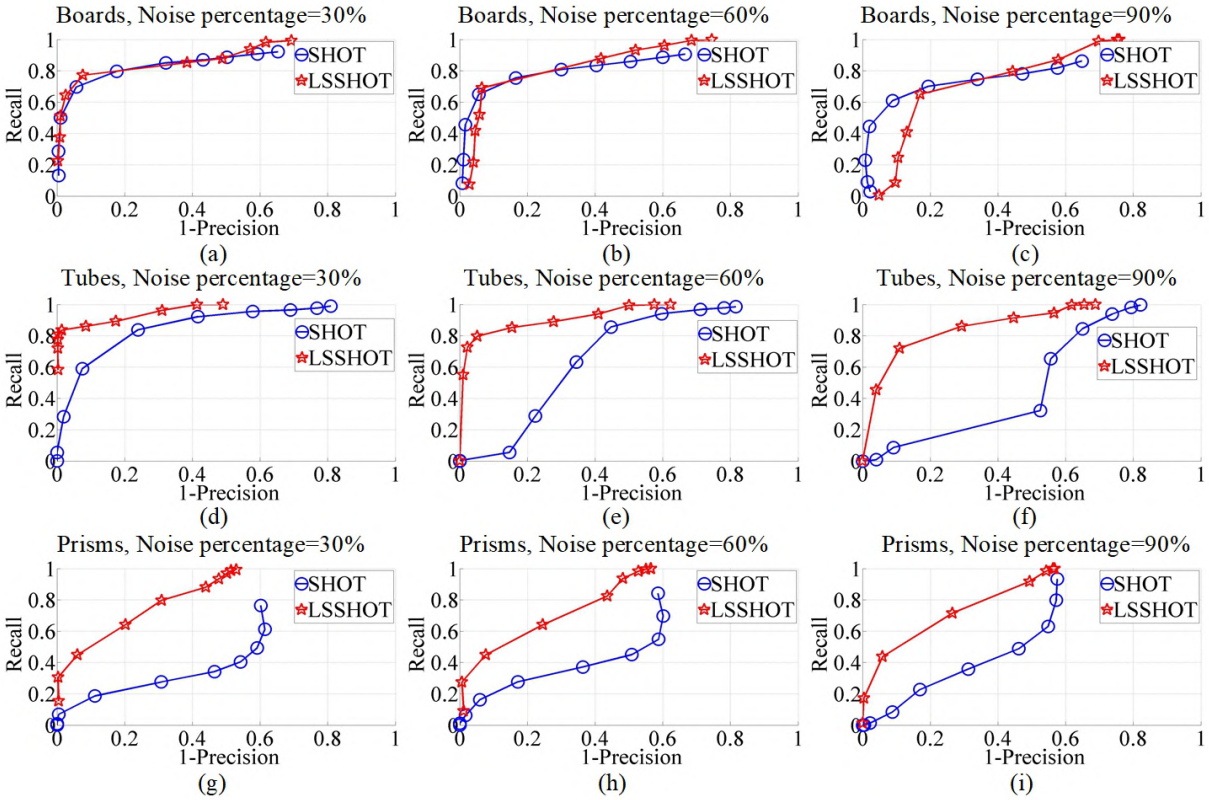


Figure 8.15: Recall versus 1-Precision curves with various noise levels.

8.3.2 Reconstruction of real scaffolding components

Preprocessed point cloud

After the statistical filtering and voxel grid based filtering processes, the dataset has been down-sampled to 855,000 points. Here, the size of the cube chosen for the voxel grid is 0.04 m, the point densities are then evenly reduced to maximum 16,000 points/m^3.

Separated building facades

In the vertical projection outcomes, the main vertical structures of both building and scaffolds shows clear dark patterns in the projection image in Fig. 8.16a, representing high densities and revealing the high overlap of points in the vertical direction due to the characteristics of vertical structures. In contrast, Fig. 8.16b exhibits the projection image after selection using local maximum value. As seen from the figure, most of the disturbing points and many dark pixels standing for horizontal structures have been filtered. Then, the point cloud representing vertical structures is given in Fig. 8.16c. It shows that the separated vertical structures mainly consist of three facades (facades I, II and III). However, it is also apparent that only two of these facades

(facades I and III) contain enough points for reconstruction. This is due to the point cloud generation, where not enough images for facade II were taken for a reliable point cloud generation. This deficiency of the point cloud we used will result in a limitation on our further reconstruction process, in which only scaffolding elements in the other two facades can be reconstructed.

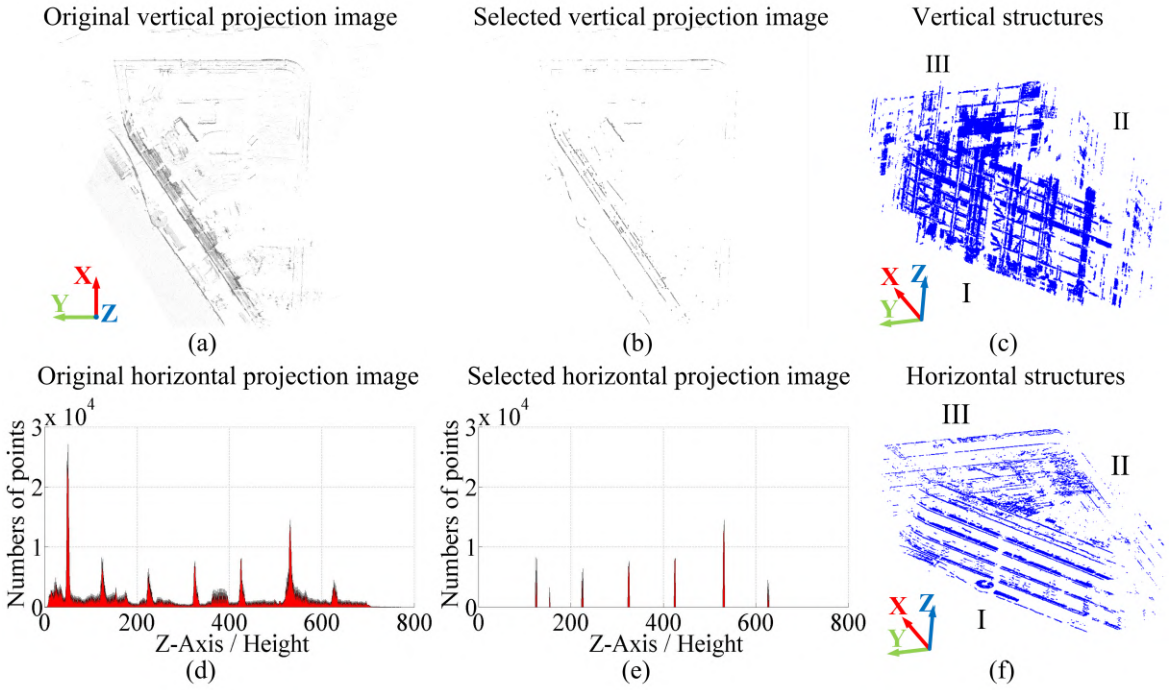


Figure 8.16: Vertical projection and horizontal slicing results. (a) Original projection image. (b) Selected projection image. (c) Vertical structures. (d) Horizontal projecting histogram. (e) Selected projecting histogram. (f) Vertical structures.

Similarly, in the horizontal projection process, the points belonging to horizontal structures appear to show up as peaks in the projecting histogram, which can be seen in Fig. 8.16d. It is noteworthy that one of the most significant peaks in this histogram near to zero height represents the points on the ground surface, which should be cut off in the selection process. In Fig. 8.16e, the peaks found by the mean-shift algorithm are illustrated. The extracted points linking to the horizontal structure, mainly floors and decks of scaffolds, are shown in Fig. 8.16f. From the figure, we can find that there are seven sliced horizontal planes, but only four of them are related to the decks of scaffolds, which will be further selected through the subsequent slicing procedure.

In Fig. 8.17, the statistics of parameters from extracted planar surfaces is given. There are in total 17 vertical planar surfaces extracted via RANSAC algorithm. The clusters appear in the figure standing for potential groups of facades with âsandwich-likeâ structure. After a selection process using the distances and parallelism, two clusters are finally chosen, representing facades I and III respectively. It is also interesting that one of the chosen clusters contains many parallel planar surfaces with very close distances showing up as overlapping dots in the figure. This phenomenon explains that during the extraction of planar surfaces, one possible planar surface (e.g., the outer row of scaffolds) may be extracted as several parallel planes due to the setting of threshold of RANSAC algorithm. Thus, such planar surfaces will be merged.

The grouping outcomes are illustrated in Fig. 8.18. As seen from the figures, the points of scaffolding components and building surfaces are separated and labeled. The red points represent the building surfaces in facades I and III, while the blue and green points stand for the outer and

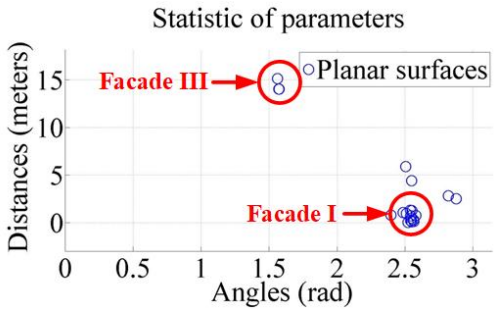


Figure 8.17: Statistical result of parameters of planar surfaces.

inner rows of scaffolds in facades I and III, respectively. Note that for facade I, both the outer and inner rows of scaffolds are extracted and labeled, but as for facade III, only the outer row of scaffolds is successfully extracted. This is due to the missing points in facade III, failing to extract planar surfaces. The same situation occurs in facade II, where no plane is extracted.

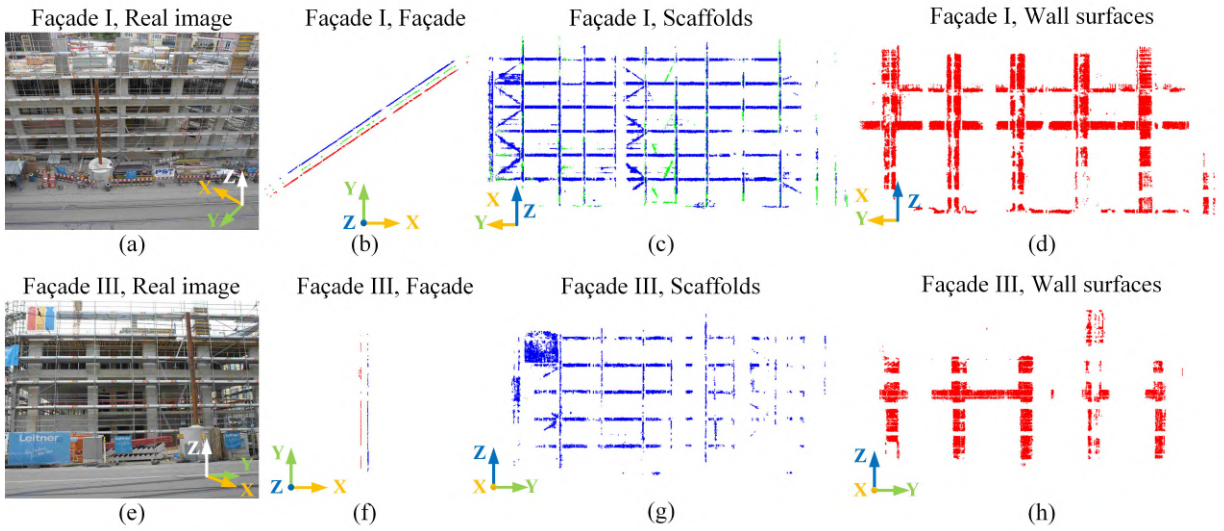


Figure 8.18: Grouping results of vertical planar surfaces in two facades. (a) and (e) Actual scenes. (b) and (f) Grouped facades. (c) and (g) Scaffolds of facades. (d) and (h) Building surfaces of facades.

In Fig. 8.19, the sliced results of decks are illustrated. As shown in Figs. 8.19c and 8.19d, the decks of scaffolds in facades I and II are sliced, with the number of cut decks being five and three respectively.

The statistical numbers of points belonging to scaffolds or building surfaces in facades I and III, which are obtained through grouping and slicing, are presented in Table 8.3.2.

Table 8.8: Number of points in grouping and slicing results.

Facade	Inner scaffolds	Outer scaffolds	Decks scaffolds	Building surface
I	19042	73158	65743	105242
III	—	30713	5108	29153

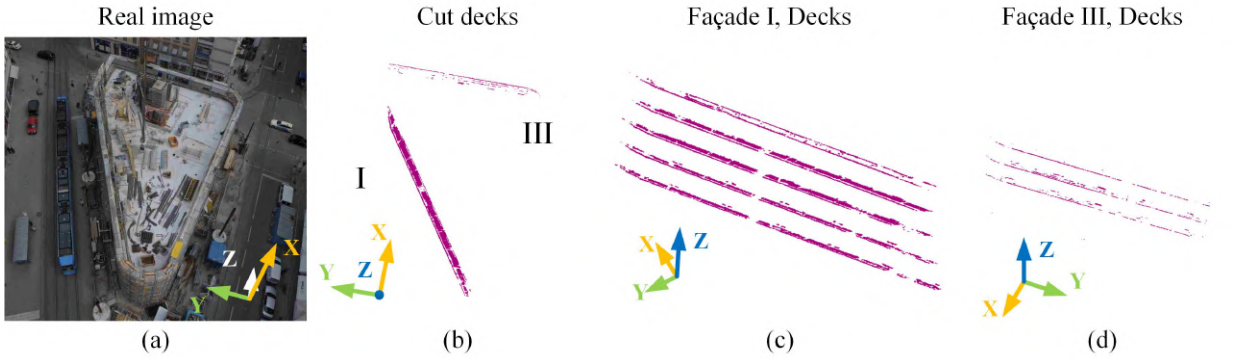


Figure 8.19: Sliced results of vertical planar surfaces. (a) Real scene. (b) Cut decks. (c) and (d) Decks of two facades.

Scaffolding points classification results

The results of the point classification are summarized in Table 8.9. Points of scaffolds in each facade are grouped into three categories: points of toeboards, points of tubes, and points of scatters. Here, the scatters denote rejection class of the points belonging to irregular point clusters with no certain shape or outliers, regarded as disturbing objects. In the training process of RF classifier, for the classes of tubes and toeboards, we select 2000 points of each class manually as training samples, whereas for the class of scatters, related to the rejection class, we also select 2000 points from the outliers and points of disturbance randomly for training. Also, to give a quantitative evaluation of the classification using actual photogrammetric point clouds, the classification results for the points in the outer row of facades I are investigated. In this case, a dataset of ground truth used for the assessment of the classification performance is segmented manually. In Fig. 8.20, the classification result of the example mentioned above points is provided. As seen from the figure, for the given example the points of toeboards are well classified (see gray points in Fig. 8.20). However, there are still many misclassifications for the points of tubes (see black points in Fig. 8.20). Especially at connections between tubes and toeboards, the points are misclassified as those of toeboards. A conceivable reason for such misclassified points is due to the ambiguity of determination of the principal axes for the points at the joint of two linear objects, and the SLRF of LSSHOT may be sensitive to the size of the support region chosen under this condition.

Table 8.9: Number of points in classification results.

facade	Toeboard	Tubes	Scatters	Total
I	49443	31044	11713	92200
III	25598	4889	226	30713

A confusion matrix of this case using the ground truth as the reference is set out in Table 8.10, for the evaluation of correctness. In this matrix, the rows represent the predicted results, namely the ground truth, while the columns stand for the actual classification results. It is apparent that the points of toeboards own a classification result reaching an accuracy of over 0.84, but the classification accuracy of points belonging to tubes is only 0.70. The overall classification accuracy of the given example is 0.78. In contrast, the classification of scatters gets an accuracy of only 0.55. This is because the points belonging to scatters in our dataset contain not only the noise and outliers but also the points of ladders, strut and bracing bars, which share similar geometric shapes with tubes and toeboards and have a linear straight feature as well. Hence,

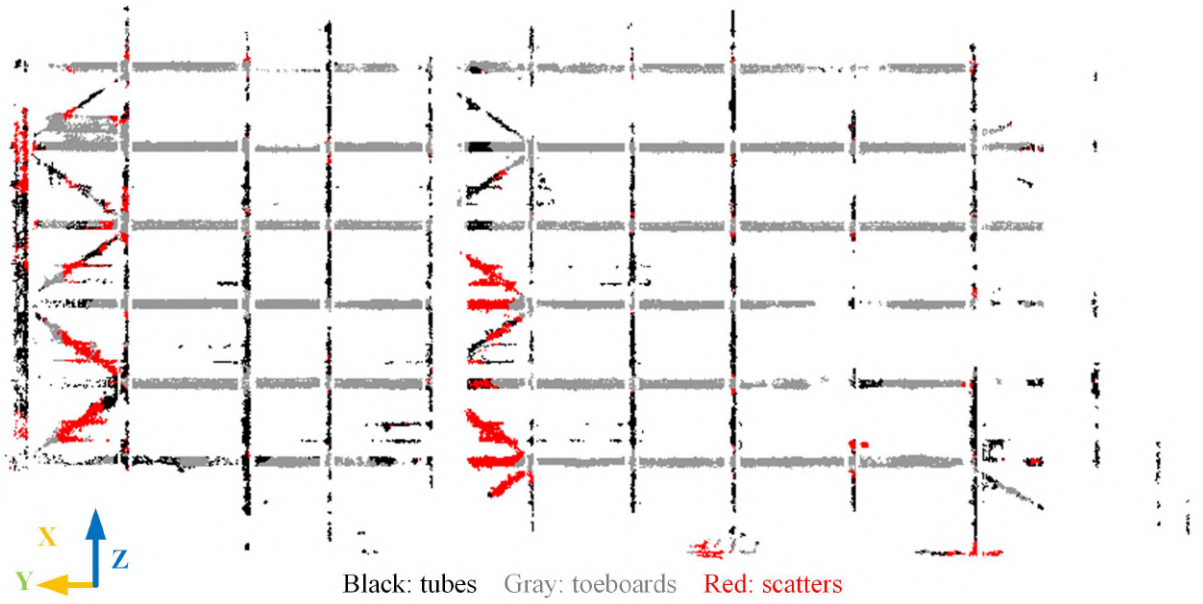


Figure 8.20: Classification of points in outer row of facade I.

when implementing the training process for the classification of scatters, it may be somehow unreliable and mixed up with points of tubes and toe boards. Moreover, due to the way of generating the point cloud, which is conducted via multi-view stereo matching, only parts of the tube can be observed, so that the points of a tube may be incomplete, leading to confusions when distinguishing points of tube or toeboard by their spatial distribution and surface curvatures.

Table 8.10: Confusion matrix of classification results for points in the outer row of facades I.

facade	Toeboards	Tubes	Scatters
Toeboards	0.84	0.09	0.07
Tubes	0.14	0.70	0.16
Scatters	0.23	0.22	0.55
Overall accuracy		0.78	

Modeling patches

In Fig. 8.21, an example of clustering outcome using region growing algorithm is displayed, which shows the classified points of toeboard components in the outer row of scaffolds of facade I. The segmented result is shown in the form of a color map, with each cluster depicted with different colors. The points of the same cluster share the same color in a surrounding neighborhood. As seen from Fig. 8.21, it is clear that most of the points representing certain objects have been divided into isotropic clusters, with irregular boundaries. Taking the result of toeboards as an example, there are also many points of other objects as well as outliers segmented into the clusters, which should be eliminated in the further modeling process.

Table 8.11 gives a statistic of all the segmented clusters for all categories in each facade. Note that only six clusters of decks are obtained from the point cloud in facade III. One explanation for this phenomenon is still due to the missing points in facade III, in which only around five thousand points are extracted for the reconstruction of decks, with a discontinuous distribution

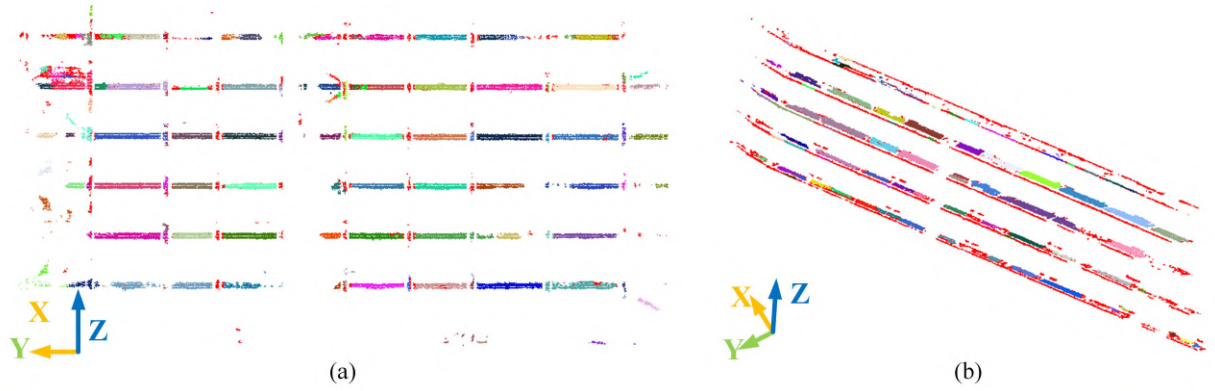


Figure 8.21: Clustering of points in the outer row of facades I. (a) Toeboards. (b) Decks.

and contaminated by a large percentage of outliers and noise, so that the clustering process is affected and can hardly achieve a good result.

Table 8.11: Number of segmented clusters.

Facade	Toeboards	Tubes	Decks	Total
I	108	89	78	275
III	86	63	6	155

The reconstructed small patches are shown in Fig. 8.22, while in Table 8.12 the statistical result of these reconstructed patches is given. As seen from Fig. 8.22, the green, and red rectangular patches stand for the toeboards and decks, and the black cylindrical patches are linked to the tubes. Apparently, these patches are discontinuous, and the sizes and orientations of them vary, revealing possible errors and biases, which require an optimization process to integrate and adjust these patches into a whole and regularized object.

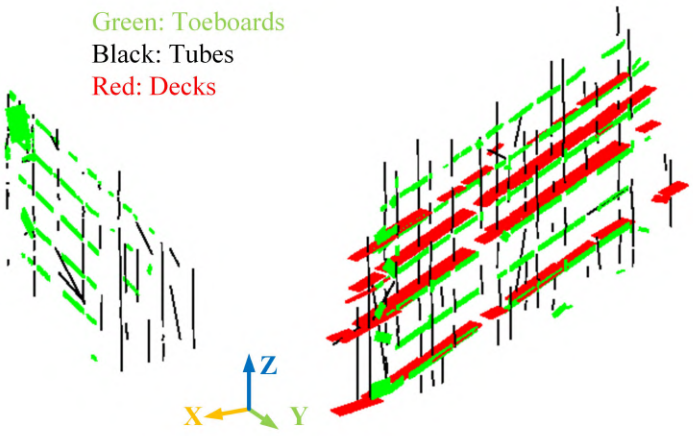


Figure 8.22: Reconstruction results of small patches.

It is interesting that for the tubes the number of reconstructed patches is larger than the corresponding number of clusters, whereas for the toeboards, these two numbers are almost the same, and for decks, the number of reconstructed patches is smaller than the corresponding

Table 8.12: Number of reconstructed patches.

Facade	Toeboards	Tubes	Decks	Total
I	107	155	48	310
III	86	65	0	151

number of clusters. This difference comes from the ways of modeling cylindrical and "board-like" objects. As for cylindrical objects, namely tubes, we utilize the RANSAC based algorithm to firstly fit the axis of point clusters and then create the cylindrical shape. Thus, it is possible to generate two or more axes in one point cluster, and then represent this cluster with several neighboring and connected cylindrical models. With respect to the âboard-likeâ objects, the reconstruction method is directly performed on the whole cluster. If the candidate points, selected by the shape matching process with the training sample, are not numerous enough to form a concave hull or cuboid, there will be no geometric representation generated, which fails.

Refined reconstructed objects

In Fig. 8.23, we illustrate the merging results for facade I, with the patches of tubes, toeboards, and decks merged via the aforementioned normalized-cut algorithm. As shown in the figure, in a neighboring area, the patches sharing the same color stand for the ones merged into one group. Theoretically, the patches belonging to one entire object should be merged. For example, the patches of toeboards situated at the same heights in one facade of scaffolds should be merged into one complete toeboard. Nonetheless, in light of the lack of points in the center part of the point cloud shown in Fig. 8.23a, these patches are merged into several adjacent groups. A similar phenomenon also occurs with tubes and decks, which can be found in Figs 8.23b and 8.23c. The statistics of numbers of merged objects in facades I and III are provided in Table 6.

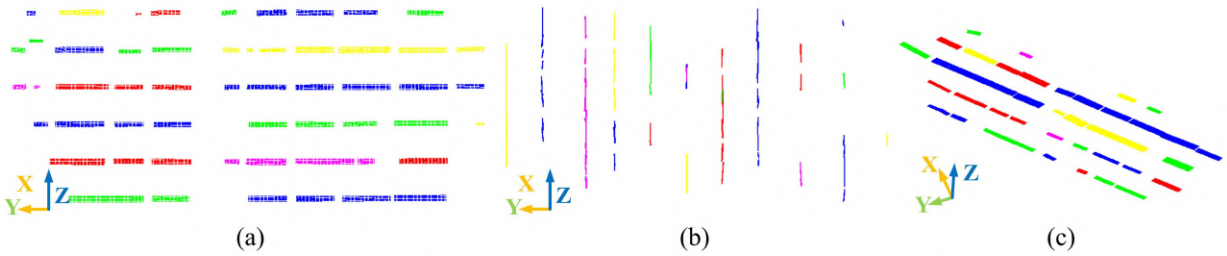


Figure 8.23: Merging results of small patches in the outer row of scaffold of facade I. (a) Toeboards. (b) Tubes. (c) Decks.

Table 8.13: Number of objects after merging process.

Facade	Toeboards	Tubes	Decks	Total
I	21	48	22	91
III	22	20	0	42

To give a clear and more detailed view of the reconstruction and refinement of objects, an example of reconstructed parts of the decks in facade I is illustrated in Fig. 8.24, in which the intermediate results of clustering, reconstruction, merging and refinement are shown in sequence. It can be observed in the figure that the original point cloud obtained from the slicing process includes not only the points of decks but also the points belonging to part of the toeboards. By

using the clustering method based on region growing, the point cloud has been cut into several separated clusters, with the points related to the toeboards influencing the reconstruction being isolated. Note that the clusters denoting deck components have various irregular shapes and blurred boundaries contaminated by outliers, which hinder the modeling process. Then, in the following reconstruction step, six small cuboid patches are generated using those clusters. These small patches have slight differences in the geometric sizes and orientations, which are possible biases when considering the reality of decks in scaffolds. Subsequently, these patches are merged into one group, with their geometric information and spatial distribution taken into consideration. After that, the merged patches are represented with one complete cuboid model to act as the final optimized reconstruction result of decks, with all the geometric parameters refined.

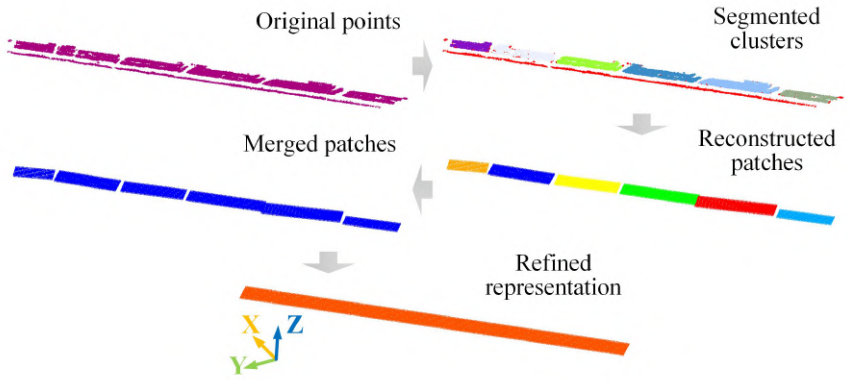


Figure 8.24: An example of reconstructing the deck components in facade I.

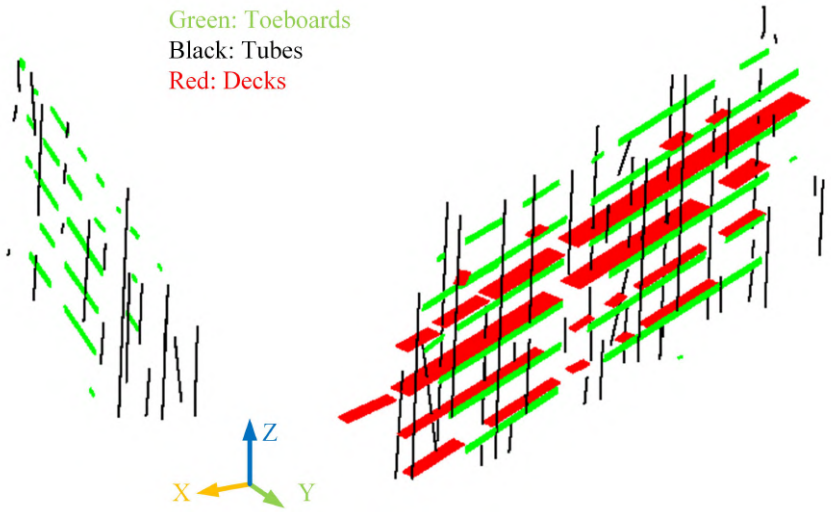


Figure 8.25: Refined results of reconstructed objects.

In Fig. 8.25, the final reconstruction result of the scaffold in facades I and III is illustrated. It is evident that a large part of scaffolds in facade I has been reconstructed, while in facade III only one side of scaffolds is partly reconstructed. For the overall reconstructed results, 43, 68, and 22 reconstructed objects representing the toeboards, tubes, and decks are obtained. It is noted that one reconstructed object after the refinement may correspond to several scaffolding elements in our case due to the merging process. For example, in the case shown in Fig. 8.24, the refined object stands for six adjacent elements in reality. However, there are still lots of scaffolding elements in

these two facades, which cannot be reconstructed. The missing points and the occlusions during the observation and generation processes are two of the most significant factors responsible for these missing components. The insufficient dataset will directly result in the failures of division and reconstruction of scaffolds from the whole point cloud of a construction site, because our division and reconstruction methods, as well as the proposed LSSHOT descriptor, are related to the spatial distribution and geometric surface of the scaffolding structures, featuring the vertical and horizontal axes and various curvature surfaces. On the other hand, the occlusions can give rise to the discontinuity and imperfection of the points of objects, which will mainly increase the difficulties of modeling the objects. Moreover, the outliers and noise existing in the point cloud will affect the final reconstruction results, which show up as geometric errors of these reconstructed objects.

Evaluations

To evaluate the performance of the proposed workflow, the completeness and geometric correctness of the reconstructed objects are assessed following the criteria depicted in evaluation metric part. In Fig. 8.26, an illustration of the reconstructed scaffolds projected to the original point cloud is given. The textures and positions of the original points are used to calculate the references. The completeness is given by statistical results about the numbers of reconstructed elements, which are shown in Tables 8.14 and 8.15. Here, the references are the real number and geometric positions of the scaffolding elements in facades I and III. It should be noted that for facades I the whole scaffolds are involved in the evaluation. Whereas, for facades III, only the outer row of scaffolds is counted.

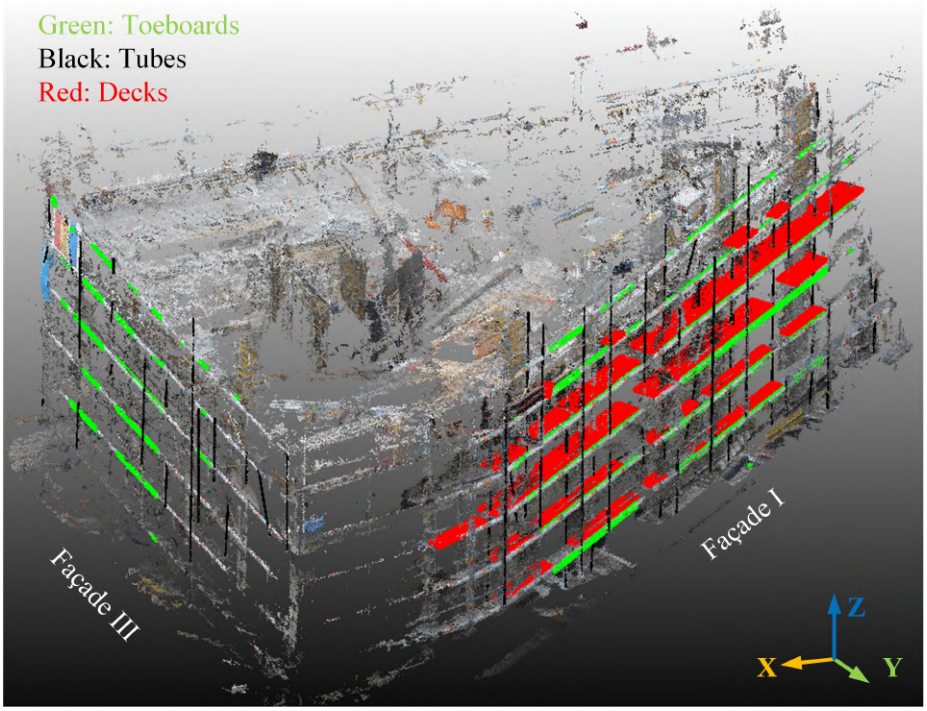


Figure 8.26: Reconstructed scaffolds projected to the original point cloud.

It is apparent from the Tables 8.14 and 8.15 that in facade I, around 86% of the toeboards, 65% of the tubes and 68% of the decks are reconstructed, whereas, for facade III, only 52% of the toeboards and approximately 50% of the tubes can be reconstructed. In facade I, more elements are reconstructed than that in facade III. Considering the results shown in Table 8.3.2,

Table 8.14: Numbers of reconstructed elements in facade I.

Facade I	Toeboards	Tubes	Decks
Reference	60	154	60
Reconstructed	52	100	41

Table 8.15: Numbers of reconstructed elements in facade III.

Facade III	Toeboards	Tubes
Reference	48	54
Reconstructed	25	27

we find that there is a strong relationship between the numbers of reconstructed elements and the number of points in divided facades. Moreover, the uneven distribution of points in different facades also contributes to the completeness or reconstruction results, resulting in many partly reconstructed elements. Furthermore, several reconstructed objects cannot correspond to any scaffolding elements, which fail in the reconstruction. As for the geometric correctness, the statistic results of the comparison between sizes, orientations, and positions of the reconstructed elements and reference ones are listed in Fig. 8.27. In these figures, the elements are assessed and counted according to different criteria (i.e., the ratio between sizes, the angle between orientations and the distance between positions). Then the reconstructed elements corresponding to different evaluating results are categorized and represented in the form of percentages. Note that the scale of the X-axis in the figure stands for a range of values rather than a single value in these figures. For example, the decks have a percentage value of 78% at the scale of 0.5 degree in Fig. 8.27b, which means that for 76% of the reconstructed decks in facade I, the angle differences range from 0 to 0.5 degree when compared with the references. Similarly, in this figure, we can also find that around 10% of the reconstructed decks in facade I have the angle differences larger than 5 degrees, which reveals inferior reconstructed orientations of these elements. Fig. 8.27 shows that more than 50% of the reconstructed elements in facade I have the size ratios ranging from 0.9 to 1.0, indicating that the sizes of these reconstructed elements are almost the same as the reference ones, namely well reconstructed in geometric sizes. About 18% of the reconstructed decks and toeboards have the size ratios of merely around 0.5, revealing that for these elements only half of their surfaces are recovered. For the orientations, more than 75% of the reconstructed elements have different angles smaller than 0.5 degrees, representing good results in orientations. In contrast, for the positions, most of the reconstructed toeboards and tubes have a distance value smaller than 7 cm. Whereas approximately 40% of the reconstructed decks show inferior results in positions, with a distance value larger than 10 cm. There are two possible explanations for the worse results of the decks. The first one is about the spatial positions of decks, which are occluded by the toeboards during the image acquisition process. Thus, the points of the decks are normally lacked and not accurate enough, which could result in errors in reconstruction. The other one is due to the manual selection of references, which may also lead to errors. Especially for the decks, they are hidden by other elements, increasing the uncertainty and difficulty when measuring geometries according to the textures.

As seen from Fig. 8.27d, most of the reconstructed elements in facade III are partly rebuilt in size with their size ratios smaller than 0.7. This can be explained by the lack of points and failure in the detection of points via projection strategy. Around 40% of the tubes are well reconstructed in size. This is because when we detect and divide the points through the vertical projection strategy, the points of tubes have commonly better projection results than that of toeboards due to their

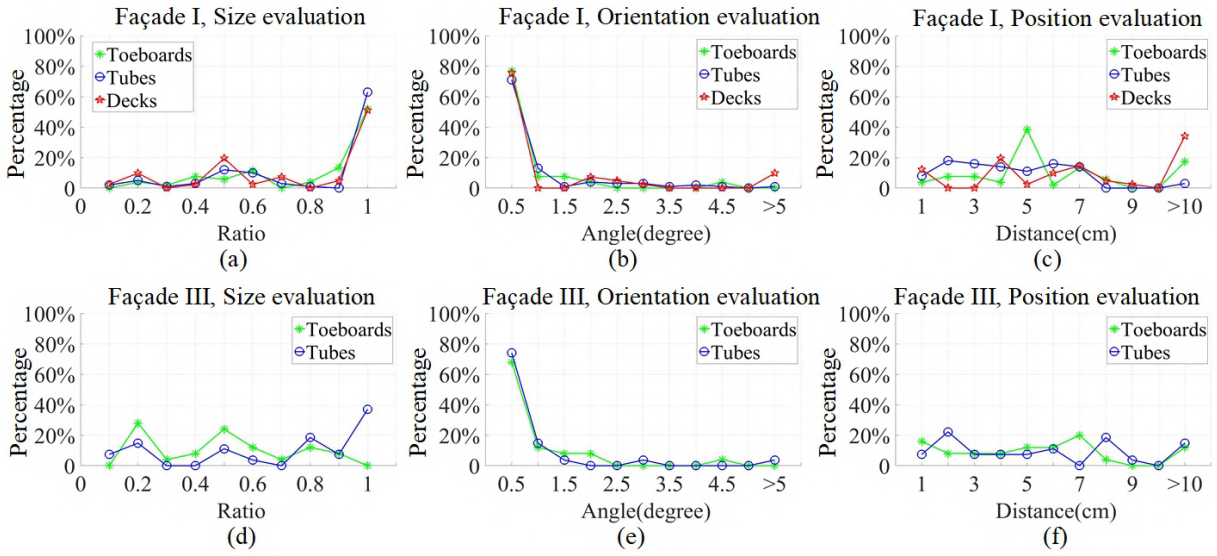


Figure 8.27: Evaluation results of reconstructed scaffolding elements. Facade I: (a) Size, (b) Orientation and (c) Position. Facade III: (d) Size, (e) Orientation and (f) Position.

structures having a denser vertical distribution. This makes them more distinguishable. Thus, the reconstruction of tubes can benefit from a higher point density in the following reconstruction process. When it comes to the orientations and positions, the reconstructed elements show similar results like that in facade I. More than 70% of the reconstructed elements have an angle difference smaller than 0.5 degrees, and approximately 80% of them have distances smaller than 8 cm. Surprisingly, nearly 20% of the elements are being reconstructed with an inferior position results with the distance greater than 10 cm. Such large errors of the positions may also be due to the lack of points. In addition, the discontinuities of the reconstructed objects in facade III may increase the geometric errors as well because this limits the ability of the merging process to adjust the reconstructed small patches.

8.4 Reconstruction of planar building objects in built environment

8.4.1 Results of semantic labeling of scenes

In semantic labeling experiments, all the algorithms used for feature extraction and initial classification are implemented via C++ and the graph-structured optimization step are performed via the GCO-v3.0 library[Boykov et al., 2001; Kolmogorov & Zabih, 2004; Boykov & Kolmogorov, 2004], both of which run on an Intel i7-6700 CPU @ 3.4GHz and with 32.0 GB RAM.

Results of using TUM datasets

When using the TUM dataset, for conducting a weakly supervised classification, we use only the area along the acissstrasse as the training set, nearly 30% of the entire dataset, while the rest of the dataset (around 70%) is used as testing set. For assessing the effectiveness of our proposed detrended features and the graph-structured optimization, we compared our method (termed as DEGO) with the method (termed as SV) using merely features from the supervoxel without the local context information, the method (termed as CX) using features from the local context information, and the method (termed as DE) using our detrended features without graph-structured

Table 8.16: Evaluation for classification results using TUM dataset with different features.

Method	SV			CX			DE		
Class	P_r	R_e	IoU	P_r	R_e	IoU	P_r	R_e	IoU
Man-made terrain	0.895	0.676	0.626	0.841	0.664	0.590	0.920	0.691	0.652
Natural terrain	0.117	0.375	0.098	0.129	0.330	0.102	0.159	0.515	0.138
High vegetation	0.343	0.797	0.316	0.305	0.451	0.222	0.415	0.794	0.375
Low vegetation	0.150	0.052	0.040	0.051	0.005	0.004	0.250	0.071	0.059
Buildings	0.897	0.742	0.684	0.822	0.754	0.648	0.912	0.826	0.765
Hard scape	0.150	0.047	0.037	0.100	0.016	0.014	0.373	0.045	0.042
Scanning artefacts	0.462	0.079	0.072	0.038	0.113	0.029	0.682	0.133	0.125
Vehicles	0.272	0.364	0.184	0.273	0.202	0.131	0.571	0.595	0.411
MEAN	0.411	0.391	0.257	0.320	0.317	0.218	0.535	0.459	0.321
OA		0.695			0.66			0.760	

optimization. The voxel size is set to 0.3 m, and the seed resolution of supervoxels is set to 1.0 m. Finally, we obtain an overall accuracy better than 0.86, for assigning eight semantic classes (i.e., man-made terrain, natural terrain, high vegetation, low vegetation, buildings, hardscape, scanning artifacts, and vehicles). The statistics of classification results of using different features are given in Table 8.16.

As seen from the table, the proposed detrended geometric features can outperform other reference methods concerning the overall accuracy, with an improvement of around 7%. For the IoU measures of all kinds of objects, our approach reveals better performance. Especially for the objects of scanning artefacts and vehicles, which are easy to be mixed up with the building facades and low vegetation, our detrended feature can achieve much better results with the IoU measures. The effectiveness of our detrended features can be backed by the analysis of features as well. In Fig. 8.28, we also provide an analysis of feature importance of different vectors in the RF process. As seen from the figure, we can find that the last three vectors representing the contextual features play a vital role in the decision trees. Besides, the vectors of detrended features from the bins 15 to 30 in the feature histogram are also essential to the creation of decision trees, with a higher averaged value of importance compared with those of the features from the supervoxel itself.

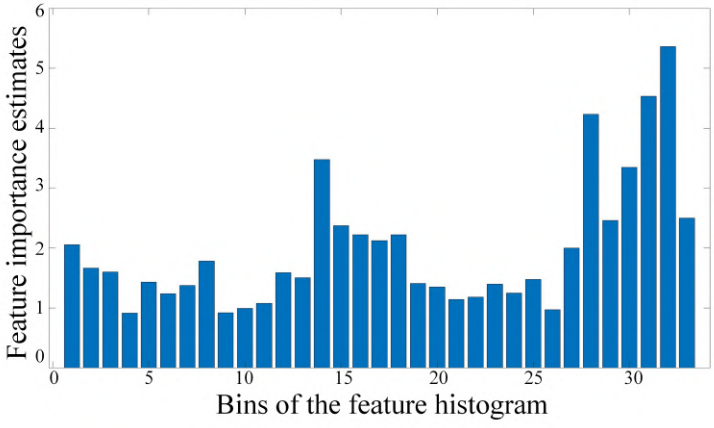


Figure 8.28: Importance of feature vectors used in RF classifier.

Table 8.17: Evaluation for classification results using TUM dataset with different Graph Cuts thresholds.

Method	DEGO, τ=0.2			DEGO, τ=0.5			DEGO, τ=0.8		
Class	P_r	R_e	IoU	P_r	R_e	IoU	P_r	R_e	IoU
Man-made terrain	0.916	0.712	0.668	0.896	0.709	0.655	0.841	0.709	0.625
Natural terrain	0.170	0.522	0.147	0.169	0.514	0.146	0.161	0.493	0.138
High vegetation	0.610	0.859	0.555	0.768	0.908	0.713	0.816	0.931	0.769
Low vegetation	0.417	0.043	0.041	0.000	0.000	0.000	0.000	0.000	0.000
Buildings	0.928	0.924	0.862	0.937	0.970	0.911	0.943	0.980	0.926
Hard scape	0.714	0.011	0.011	0.833	0.011	0.011	0.833	0.011	0.011
Scanning artefacts	0.919	0.106	0.105	0.945	0.081	0.081	0.951	0.071	0.071
Vehicles	0.703	0.667	0.520	0.697	0.544	0.440	0.697	0.264	0.237
MEAN	0.672	0.480	0.364	0.656	0.467	0.370	0.655	0.433	0.347
OA		0.835			0.866			0.870	

While the statistics of classification results of using different Graph Cuts thresholds τ are given in Table 8.17. As seen from the table, it seems that with an increasing threshold of τ, we can achieve a better overall accuracy. However, what stands out is the low vegetation, which has been totally categorized into wrong classes after the optimization process when using a large threshold for graph optimization. This is because that a large threshold τ will result in large cliques in the graphical model after the optimization and vice versa. The generation of such large cliques will force small cliques or nearby nodes to merge into a large clique so that for small objects with different initial labels, they will be wrongly optimized.

Compared with the classical point-based classification methods, one of the major advantages of the segment or primitive based classification method is that they are more insensitive to outliers and noise existing in the dataset, benefiting from the pre-clustering process. However, the use of supervoxel structures also has some drawbacks. For example, the selection of the size of the voxels is a trade-off between the suppression of noise and uneven density and the preservation of details. The larger the voxel, the more details will be smoothed. It is clear that for the conventional boundaries, like the right-angle sides of the corners formed by the smooth wall surfaces, our refined boundary supervoxels can find borders accurately. However, when it comes to the irregular edges, for example, the edges between the wall and the ground surface, due to the existence of the French windows, the boundaries found by supervoxels are biased. Besides, for small objects, like small scanning artefacts, if the size of the supervoxel is too large, they are easy to be blurred by their background so that cannot be described correctly. This can be seen from the initial result that large objects like buildings and high vegetation can always achieve a good IoU value.

We apply the trained classifier mentioned above to the entire TUM dataset, which is nearly four times larger than the training Arcisstrasse. The performance of the method is shown in Fig. 8.29. As seen from the figure, as the performance for capturing some certain objects shown in Table 8.17, a general interpretation of the scene is obtained. For example, the buildings, man-made terrain and high vegetation with high extension are well recognized.

Besides, the purposed method shows excellent potential in positioning stationary vehicles, although this can only be achieved after the optimization. In Fig. 8.30, we give a detailed view of the optimization of vehicles and buildings. The initial classification result of vehicles is questionable because in fact all the points of vehicles are occluded due to the view direction of observation. These occluded observations of vehicles make the supervoxel of cars and buses look like part of hardscapes. Only after the optimization, those wrongly labeled supervoxels can be

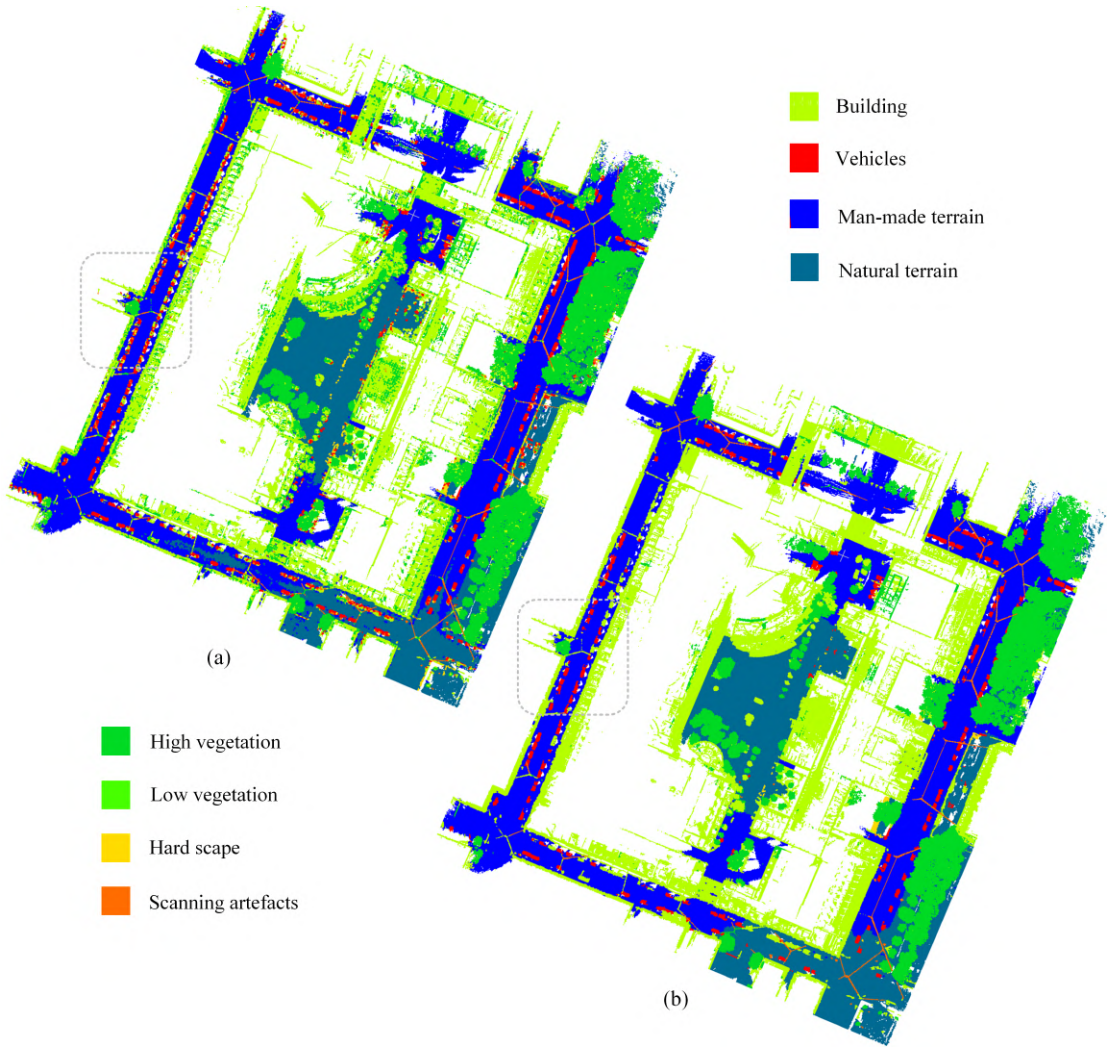


Figure 8.29: Classification results of the scene TUM City Campus. (a) Initial classification result. (b) Refined classification result.

corrected, however, if the initial label of the local area is biased, as the lower left corner of the campus scene in Fig. 8.29, the optimization may make the situation worse. In this area, large parts of the road surface are initially labeled as natural terrain, so that the optimization is failed because for nodes in the graphical model all their neighbors have wrong labels. Furthermore, there is misclassification like "sparkling effect" [Sun et al., 2018] due to the complexity of such a large scene, in which the areas inside building facade are generally incomplete and far too complex for classification. Therefore, some part of the internal structures of buildings is recognized as other objects. Although a part of the fragmented wrong labels is regularized, there remain large areas with incorrect labels.

Results of using ETH datasets

To further exploit the potential of our proposed feature extraction method, we also test our classification method on the popular Semantic 3D dataset published by ETH Zürich [Hackel et al., 2016b] for a preliminary validation. It is notable that the Semantic 3D dataset is a TLS dataset so that the density of points varies with the distance from the observation station to the objects. Here, the voxel size is set to 0.3 m, and the seed resolution of supervoxels is set to 1.0 m. In Fig. 8.31, we provide testing results of testing scenes consisting of main buildings, man-made

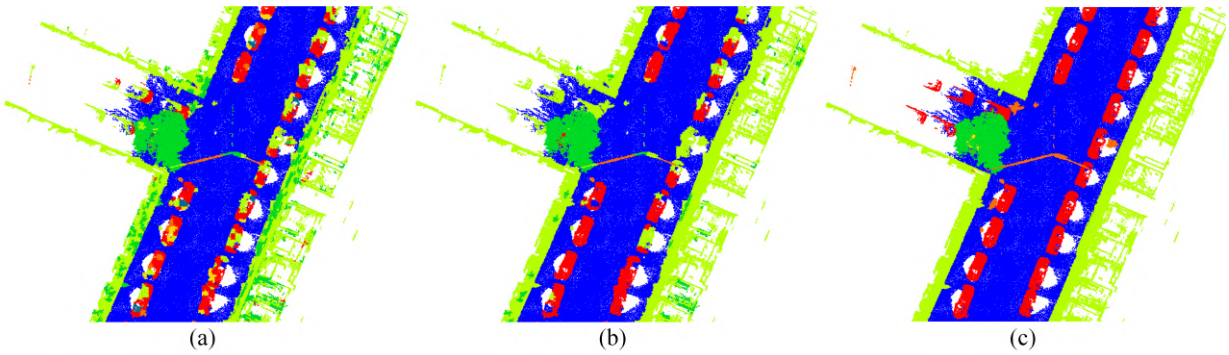


Figure 8.30: Comparison of TUM results before and after the optimization (the area of dash line boxes in Fig. 8.29). (a) Initial classification result, (b) optimized classification result, and (c) ground truth.

terrain, and high vegetations. As seen from the figure, it is clear that for our major concerns (i.e., buildings, roads, and trees), the results of our proposed method still reveals the promising potential of our feature extraction method. It is obvious that our optimization works well for the points of buildings and high vegetation, but for the area with significant changes of densities, the man-made terrain is easy to be recognized as the natural one.

In Tables 8.18 and 8.19, we show a evaluation result of using this two datasets. In the experiments of Bildstein scene, the scan we used for training is Bildstein 3,5. For evaluation, the scan of Bildstein 1 is used, involving all the eight classes of objects. As seen from the table, it is apparent that our proposed method can still achieve a good result, with the overall accuracy reaching 0.811. It is noteworthy that in this experiments it is failed to classify the objects of vehicles, low vegetation, and hardscapes. One of the possible explanations is the limited number of training samples of these kinds of objects. This is also a common problem for all the segment based weakly supervised classification methods, namely, for objects without sufficient training samples (e.g., hardscapes and vehicles), the labeling is always failed. In contrast, for objects with sufficient training samples, for example, buildings and man-made terrain (see Fig. 8.32b), satisfying results can be achieved.

Table 8.18: Evaluation for classification results using Bildstein dataset.

Method	DE			DEGO, τ=0.5		
Class	P_r	R_e	IoU	P_r	R_e	IoU
Man-made terrain	0.861	0.655	0.593	0.850	0.644	0.578
Natural terrain	0.627	0.875	0.576	0.632	0.943	0.609
High vegetation	0.759	0.983	0.749	0.883	0.999	0.882
Low vegetation	0.050	0.030	0.019	1.000	0.015	0.015
Buildings	0.804	0.770	0.648	0.803	0.949	0.770
Hard scape	0.838	0.067	0.066	1.000	0.001	0.001
Scanning artefacts	0.941	0.314	0.308	0.962	0.245	0.243
Vehicles	0.083	0.016	0.014	0.000	0.000	0.000
MEAN	0.620	0.464	0.371	0.766	0.474	0.387
OA		0.754			0.811	

In the experiments of the Untermaederbrunnen scene, the scan we used for training is Untermaederbrunnen 1. For evaluation, the scan of Untermaederbrunnen 3 is used, involving all the eight classes of objects. As seen from the table, our proposed method can achieve an overall

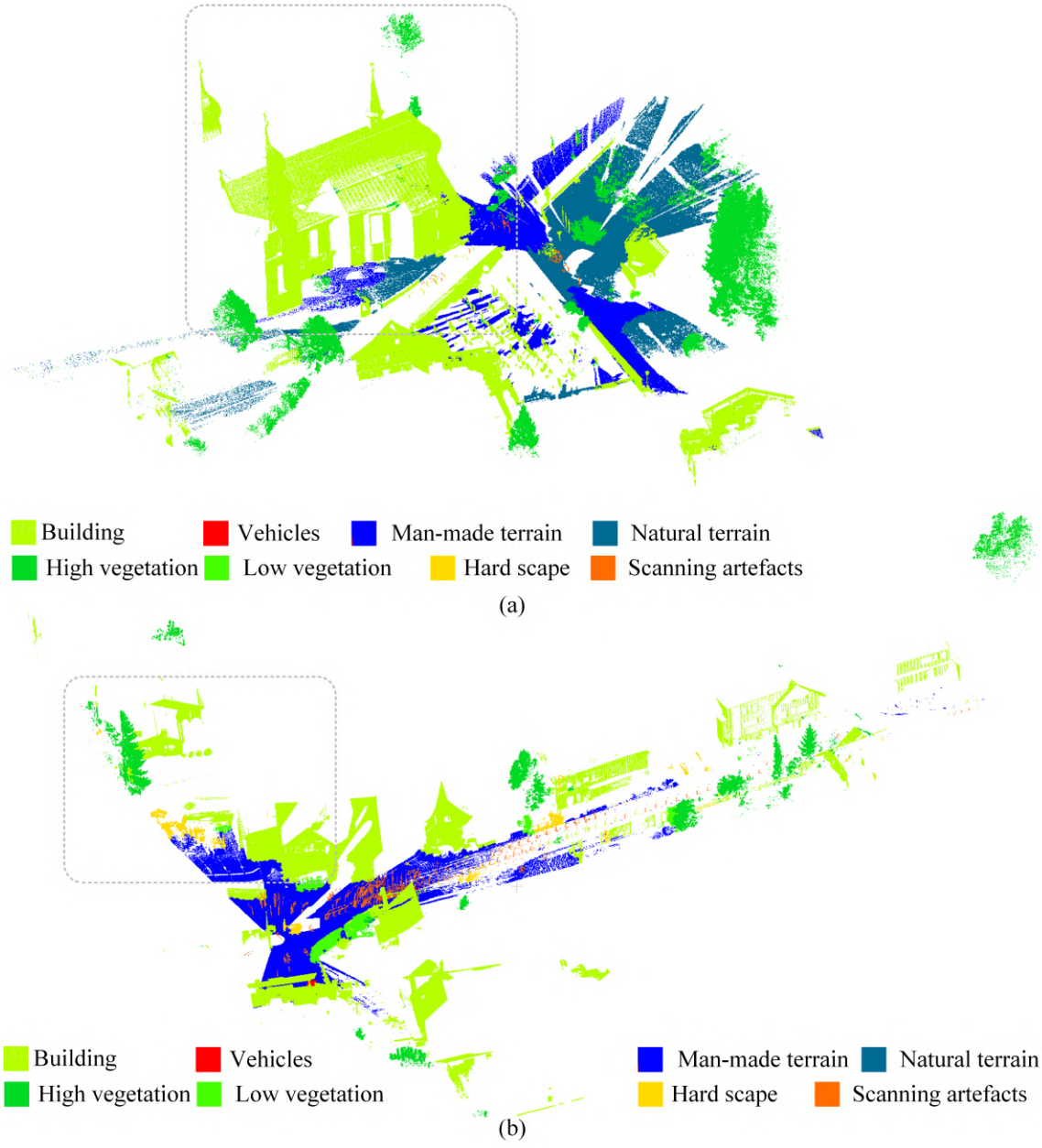


Figure 8.31: Classification results of scenes (a) Bildstein and (b) Untermaederbrunnen.

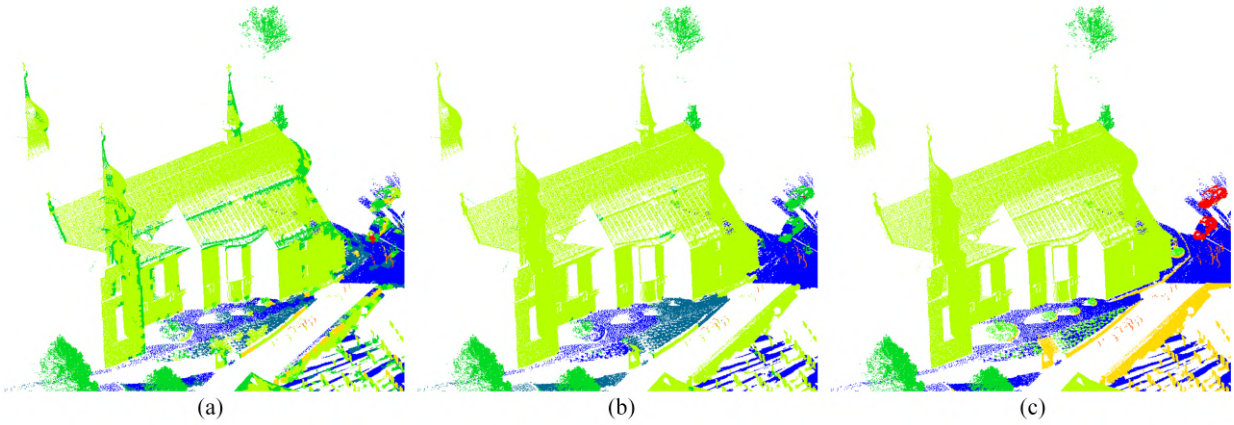


Figure 8.32: Comparison of Bildstein results before and after the optimization (the area of dash line box in Fig. 8.31a). (a) Initial classification result, (b) optimized classification result, and (c) ground truth.

accuracy of 0.804. Similar to the result of the previous scene, buildings, man-made terrain, high vegetation (see Fig. 8.33) can obtain a good overall accuracy, but for the natural terrain and vehicles both the initial labeling and optimization failed due to the insufficient training samples. It is noted that for vehicles, TLS laser scanning may cause more serious occlusions so that the scanning points of vehicles have confusing geometry like that of hardscapes.

Table 8.19: Evaluation for classification results using Untermaederbrunnen dataset.

Method		DE			DEGO, τ=0.5	
Class	P_r	R_e	IoU	P_r	R_e	IoU
Man-made terrain	0.805	0.966	0.783	0.750	0.979	0.738
Natural terrain	0.667	0.045	0.044	0.500	0.003	0.003
High vegetation	0.743	0.702	0.565	0.909	0.854	0.787
Low vegetation	0.374	0.397	0.239	0.532	0.361	0.274
Buildings	0.799	0.891	0.728	0.830	0.979	0.815
Hard scape	0.400	0.389	0.246	0.452	0.266	0.201
Scanning artefacts	0.886	0.607	0.563	0.942	0.722	0.692
Vehicles	0.412	0.253	0.186	0.889	0.096	0.095
MEAN	0.636	0.531	0.419	0.725	0.533	0.451
OA		0.747			0.804	

It is true that the result of using the ETH dataset cannot compare with the state of the art methods which have higher overall accuracy reaching 0.9, but considering that our proposed method is weakly supervised and requires much less training dataset, the results are satisfying. Besides, for the object of our primary concern (i.e., buildings), both our proposed detrended features and graph-structured optimization perform well, and the voxel-based data structure significantly accelerates the processing speed. For practical applications, LiDAR points are always textured with reliable intensity or RGB color, the performance of our method can be further improved by this radiometric information like in the work of [Sun et al., 2018]. Moreover, in our proposed method, we only use the RF classifier for obtaining the initial labels, but in fact, any classifier providing soft labels can be used in our methods. A good initial labeling result can significantly improve the effectiveness of graph-structured optimization. In the future work, a better initial labeling method could be utilized.

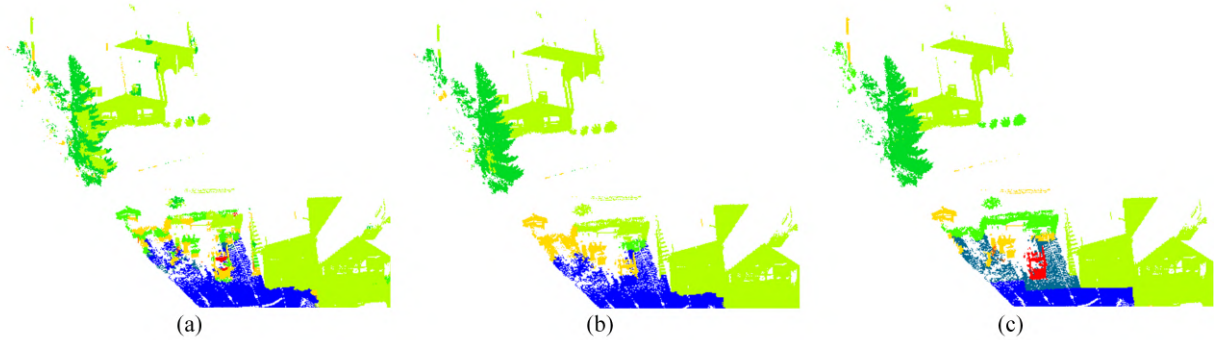


Figure 8.33: Comparison of Untermaederbrunnen results before and after the optimization (the area of dash line box in Fig. 8.31b). (a) Initial classification result, (b) optimized classification result, and (c) ground truth.

8.4.2 Results of modeling planar objects

In geometric modeling experiments, all the algorithms are implemented via C++ with the help of PCL 1.8.0, and the cell decomposition is achieved by using CGAL 4.3.0. The Graph Cuts algorithm is achieved via the GCO-v3.0 library [Boykov et al., 2001; Kolmogorov & Zabih, 2004; Boykov & Kolmogorov, 2004]. All experiments run on an Intel i7-6700 CPU @ 3.4GHz and with 32.0 GB RAM.

Point cloud segmentation

In these experiments, we first test our proposed GGBC method on the reference datasets, which has been mentioned and used in Section 7.1.3, including Samples 4 and 5. The voxel sizes used in our is set to 0.1m, and seed resolutions of supervoxels in our proposed method, LCCP, and SVGS are both set to 0.25 m. The size of the neighborhood for aggregating the supervoxels is set to 0.5m. The threshold for graph segmentation is set to 0.85. In Table 8.20, we provide the comparison of results using the local graph structure (i.e., UHGC) and the global graph structure (i.e., GGBC). As shown in the table, it is clear that the new GGBC method has comparable performance as the UHGC method, but it seems that the result of GGBC can provide better recall values, especially for the tests of using Sample 5. This reveals that the GGBC method prefers to generate under-segmented segments, but the accuracy of found boundaries of segments still needs to be improved. When clustering the global graphical model, the inner difference of a clique is less essential compared to the difference between cliques of different objects, resulting in a better ability of clustering points from the same object but having irregular shapes (e.g., bushes, flowers, and window frames). Benefiting from this advantage, GGBC is quite suitable for applying on datasets with a scene of mixture objects, namely urban scenes containing buildings, vegetation, and vehicles. In Fig. 8.34a, we illustrate the result of segmenting the MLS dataset of the area near Arcisstrasse of TUM city campus using GGBC. This dataset is contaminated with outliers and points of moving vehicles and pedestrians. As seen from the figure, major structures like facades, ground surface, vehicles, tree stems, and walls are well separated. Considering the semantic labels obtained in the semantic labeling step, we can easily extract individual building structural components from the entire scanned point cloud.

In addition, the framework of global graph-based clustering can be elegantly applied to any kind of elements from different data structures. For example, we can directly regard each voxel as a node in the graphical model without conducting the supervoxelization process. The weighted edges between nodes are determined by the affinity of geometric features of connecting nodes. In

Figure 8.34: Illustration of segmentation results using GGBC. (a) Segmentation result of the TUM scene (Arcisstrasse), (b) segmentation results of the Town Square scene (Sample 4), and (c) segmentation results of the St. Gallen scene (Sample 5).

Table 8.20: Comparison of segmentation results using local and global graphical models.

Test scenes	GGBC			UHGC		
	P_r	R_e	F_1	P_r	R_e	F_1
Sample4 Ref1	0.7004	0.7563	0.7273	0.8769	0.6352	0.7367
Sample4 Ref2	0.7134	0.6571	0.6841	0.7381	0.5988	0.6612
Sample5 Ref1	0.5911	0.8369	0.6928	0.6840	0.6847	0.6843
Sample5 Ref2	0.5841	0.8126	0.6796	0.6904	0.6510	0.6701

Fig. 8.35, we display an example of using voxels as basic elements under the GGBC framework. Compared with our former voxel-based method, the optimization at the global scale can provide a good ability of clustering points of objects with irregular shapes, and at meanwhile, preserve the boundaries of objects well. This phenomenon can be clearly observed from the comparison of segmented flowers and fences on the balcony shown in Figs. 8.35b and 8.35c. For our former voxel-based method, the flowers can only be clustered as a set of small cliques of points without completeness. In contrast, when utilizing the voxel-based GGBC method, all the flower clusters can be separated neatly.

Plane extraction

Based on the segmentation result, we conduct the extraction of planes from these segments. In this step, the smoothness and the curvature thresholds are both set to 0.01. Whereas the threshold of RANSAC fitting is set to 0.1m. In Table 8.21, we give the numbers of extracted

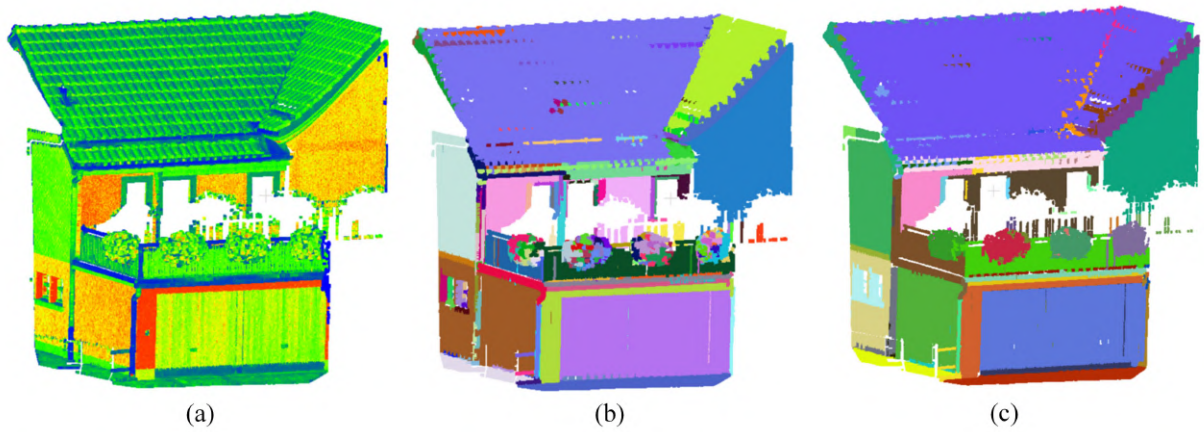


Figure 8.35: Comparison of segmentation results using GGBC. (a) Original point clouds, (b) segmentation results using method from [Xu et al., 2018d], and (c) segmentation results using voxel-based GGBC.

planar segments for the aforementioned three testing datasets, reporting that finally there are 10, 27, and 17 planes extracted from the above-mentioned testing scenes, respectively.

Table 8.21: Number of extracted planes.

Scenes	TUM	St. Gallen	Town square
Points	1886937	3336659	2359831
Supervoxels	17559	14537	21016
Segments	1057	371	308
Initial planes	10	48	20
Merged planes	10	27	17

In Fig. 8.36, the illustration of extracted planes is given. As seen from the table and the figure, we can find that the major planar structures of the scene are extracted, and small planar segments have been merged into a large complete surface. However, the merging step is a double-blade sword, which can optimize the completeness of the extracted plane and at the meanwhile, the merging is counterproductive to the separation of co-planar objects, for example, the facades of two houses in Fig. 8.36c are merged as one large facade. In addition, when using RANSAC to refine the planar segment, the fitting threshold is crucial to the number of extracted planes. In some cases (e.g., the left facade in Fig. 8.36b), multiple overlapped planes may be generated from points of the same planar object.

Plane modeling

Once the planes of the major structures are extracted, the contour hulls are extracted by the use of the alpha-shape algorithm. Here, the alpha value is set to 0.1m for the datasets of scenes Town Square and St. Gallen, while for the dataset of the scene Arcissstrasse, the alpha value is set to 0.2m. In Fig. 8.37, we illustrate the extracted contour hulls of planes in these three scenes. As shown in the figure, the extracted contour hulls have covered salient boundaries of the planar object. However, for the area with unevenly distributed densities of points, especially in the scene Arcisstrasse, the data of which is measured by MLS, the extracted hulls follow the patterns formed by the scanning lines instead of the borders of the object.

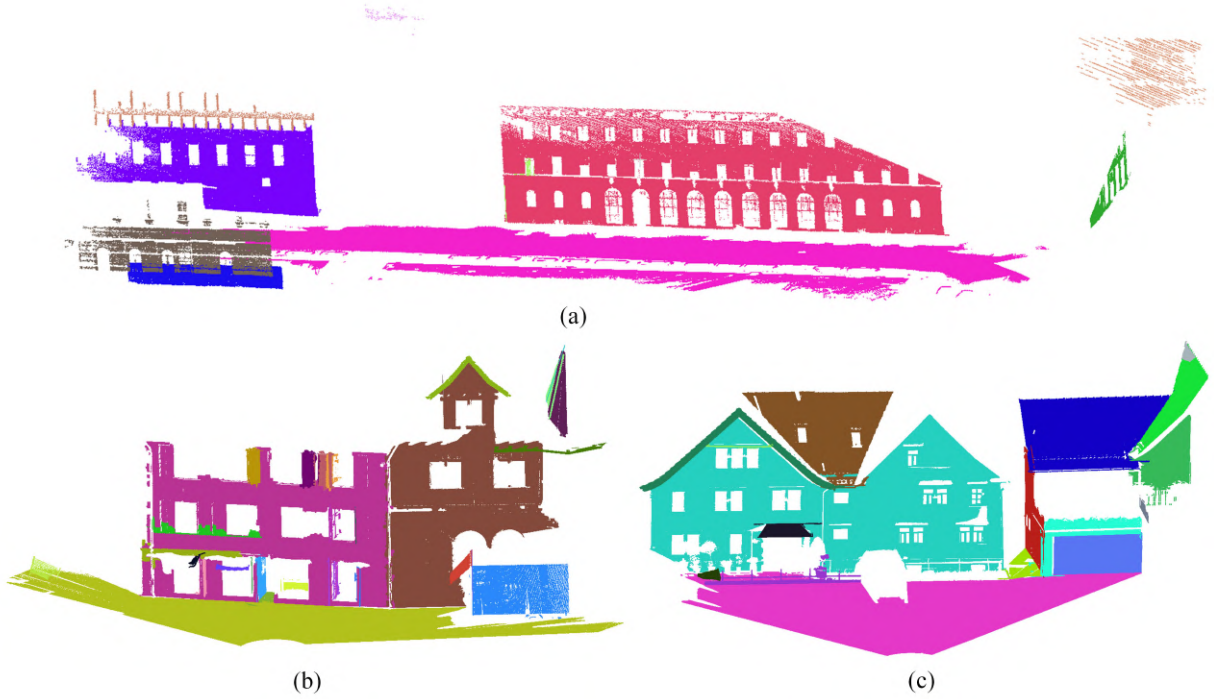


Figure 8.36: Illustration of extracted planes. (a) Extracted planes of the TUM scene (Arcisstrasse), (b) extracted planes of the Town Square scene (Sample 4), and (c) extracted planes of the St. Gallen scene (Sample 5).

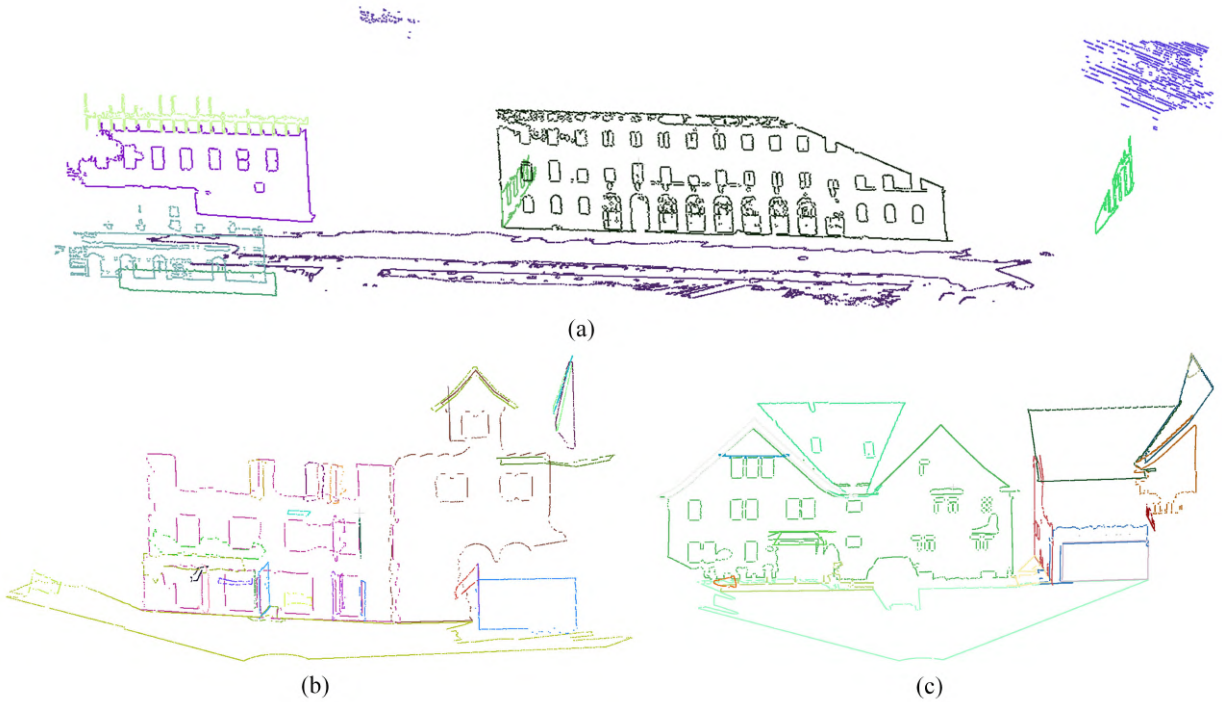


Figure 8.37: Illustration of extracted hulls. (a) Extracted hulls of the TUM scene (Arcisstrasse), (b) extracted hulls of the Town Square scene (Sample 4), and (c) extracted hulls of the St. Gallen scene (Sample 5).

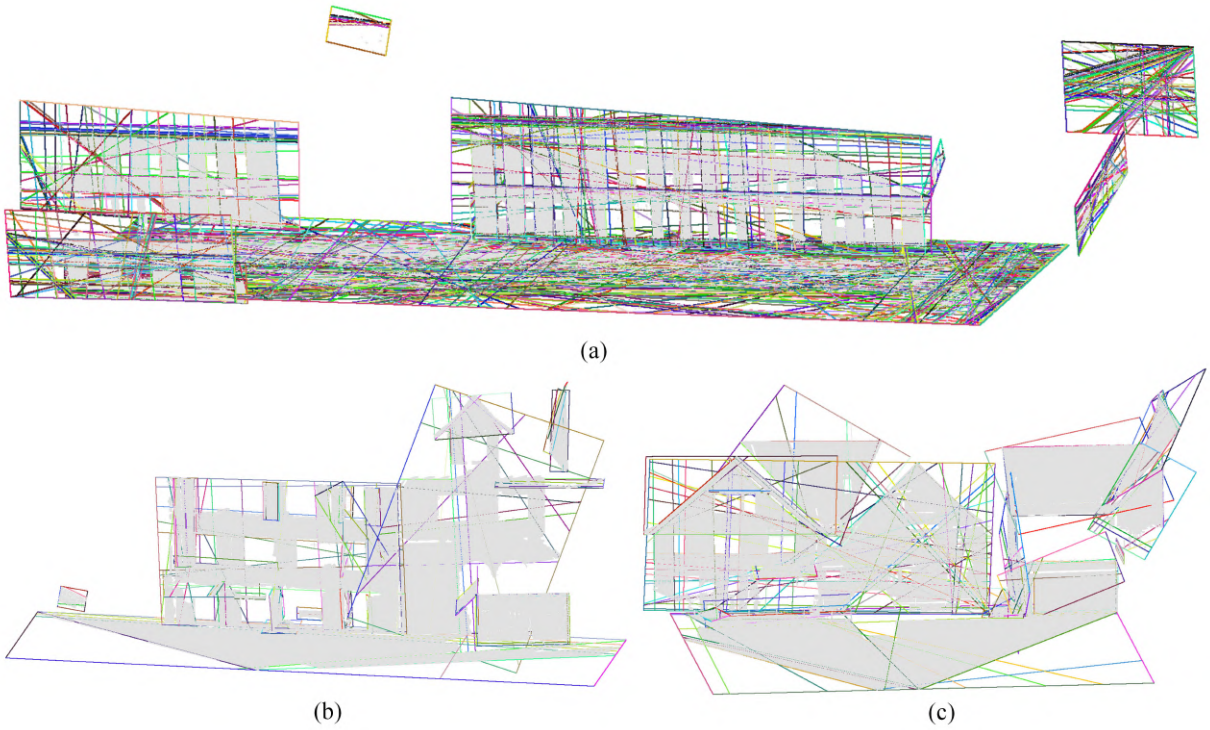


Figure 8.38: Illustration of cell decomposition results. (a) Cell decomposition of the TUM scene (Arcisstrasse), (b) cell decomposition of the Town Square scene (Sample 4), and (c) cell decomposition of the St. Gallen scene (Sample 5).

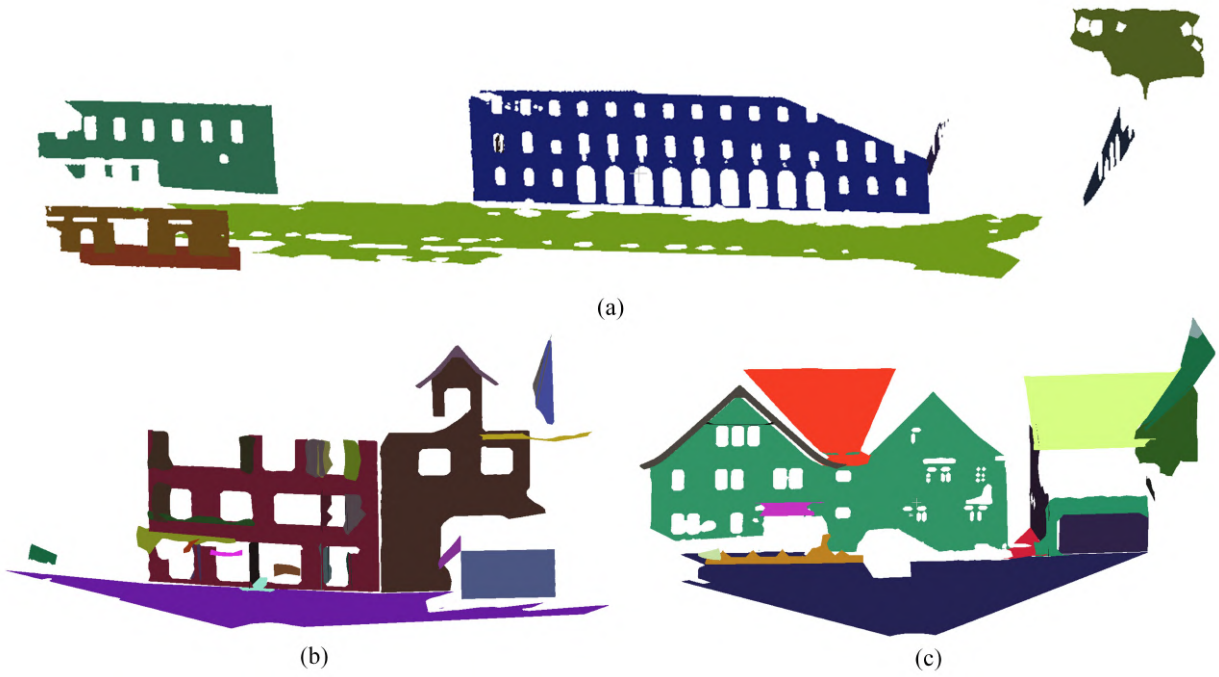


Figure 8.39: Illustration of generated models. (a) Mesh models of the TUM scene (Arcisstrasse), (b) mesh models of the Town Square scene (Sample 4), and (c) mesh models of the St. Gallen scene (Sample 5).

Table 8.22: Statistic result of cell decomposition.

Scenes	TUM	St. Gallen	Town square
Faces	33265	394	1362
Edges	67229	907	2893
Vertices	33984	577	1565

Based on the extracted hulls of each plane, linear primitives can be extracted and then refined. With these extracted linear primitives, we can achieve the cell decomposition for defining boundaries of the planes. In Fig. 8.38, an illustration of the cell decomposition results of three testing scenes. It can be seen from the figure that the plane having complex edges, for instance, the ground surface of the Arcisstrasse scene, will generate a large number of cells consisting of faces, edges, and vertices. As the statistic results given in Table 8.22, we can find that the number of faces (i.e., the cells) generated from the Arcisstrasse scene is much more than those from other scenes, and many of them are just tiny cells, whose outlines form the complex edges of the plane. Finally, in Fig. 8.39, we display the reconstructed models of the planar elements based on the edges. In this display, the ground surface is still kept but can be removed easily according to the semantic labels of corresponding points. It is also noted that the quality of the reconstructed models is directly influenced by the quality of the extracted contour hulls. All the details of the facades can only be preserved on condition that the contour hulls extract sufficient points. For example, in the Town Square scene, windows of the facade are well reconstructed due to the dense points of the extracted contour hulls. In contrast, for the facade in the St. Gallen scenes, the shapes of reconstructed windows are biased with round corners, and in another facade even no windows modeled. Regarding the quality of the reconstructed models, it is indeed that there is still a large room for improvement when compared with the state of the art modeling algorithm and strategies. Nevertheless, considering that all the major planar building elements have been recognized and extracted, with their boundaries extracted and vectorized. It would not be a challenging task for further refinement of the representation of models. This would also be one of our further work in the future.

9 Conclusions and Outlook

The purpose of this last chapter is two-fold - to illustrate the most important conclusions produced by the work underlying this paper, and to propose new directions for further research based on the limitations associated with the proposed method. The conclusions are drawn according to their relationship to the specific goals pursued in the thesis and the four research questions presented in Section 1.3.

9.1 Conclusions

Framework of reconstructing objects in construction sites and urban scenes

For the question of a general strategy and framework of reconstructing objects from construction sites and urban scenes, in this work, we have actually provided two frameworks for reconstructing linear structural elements in the construction site and planar building objects in the built environment, respectively. Our proposed framework for reconstructing linear structural elements (taking scaffolds as target example) can successfully detect and distinguish the points belonging to scaffolds and building structures from the entire point cloud of the construction site with a complex environment. The reconstruction of geometric representations for scaffolding elements is accomplished through our proposed method as well. These achievements indicate a promising prospect for the reconstruction of as-built BIM, using photogrammetric point clouds. By the use of involved methods and algorithms, the points belonging to these objects are correctly identified and then reconstructed with the cylinder and cuboid models. To be specific, in the results of one facade of the building, around 65% of the scaffolding components are reconstructed. Being limited to missing points resulting from the number of input images, for the other facade, only approximately 50% of the scaffolding components are reconstructed, but it is noticeable that parts of the failed reconstructions can be attributed to the lack of points generated through the SfM and dense matching process rather than our method itself. Thus, for the dataset with enough points, our method is still qualified and promising. As a conclusion, all these experimental results reveal a high potential for the as-built BIM reconstruction, especially for the identification of useful points and the reconstruction of objects. Our proposed framework for reconstructing planar building objects can also successfully label, segment, and reconstruct points belonging to planar building objects from the complex built environment. The entire framework consists of the semantic labeling of point clouds using context-based features and global graph-based optimization and the modeling of planar building elements using global graph clustering and cell decomposition. The first phase is designed for classifying MLS point clouds by the use of supervised classification. Here, the detrended geometric features extracted from the points of the supervoxel-based local context can be regarded as an improvement of the conventional multi-scale supervoxel solution. However, the combination of feature vectors from the detrending and local context is not well handled. An unsupervised clustering like latent Dirichlet allocation [Zhang et al., 2016] should be adopted to remove redundant information. A regularization processing using global graph optimization is also applied to improve the quality of the classification result. Experiments using the

complex urban scene approve the performance of proposed method tuning with different features is analyzed. For the major testing dataset (i.e., TUM city campus), we can achieve improvement with the overall accuracy of 0.05 when using the detrending. Meanwhile, regularization processing contributes to another improvement of 0.1 for the overall accuracy. The second phase is a bottom-up reconstruction method that utilizes global graph-based optimization and boundary representation with cell decomposition, enabling an automatic and unsupervised segmentation of point clouds. Here, the key step is the planarity-based extraction of planar segments. The modeling is accomplished by a cell decomposition method getting the polygon representation of extracted planes. However, the generalization of modeling should be further improved for irregularly shaped objects. According to experiments, the testing dataset can automatic and well segmented into the planar objects and then reconstructed with surface models. As a conclusion, our proposed workflow can be competent to the task of building objects reconstruction. However, the generalization of modeling should be further improved for irregularly shaped objects.

Organizing unstructured point clouds with optimized data structures

For the question of organizing unstructured point clouds with optimized data structures, we have applied voxel-based data structure (i.e., voxels, supervoxels, boundary refined supervoxels, and supervoxel local context) for both segmentation and classification. Especially, we contribute unsupervised voxel-based point cloud segmentation method under a hierarchical clustering framework, combining merits of supervoxel structure and probabilistic formulation encapsulating Gestalt principles, which is designed to automatically and adaptively partition the 3D scene. We test our method with LiDAR point clouds of different scenes and compare its results with those of representative reference methods. The qualitative and quantitative results reveal that our approach can outperform some representative algorithms (e.g., RG) as well as our former method (e.g., SVGS) for our datasets under a complicated situation of urban scenes, with the overall F_1 measure better than 0.66. Testing results indicate that our proposed hierarchical clustering framework for point cloud segmentation can keep the right balance between effectiveness and efficiency. Besides, we have proved that all the voxel-based solutions have high potentials in suppressing outliers and against the unevenly distributed points. However, the selection of the optimized voxel and supervoxel sizes, namely the granularity of the entire voxel and superstructures, is crucial to the quality of final segments. We investigated the influence of using different voxel sizes. Theoretically, the larger the voxels and supervoxels are, the more details of the scene will be blurred in the segmentation result. However, one has to find a trade-off between accuracy and efficiency. This is also a common problem for the majority of the voxel-based segmentation methods. In the general case, the sizes of voxels and supervoxels are identified empirically, mainly depending on the requirement within the accuracy of segments and the data quality. The selection of voxel sizes should be designed as an adaptive step in our future work.

Robust and discriminative features for both segmentation and classification tasks

For the question of designing features for both segmentation and classification tasks, we developed features for both point-based data structure and voxel-based structure. Specifically, for the feature of the point-based data structure, we proposed a novel 3D local feature descriptor LSSHOT, which is designed for the feature extraction of points belonging to objects with elongated objects, also shows satisfactory performance, which can be used for other applications concerning point cloud processing, such as structure segmentation and indoor scene recognition. The fundamental principle of our proposed descriptor is to sacrifice the generality for the robustness. However, since the current descriptor requires the estimation of normal vectors as accumulating information, the outliers and noises existing in the point cloud still affect its performance, which limits

the use in critical conditions. It would be interesting to remove those outliers and noise with some advanced algorithms before applying the descriptor. Based on a series of experiments on simulated data, the proposed descriptor outperforms the original SHOT descriptors in our cases, giving more accurate and reliable results in shape matching tests. For the classification of points using our descriptor, the accuracy of results for two essential components (toeboards and tubes) reaches more than 70% in our examples. For the feature of the voxel-based data structure, we proposed a strategy combining voxel structure, perceptual grouping laws, and supervoxel local context. The eigenvalue-based feature calculates the unary feature for points within the voxel or supervoxel. Here, an idea of using detrended geometric features extracted from the points of the supervoxel-based local context is also developed, which reveals significant advantages compared with conventional methods using multi-scale strategy. While the binary feature describing the relation between voxels and supervoxels is estimated by the geometric cues considering perceptual grouping laws. By mimicking the human-vision system, the use of perceptual grouping laws provides us with reasonable weights when we evaluate the consistency between two voxels or supervoxels. Experiments also validate the superior performance of the proposed methods using perceptual grouping laws, especially for segmenting complex scenes and nonplanar surfaces of objects. The use of multiple perceptual grouping laws provides a purely geometric solution for the segmentation task, avoiding the use of color or intensity information of points, which are usually limited by the data collection techniques and lighting conditions.

Post-processing and refinement of segmentation and classification results

For the question of approaches for post-processing and refinement of segmentation and classification results, we developed the graph-based optimization method for both segmentation and refinement of classification. To be specific, both local graphical model and global graphical model are utilized for encoding the geometry and topology of the 3D scene, which has a natural relation to each other. For the segmentation task, we combined the voxel structure with the graphical model, so that the segmentation task has been converted to the construction and partition of the graph. The use of graphical models can help us utilize local and global information rather than pairwise information between elements, which could be more reliable when judging whether two elements should be connected or not. The segmentation experiments also confirmed the feasibility of using the graphical model. However, during the evaluation, although the rule for manual segmentation is fixed, namely each segment should correspond to a semantic object of the building component, personal preferences of people will affect the reliability of ground truth when they manually segment the dataset. In some recent work [Vo et al., 2015], multiple reference datasets are generated, and each of them is segmented independently by point cloud processing experts. Moreover, the graphical model we used currently can still not fully reflect the topology of the real 3D space. Some recent work using the Riemannian graph with Geodetic distance to organize the voxel structure has gained impressive results [Li, 2018]. For the task of semantic labeling, we used the graphical model for the refinement of initial labels. The labeling task can be naturally represented in terms of energy minimization [Szeliski et al., 2008]. We minimize the energy function constructed with a data term and piecewise smoothness term via a graph cut algorithm, i.e., a-expansion proposed in [Boykov et al., 2001], which finds a good approximate solution by iteratively running min-cut/max-flow algorithms on an appropriate graph. This move making algorithm iteratively selects a label and considers moves which increase the set of pixels that are given this label if the movement has lower energy. Besides, the constructed energy function can be justified in the context of maximum a posteriori estimation of Markov Random Fields. The discrete multilabel MRF are solved by apply min-cut/maxflow iteratively to binary-labeled piecewise MRF [Lempitsky & Zisserman, 2010]. At the moment, our result of using the ETH dataset cannot compare with the state of the art methods with overall accuracy reaching 0.9, but for the

object of our primary concern (i.e., buildings), both our proposed detrended features and graph-structured optimization perform well, and the voxel-based data structure significantly accelerates the processing speed. In our proposed method, we only use the RF classifier for obtaining the initial labels, but in fact, any classifier providing soft labels can be used in our methods. With better initial labeling result, the optimization can significantly improve the quality of final labels.

9.2 Outlook

This thesis has succeeded in its aim of recognizing and reconstructing building objects from the construction site and built environment, but naturally, there is still room for improvement. In the future, the following tasks would be further studied and investigated:

The first task is to achieve the classification and the semantic interpretation of points with more efficient algorithms and methods. The performance should be further improved. Our primary purpose (i.e., building structures) will also be taken into consideration. Currently, plenty of researches using deep learning based algorithm for the semantic labeling in the 3D space have been reported and drawn increasing attention. The geometric features estimated by deep neural networks have shown significant advantages compared with the hand-crafted features with better distinctiveness. The use of some newly developed neural networks (i.e., Generative Adversarial Network) can also help us to overcome the drawbacks of the acquisition of datasets by inpainting the missing points caused by occlusion.

Besides, the voxel-based data structure should be further consummated. The selection of voxel size should be designed as an adaptive step. In contrast, the sizes of the voxel and supervoxel are identified empirically in our current work, which largely depends on the demand for the accuracy of the segments and the data quality. Moreover, the boundary refined voxel or supervoxel based data structure could be another research focus. With a deformable shape, the generated voxels can provide better accuracy of the boundaries of segments, avoiding the "zig-zag" effect at the edge of the segment. During the supervoxelization of points, not only the local contextual information but also the global information should be taken into consideration. This can be achieved by optimizing a combined local- and global-graphical model of the entire 3D scene.

At last, the geometric modeling of the object should be further studied, especially in the case that the data is acquired in a partly occluded observation. The surface model should have comparable accuracy to the manually generated ones and have the ability to compensate for the missing area resulting from the occlusion. The regularization of the boundaries should be further developed with constraint based on prior knowledge. Besides, the improvement should not be limited to the accuracy of the reconstructed surface models, but also the possibility of linking the recognized object to the volumetric representation, which is closer to the real demand of the as-built BIM reconstruction.

Bibliography

Adams R, Bischof L (1994) Seeded region growing. IEEE Transactions on Pattern Analysis and Machine Intelligence, 16 (6): 641–647.

Aijazi A, Checchin P, Trassoudaine L (2013) Segmentation based classification of 3D urban point clouds: A super-voxel based approach with evaluation. Remote Sensing, 5 (4): 1624–1650.

Aijazi AK, Checchin P, Trassoudaine L (2014) Super-voxel based segmentation and classification of 3D urban landscapes with evaluation and comparison. Berlin, Heidelberg: Springer Berlin Heidelberg.

Aldoma A, Marton ZC, Tombari F, Wohlkinger W, Potthast C, Zeisl B, Rusu R, Gedikli S, Vincze M (2012) Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation. IEEE Robotics & Automation Magazine, 19 (3): 80–91.

Aljumaily H, Laefer DF, Cuadra D (2017) Urban point cloud mining based on density clustering and mapreduce. Journal of Computing in Civil Engineering, 31 (5): 04017021.

Alpert S, Galun M, Brandt A, Basri R (2012) Image segmentation by probabilistic bottom-up aggregation and cue integration. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34 (2): 315–327.

Arya S, Mount DM, Netanyahu NS, Silverman R, Wu AY (1998) An optimal algorithm for approximate nearest neighbor searching fixed dimensions. Journal of the ACM, 45 (6): 891–923.

Awrangjeb M, Fraser C (2014) Automatic segmentation of raw LiDAR data for extraction of building roofs. Remote Sensing, 6 (5): 3716–3751.

Awrangjeb M, Fraser CS (2013) Rule-based segmentation of LIDAR point cloud for automatic extraction of building roof planes. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, 2 (3/W1): 1–6.

Babahajiani P, Fan L, Kämäräinen JK, Gabbouj M (2017) Urban 3D segmentation and modelling from street view images and LiDAR point clouds. Machine Vision and Applications, 28 (7): 679–694.

Besl P, Jain R (1988) Segmentation through variable-order surface fitting. IEEE Transactions on Pattern Analysis and Machine Intelligence, 10 (2): 167–192.

Bosché F (2010) Automated recognition of 3D CAD model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction. Advanced Engineering Informatics, 24 (1): 107–118.

Boykov Y, Kolmogorov V (2004) An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26 (9): 1124–1137.

Boykov Y, Veksler O, Zabih R (2001) Fast approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (11): 1222–1239.

Breiman L (1996) Bagging predictors. Machine Learning, 24 (2): 123–140.

Breiman L (2001) Random forests. Machine Learning, 45 (1): 5–32.

Cabaleiro M, Riveiro B, Arias P, Caamaño J, Vilán J (2014) Automatic 3D modelling of metal frame connections from LiDAR data for structural engineering purposes. ISPRS Journal of Photogrammetry and Remote Sensing, 96: 47–56.

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

Chen D, Zhang L, Li J, Liu R (2012) Urban building roof segmentation from airborne lidar point clouds. International Journal of Remote Sensing, 33 (20): 6497–6515.

Chen D, Zhang L, Mathiopoulos PT, Huang X (2014) A methodology for automated segmentation and reconstruction of urban 3-D buildings from ALS point clouds. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 7 (10): 4199–4217.

Chen H, Bhanu B (2007) 3D free-form object recognition in range images using local surface patches. Pattern Recognition Letters, 28 (10): 1252–1262.

Cheng Y (1995) Mean shift, mode seeking, and clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17 (8): 790–799.

Cour T, Benezit F, Shi J (2005) Spectral segmentation with multiscale graph decomposition. In: IEEE Conference on Computer Vision and Pattern Recognition, 2: 1124–1131.

Dong W, Lan J, Liang S, Yao W, Zhan Z (2017) Selection of LiDAR geometric features with adaptive neighborhood size for urban land cover classification. International Journal of Applied Earth Observation and Geoinformation, 60: 99–110.

Dutta A, Engels J, Hahn M (2014) A distance-weighted Graph-Cut method for the segmentation of laser point clouds. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XL-3 (3): 81–88.

Edelsbrunner H, Kirkpatrick D, Seidel R (1983) On the shape of a set of points in the plane. IEEE Transactions on Information Theory, 29 (4): 551–559.

Erdős G, Nakano T, Horváth G, Nonaka Y, Váncza J (2015) Recognition of complex engineering objects from large-scale point clouds. CIRP Annals, 64 (1): 165–168.

Everingham M, Van Gool L, Williams CK, Winn J, Zisserman A (2010) The pascal visual object classes (voc) challenge. International Journal of Computer Vision, 88 (2): 303–338.

Felzenszwalb PF, Huttenlocher DP (2004) Efficient graph-based image segmentation. International Journal of Computer Vision, 59 (2): 167–181.

Filin S, Pfeifer N (2005) Neighborhood systems for airborne laser data. Photogrammetric Engineering & Remote Sensing, 71 (6): 743–755.

Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24 (6): 381–395.

Frome A, Huber D, Kolluri R, Bï¿½ï¿½low T, Malik J (2004) Recognizing objects in range data using regional point descriptors. In: Computer Vision – ECCV 2004 (pp. 224–237). Springer Berlin Heidelberg.

Funkhouser A, Golovinsky A (2009) Min-cut based segmentation of point clouds. In: Proc. Int. Conf. Workshop Comput. Vision: 39–46.

Gehrung J, Hebel M, Arens M, Stilla U (2017) An approach to extract moving objects from MLS data using a volumetric background representation. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, IV-1/W1: 107–114.

Golovinskiy A, Funk T (2009) Min-cut based segmentation of point clouds. In: IEEE International Conference on Computer Vision Workshops, ICCV Workshops: 39–46.

Golovinskiy A, Kim VG, Funkhouser T (2009) Shape-based recognition of 3D point clouds in urban environments. In: Computer Vision, 2009 IEEE 12th International Conference on: 2154–2161.

Green WR, Grobler H (2015) Normal distribution transform graph-based point cloud segmentation. In: Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech): 54–59.

Grilli E, Menna F, Remondino F (2017) A review of point clouds segmentation and classification algorithms. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII-2/W3: 339–344.

Guinard S, Landrieu L (2017) Weakly supervised segmentation-aided classification of urban scenes from 3D LiDAR point clouds. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII-1/W1: 151–157.

Guo L, Chehata N, Mallet C, Boukir S (2011) Relevance of airborne lidar and multispectral image data for urban scene classification using Random Forests. ISPRS Journal of Photogrammetry and Remote Sensing, 66 (1): 56–66.

Guo Y, Bennamoun M, Sohel F, Lu M, Wan J (2014) 3D object recognition in cluttered scenes with local surface features: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36 (11): 2270–2287.

Guo Y, Bennamoun M, Sohel F, Lu M, Wan J, Kwok NM (2015) A comprehensive performance evaluation of 3D local feature descriptors. International Journal of Computer Vision, 116 (1): 66–89.

Hackel T, Savinov N, Ladicky L, Wegner JD, Schindler K, Pollefeys M (2017) Semantic3D. net: A new Large-scale Point Cloud Classification Benchmark. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, IV-1/W1: 91–98.

Hackel T, Wegner JD, Schindler K (2016a) Contour detection in unstructured 3D point clouds. In: IEEE Conference on Computer Vision and Pattern Recognition: 1610–1618.

Hackel T, Wegner JD, Schindler K (2016b) Fast semantic segmentation of 3D point clouds with strongly varying density. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, III-3: 177–184.

Hagen L, Kahng AB (1992) New spectral methods for ratio cut partitioning and clustering. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 11 (9): 1074–1085.

Hane C, Zach C, Cohen A, Angst R, Pollefeys M (2013) Joint 3D scene reconstruction and class segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 97–104.

Hao W, Wang Y (2016) Structure-based object detection from scene point clouds. Neurocomputing, 191: 148–160.

Hirschmuller H (2008) Stereo processing by semiglobal matching and mutual information. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30 (2): 328–341.

Huang H, Brenner C, Sester M (2013) A generative statistical approach to automatic 3D building roof reconstruction from laser scanning data. ISPRS Journal of Photogrammetry and Remote Sensing, 79: 29–43.

Huber D, Akinci B, Oliver AA, Anil E, Okorn BE, Xiong X (2011) Methods for automatically modeling and representing as-built building information models. In: Proceedings of the NSF CMMI Research Innovation Conference

Ioannou Y, Taati B, Harrap R, Greenspan M (2012) Difference of normals as a multi-scale operator in unorganized point clouds. In: International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission: 501–508.

Johnson A, Hebert M (1999) Using spin images for efficient object recognition in cluttered 3D scenes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21 (5): 433–449.

Johnson EL, Mehrotra A, Nemhauser GL (1993) Min-cut clustering. Mathematical programming, 62 (1-3): 133–151.

Jung J, Hong S, Jeong S, Kim S, Cho H, Hong S, Heo J (2014) Productive modeling for development of as-built BIM of existing indoor structures. Automation in Construction, 42: 68–77.

Jutzi B, Gross H (2009) Nearest neighbour classification on laser point clouds to gain object structures from buildings. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 38 (Part 1): 4–7.

Kada M (2007) Scale-dependent simplification of 3D building models based on cell decomposition and primitive instancing. In: International Conference on Spatial Information Theory: 222–237.

Kada M, McKinley L (2009) 3D building reconstruction from LiDAR based on a cell decomposition approach. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 38 (Part 3): W4.

Kada M, Wichmann A (2013) Feature-driven 3D building modeling using planar halfspaces. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, 3: W3.

Kahler O, Reid I (2013) Efficient 3d scene labeling using fields of trees. In: Proceedings of the IEEE International Conference on Computer Vision: 3064–3071.

Kim Bs, Kohli P, Savarese S (2013) 3D scene understanding by voxel-CRF. In: Proceedings of the IEEE International Conference on Computer Vision: 1425–1432.

Kolbe TH (2009) Representing and exchanging 3D city models with CityGML. In: 3D geo-information sciences (pp. 15–31). Springer.

Kolmogorov V, Zabih R (2004) What energy functions can be minimized via graph cuts? IEEE Transactions on Pattern Analysis and Machine Intelligence, 26 (2): 147–159.

Laefer DF, Truong-Hong L (2017) Toward automatic generation of 3D steel structures for building information modelling. Automation in Construction, 74: 66–77.

Lafarge F, Mallet C (2012) Creating large-scale city models from 3D-point clouds: a robust approach with hybrid representation. International Journal of Computer Vision, 99 (1): 69–85.

Landrieu L, Raguet H, Vallet B, Mallet C, Weinmann M (2017) A structured regularization framework for spatially smoothing semantic labelings of 3D point clouds. ISPRS Journal of Photogrammetry and Remote Sensing, 132: 102–118.

Landrieu L, Simonovsky M (2018) Large-Scale Point Cloud Semantic Segmentation With Superpoint Graphs. In: IEEE Conference on Computer Vision and Pattern Recognition

Lari Z, Habib A (2014) An adaptive approach for the segmentation and extraction of planar and linear/cylindrical features from laser scanning data. ISPRS Journal of Photogrammetry and Remote Sensing, 93: 192–212.

Lee I, Schenk T (2002) Perceptual organization of 3D surface points. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 34 (3/A): 193–198.

Lee J, Son H, Kim C, Kim C (2013) Skeleton-based 3D reconstruction of as-built pipelines from laser-scan data. Automation in Construction, 35: 199–207.

Lehtomäki M, Jaakkola A, Hyyppä J, Lampinen J, Kaartinen H, Kukko A, Puttonen E, Hyyppä H (2016) Object classification and recognition from mobile laser scanning point clouds in a road environment. IEEE Transactions on Geoscience and Remote Sensing, 54 (2): 1226–1239.

Lempitsky V, Zisserman A (2010) Learning to count objects in images. In: Advances in neural information processing systems: 1324–1332.

Li L, Su F, Yang F, Zhu H, Li D, Zuo X, Li F, Liu Y, Ying S (2018) Reconstruction of Three-Dimensional (3D) Indoor Interiors with Multiple Stories via Comprehensive Segmentation. Remote Sensing, 10 (8): 1281.

Li M (2018) A super voxel-based riemannian graph for multi scale segmentation of lidar point clouds. ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences, 4 (3).

Li M, Nan L, Smith N, Wonka P (2016) Reconstructing building mass models from UAV images. Computers & Graphics, 54: 84–93.

Li Z, Zhang L, Mathiopoulos PT, Liu F, Zhang L, Li S, Liu H (2017) A hierarchical methodology for urban facade parsing from TLS point clouds. ISPRS Journal of Photogrammetry and Remote Sensing, 123: 75–93.

Lim EH, Suter D (2009) 3D terrestrial LIDAR classifications with super-voxels and multi-scale conditional random fields. Computer-Aided Design, 41 (10): 701–710.

Lin H, Gao J, Zhou Y, Lu G, Ye M, Zhang C, Liu L, Yang R (2013) Semantic decomposition and reconstruction of residential scenes from LiDAR data. ACM Transactions on Graphics, 32 (4): 66.

Linsen L, Prautzsch H (2001) Local versus global triangulations. In: Eurographics 2001 - Short Presentations

Liu Y, Xiong Y (2008) Automatic segmentation of unorganized noisy point clouds based on the Gaussian map. Computer-Aided Design, 40 (5): 576–594.

Liu YJ, Zhang JB, Hou JC, Ren JC, Tang WQ (2013) Cylinder detection in large-scale point cloud of pipeline plant. IEEE Transactions on Visualization and Computer Graphics, 19 (10): 1700–1707.

Lodha SK, Fitzpatrick DM, Helmbold DP (2007) Aerial lidar data classification using adaboost. In: Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007): 435–442.

Lodha SK, Kreps EJ, Helmbold DP, Fitzpatrick D (2006) Aerial LiDAR Data Classification Using Support Vector Machines (SVM). In: Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06): 567–574.

Lu X, Yao J, Tu J, Li K, Li L, Liu Y (2016) Pairwise linkage for point cloud segmentation. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, III-3: 201–208.

Maalek R, Lichti DD, Ruwanpura JY (2018) Robust Segmentation of Planar and Linear Features of Terrestrial Laser Scanner Point Clouds Acquired from Construction Sites. Sensors, 18 (3): 819.

Maas HG (1999) The potential of height texture measures for the segmentation of airborne laserscanner data. In: Fourth international airborne remote sensing conference and exhibition/21st Canadian symposium on remote sensing, 1: 154–161.

Mahmoudabadi H, Olsen MJ, Todorovic S (2016) Efficient terrestrial laser scan segmentation exploiting data structure. ISPRS Journal of Photogrammetry and Remote Sensing, 119: 135–150.

Marshall D, Lukacs G, Martin R (2001) Robust segmentation of primitives from range data in the presence of geometric degeneracy. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (3): 304–314.

Mesolongitis A, Stamos I (2012) Detection of windows in point clouds of urban scenes. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference On: 17–24.

Mian A, Bennamoun M, Owens R (2006) Three-dimensional model-based object recognition and segmentation in cluttered scenes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28 (10): 1584–1601.

Michaelsen E, Stilla U, Soergel U, Doktorski L (2010) Extraction of building polygons from SAR images: Grouping and decision-level in the GESTALT system. Pattern Recognition Letters, 31 (10): 1071–1076.

Mikolajczyk K, Schmid C (2005) A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27 (10): 1615–1630.

Morsdorf F, Meier E, Allgöwer B, Nüesch D (2003) Clustering in airborne laser scanning raw data for segmentation of single trees. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 34 (part 3): W13.

Muja M, Lowe DG (2009) Fast approximate nearest neighbors with automatic algorithm configuration. VISAPP (1), 2 (331-340): 2.

Nahangi M, Czerniawski T, Haas CT (2015) Automated 3D Shape Detection and Outlier Removal in Cluttered Laser Scans of Industrial Assemblies. In: Proceedings of the International Conference of Innovation in Construction: 0–10.

Nan L, Sharf A, Xie K, Wong TT, Deussen O, Cohen-Or D, Chen B (2011) Conjoining gestalt rules for abstraction of architectural drawings, volume 30. ACM.

Nan L, Wonka P (2017) Polyfit: Polygonal surface reconstruction from point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Venice, Italy: 2353–2361.

Niemeyer J, Rottensteiner F, Soergel U (2014) Contextual classification of lidar data and building object detection in urban areas. ISPRS Journal of Photogrammetry and Remote Sensing, 87: 152–165.

Nurunnabi A, Belton D, West G (2012) Robust segmentation for multiple planar surface extraction in laser scanning 3D point cloud data. In: Pattern Recognition (ICPR), 2012 21st International Conference on: 1367–1370.

Nurunnabi A, West G, Belton D (2015) Outlier detection and robust normal-curvature estimation in mobile laser scanning 3D point cloud data. Pattern Recognition, 48 (4): 1404–1419.

Ochmann S, Vock R, Wessel R, Klein R (2016) Automatic reconstruction of parametric building models from indoor point clouds. Computers & Graphics, 54: 94–103.

Oesau S, Lafarge F, Alliez P (2014) Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut. ISPRS Journal of Photogrammetry and Remote Sensing, 90: 68–82.

Okorn B, Xiong X, Akinci B, Huber D (2010) Toward automated modeling of floor plans. In: Proceedings of the symposium on 3D data processing, visualization and transmission, 2.

Pal M (2005) Random forest classifier for remote sensing classification. International Journal of Remote Sensing, 26 (1): 217–222.

Papon J, Abramov A, Schoeler M, Worgotter F (2013) Voxel cloud connectivity segmentation-supervoxels for point clouds. In: IEEE Conference on Computer Vision and Pattern Recognition: 2027–2034.

Pătrăucean V, Armeni I, Nahangi M, Yeung J, Brilakis I, Haas C (2015) State of research in automatic as-built modelling. Advanced Engineering Informatics, 29 (2): 162–171.

Peng B, Zhang L, Zhang D (2013) A survey of graph theoretical approaches to image segmentation. Pattern Recognition, 46 (3): 1020–1038.

Pham TT, Eich M, Reid I, Wyeth G (2016a) Geometrically consistent plane extraction for dense indoor 3D maps segmentation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS): 4199–4204.

Pham TT, Rezatofighi SH, Reid I, Chin TJ (2016b) Efficient point process inference for large-scale object detection. In: IEEE Conference on Computer Vision and Pattern Recognition

Plaza-Leiva V, Gomez-Ruiz J, Mandow A, García-Cerezo A (2017) Voxel-based neighborhood for spatial shape pattern classification of lidar point clouds with supervised learning. Sensors, 17 (3): 594.

Polewski P, Yao W, Heurich M, Krzystek P, Stilla U (2015) Detection of fallen trees in ALS point clouds using a Normalized Cut approach trained by simulation. ISPRS Journal of Photogrammetry and Remote Sensing, 105: 252–271.

Potts RB (1952) Some generalized order-disorder transformations. In: Mathematical proceedings of the cambridge philosophical society, 48 (1): 106–109.

Poullis C (2013) A framework for automatic modeling from pointcloud data. IEEE Transactions on Pattern Analysis and Machine Intelligence, : 1.

Previtali M, Díaz-Vilariño L, Scaioni M (2018) Indoor Building Reconstruction from Occluded Point Clouds Using Graph-Cut and Ray-Tracing. Applied Sciences, 8 (9): 1529.

Pu S, Vosselman G (2009) Knowledge based reconstruction of building models from terrestrial laser scanning data. ISPRS Journal of Photogrammetry and Remote Sensing, 64 (6): 575–584.

Rabbani T, Van Den Heuvel F, Vosselmann G (2006) Segmentation of point clouds using smoothness constraint. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 36 (5): 248–253.

Ramiya AM, Nidamanuri RR, Ramakrishnan K (2016) A supervoxel-based spectro-spatial approach for 3D urban point cloud labelling. International Journal of Remote Sensing, 37 (17): 4172–4200.

Richtsfeld A, M?rwald T, Prankl J, Zillich M, Vincze M (2014) Learning of perceptual grouping for object segmentation on RGB-D data. Journal of Visual Communication and Image Representation, 25 (1): 64–73.

Rothermel M, Wenzel K, Fritsch D, Haala N (2012) Sure: Photogrammetric surface reconstruction from imagery. In: Proceedings LC3D Workshop, Berlin, 8.

Rusu RB (2010) Semantic 3d object maps for everyday manipulation in human living environments. KI-Künstliche Intelligenz, 24 (4): 345–348.

Rusu RB, Blodow N, Marton ZC, Beetz M (2009a) Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in domestic environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems: 1–6.

Rusu RB, Cousins S (2011) 3D is here: Point cloud library (pcl). In: IEEE International Conference on Robotics and Automation: 1–4.

Rusu RB, Holzbach A, Blodow N, Beetz M (2009b) Fast geometric point labeling using conditional random fields. In: IEEE/RSJ International Conference on Intelligent Robots and Systems: 7–12.

Rusu RB, Marton ZC, Blodow N, Dolha M, Beetz M (2008) Towards 3D point cloud based object maps for household environments. Robotics and Autonomous Systems, 56 (11): 927–941.

Rusu RB, Marton ZC, Blodow N, Holzbach A, Beetz M (2009c) Model-based and learned semantic object labeling in 3D point cloud maps of kitchen environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems: 3601–3608.

Salti S, Tombari F, Di Stefano L (2014) Shot: Unique signatures of histograms for surface and texture description. Computer Vision and Image Understanding, 125: 251–264.

Sampath A, Shan J (2010) Segmentation and reconstruction of polyhedral building roofs from aerial lidar point clouds. IEEE Transactions on Geoscience and Remote Sensing, 48 (3): 1554–1567.

Sanchez V, Zakhor A (2012) Planar 3D modeling of building interiors from point cloud data. In: Image Processing (ICIP), 2012 19th IEEE International Conference on: 1777–1780.

Sarkar S, Boyer K (1993) Perceptual organization in computer vision: A review and a proposal for a classificatory structure. IEEE Transactions on Systems, Man, and Cybernetics, 23 (2): 382–399.

Schnabel R, Wahl R, Klein R (2007) Efficient RANSAC for point-cloud shape detection. Computer Graphics Forum, 26 (2): 214–226.

Schuster HF (2004) Segmentation of lidar data using the tensor voting framework. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 35 (B3): 1073–1078.

Secord J, Zakhor A (2006) Tree detection in aerial lidar and image data. In: International Conference on Image Processing: 2317–2320.

Shahzad M, Zhu XX (2015) Robust reconstruction of building facades for large areas using spaceborne TomoSAR point clouds. IEEE Transactions on Geoscience and Remote Sensing, 53 (2): 752–769.

Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (8): 888–905.

Shi Y, Long P, Xu K, Huang H, Xiong Y (2016) Data-driven contextual modeling for 3d scene understanding. Computers & Graphics, 55: 55–67.

Son H, Kim C (2010) 3D structural component recognition and modeling method using color and 3D data for construction progress monitoring. Automation in Construction, 19 (7): 844–854.

Son H, Kim C, Kim C (2015) 3D reconstruction of as-built industrial instrumentation models from laser-scan data and a 3D CAD database based on prior knowledge. Automation in Construction, 49: 193–200.

Stein SC, Schoeler M, Papon J, Worgotter F (2014) Object partitioning using local convexity. In: IEEE Conference on Computer Vision and Pattern Recognition: 304–311.

Stone CJ, Feller W (1969) An introduction to probability theory and its applications: volume I. Journal of the American Statistical Association, 64 (328): 1676.

Su YT, Bethel J, Hu S (2016) Octree-based segmentation for terrestrial LiDAR point cloud data in industrial applications. ISPRS Journal of Photogrammetry and Remote Sensing, 113: 59–74.

Sun Z, Xu Y, Hoegner L, Stilla U (2018) Classification of MLS point clouds in urban scenes using detrended geometric features from supervoxel-based local context. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, IV-2: 271–278.

Szeliski R, Zabih R, Scharstein D, Veksler O, Kolmogorov V, Agarwala A, Tappen M, Rother C (2008) A comparative study of energy minimization methods for markov random fields with smoothness-based priors. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30 (6): 1068–1080.

Tang P, Huber D, Akinci B, Lipman R, Lytle A (2010) Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques. Automation in Construction, 19 (7): 829–843.

Tarsha-Kurdi F, Landes T, Grussenmeyer P et al. (2007) Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data. In: Proceedings of the ISPRS Workshop on Laser Scanning, 36: 407–412.

Tombari F, Salti S, Stefano LD (2010) Unique signatures of histograms for local surface description. In: Computer Vision – ECCV 2010 (pp. 356–369). Springer Berlin Heidelberg.

Torr P, Zisserman A (2000) MLESAC: A new robust estimator with application to estimating image geometry. Computer Vision and Image Understanding, 78 (1): 138–156.

Toussaint GT (1983) Solving geometric problems with the rotating calipers. In: Proc. IEEE Melecon, 83: A10.

Tóvári D, Pfeifer N (2005) Segmentation based robust interpolation-a new approach to laser data filtering. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 36 (3/W19): 79–84.

Truong-Hong L, Laefer DF, Hinks T, Carr H (2012) Combining an angle criterion with voxelization and the flying voxel method in reconstructing building models from LiDAR data. Computer-Aided Civil and Infrastructure Engineering, 28 (2): 112–129.

Turkan Y, Bosche F, Haas CT, Haas R (2012) Automated progress tracking using 4D schedule and 3D sensing technologies. Automation in Construction, 22: 414–421.

Tuttas S, Braun A, Borrmann A, Stilla U (2014) Comparision of photogrammetric point clouds with BIM building elements for construction progress monitoring. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XL-3 (3): 341–345.

Tuttas S, Braun A, Borrmann A, Stilla U (2017) Acquisition and consecutive registration of photogrammetric point clouds for construction progress monitoring using a 4D BIM. PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science, 85 (1): 3–15.

Tuttas S, Stilla U (2011) Window detection in sparse point clouds using indoor points. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 38: 3.

Vo AV, Truong-Hong L, Laefer DF, Bertolotto M (2015) Octree-based region growing for point cloud segmentation. ISPRS Journal of Photogrammetry and Remote Sensing, 104: 88–100.

Von Luxburg U (2007) A tutorial on spectral clustering. Statistics and computing, 17 (4): 395–416.

Vosselman G, Coenen M, Rottensteiner F (2017) Contextual segment-based classification of airborne laser scanner data. ISPRS Journal of Photogrammetry and Remote Sensing, 128: 354–371.

Vosselman G, Gorte BG, Sithole G, Rabbani T (2004) Recognising structure in laser scanner point clouds. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 46 (8): 33–38.

Vosselman G, Maas HG (2010) Airborne and terrestrial laser scanning. Whittles Publishing.

Wang C, Cho YK, Kim C (2015a) Automatic BIM component extraction from point clouds of existing buildings for sustainability applications. Automation in Construction, 56: 1–13.

Wang M, Tseng YH (2011) Incremental segmentation of lidar point clouds with an octree-structured voxel space. The Photogrammetric Record, 26 (133): 32–57.

Wang R, Xie L, Chen D (2017) Modeling indoor spaces using decomposition and reconstruction of structural elements. Photogrammetric Engineering & Remote Sensing, 83 (12): 827–841.

Wang Y, Zhang L, Mathiopoulos PT, Deng H (2015b) A Gestalt rules and graph-cut-based simplification framework for urban building models. International Journal of Applied Earth Observation and Geoinformation, 35: 247–258.

Wang Z, Zhang L, Fang T, Mathiopoulos PT, Tong X, Qu H, Xiao Z, Li F, Chen D (2015c) A multiscale and hierarchical feature extraction method for terrestrial laser scanning point cloud classification. IEEE Transactions on Geoscience and Remote Sensing, 53 (5): 2409–2425.

Wei Y, Yao W, Wu J, Schmitt M, Stilla U (2012) Adaboost-based feature relevance assessment in fusing lidar and image data for classification of trees and vehicles in urban scenes. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, 7: 323–328.

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

Wu B, Yu B, Yue W, Shu S, Tan W, Hu C, Huang Y, Wu J, Liu H (2013) A voxel-based method for automated identification and morphological parameters estimation of individual street trees from mobile laser scanning data. Remote Sensing, 5 (2): 584–611.

Wu C (2013) Towards linear-time incremental structure from motion. In: International Conference on 3D Vision: 127–134.

Xie L, Hu H, Zhu Q, Wu B, Zhang Y (2017) Hierarchical Regularization of Polygons for Photogrammetric Point Clouds of Oblique Images. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 42.

Xiong B, Elberink SO, Vosselman G (2014) A graph edit dictionary for correcting errors in roof topology graphs reconstructed from point clouds. ISPRS Journal of Photogrammetry and Remote Sensing, 93: 227–242.

Xiong X, Adan A, Akinci B, Huber D (2013) Automatic creation of semantically rich 3D building models from laser scanner data. Automation in Construction, 31: 325–337.

Xu Y, He J, Tuttas S, Stilla U (2015) Reconstruction of scaffolding components from photogrammetric point clouds of a construction site. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, : 401–408.

Xu Y, Heogner L, Tuttas S, Stilla U (2018a) A voxel- and graph-based strategy for segmenting man-made infrastructures using perceptual grouping laws: comparison and evaluation. Photogrammetric Engineering & Remote Sensing.

Xu Y, Hoegner L, Tuttas S, Stilla U (2017a) Voxel- and graph-based point cloud segmentation of 3d scenes using perceptual grouping laws. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, IV-1/W1: 43–50.

Xu Y, Sun Z, Boerner R, Koch T, Hoegner L, Stilla U (2018b) Generation of ground truth datasets for the analysis of 3D point clouds in urban scenes acquired via different sensors. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII-3: 2009–2015.

Xu Y, Tuttas S, Heogner L, Stilla U (2018c) Reconstruction of scaffolds from a photogrammetric point cloud of construction sites using a novel 3D local feature descriptor. Automation in Construction, 85.

Xu Y, Tuttas S, Hoegner L, Stilla U (2016a) Classification of photogrammetric point clouds of scaffolds for construction site monitoring using subspace clustering and PCA. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLI-B3: 725–732.

Xu Y, Tuttas S, Hoegner L, Stilla U (2017b) Geometric Primitive Extraction From Point Clouds of Construction Sites Using VGS. IEEE Geoscience and Remote Sensing Letters, 14 (3): 424–428.

Xu Y, Tuttas S, Hoegner L, Stilla U (2018d) Voxel-based segmentation of 3D point clouds from construction sites using a probabilistic connectivity model. Pattern Recognition Letters, 102: 67–74.

Xu Y, Tuttas S, Stilla U (2016b) Segmentation of 3D outdoor scenes using hierarchical clustering structure and perceptual grouping laws. In: Pattern Recogniton in Remote Sensing (PRRS), 2016 9th IAPR Workshop on: 1–6.

Xu Y, Yao W, Tuttas S, Hoegner L, Stilla U (2018e) Unsupervised segmentation of point clouds from buildings using hierarchical clustering based on gestalt principles. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, : 1–17.

Yang B, Dong Z (2013) A shape-based segmentation method for mobile laser scanning point clouds. ISPRS Journal of Photogrammetry and Remote Sensing, 81: 19–30.

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

Yang B, Dong Z, Zhao G, Dai W (2015) Hierarchical extraction of urban objects from mobile laser scanning data. ISPRS Journal of Photogrammetry and Remote Sensing, 99: 45–57.

Yao W, Hinz S, Stilla U (2009) Object extraction based on 3d-segmentation of LiDAR data by combining mean shift with normalized cuts: two examples from urban areas. In: Urban Remote Sensing Event, 2009 Joint: 1–6.

Yao W, Hinz S, Stilla U (2010) Automatic vehicle extraction from airborne LiDAR data of urban areas aided by geodesic morphology. Pattern Recognition Letters, 31 (10): 1100–1108.

Yao W, Polewski P, Krzystek P (2017) Semantic labelling of ultra dense MLS point clouds in urban road corridors based on fusing CRF with shape priors. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII-2/W7: 971–976.

Yeung J, Nahangi M, Shahtaheri Y, Haas C, Walbridge S, West J (2014) Comparison of methods used for detecting unknown structural elements in three-dimensional point clouds. In: Construction Research Congress 2014: Construction in a Global Network: 945–954.

Yu Y, Li J, Guan H, Wang C (2016) Automated detection of three-dimensional cars in mobile laser scanning point clouds using DBM-Hough-Forests. IEEE Transactions on Geoscience and Remote Sensing, 54 (7): 4130–4142.

Zhang Z, Zhang L, Tong X, Guo B, Zhang L, Xing X (2016) Discriminative-dictionary-learning-based multilevel point-cluster features for ALS point-cloud classification. IEEE Transactions on Geoscience and Remote Sensing, 54 (12): 7309–7322.

Zhu XX, Shahzad M (2014) Facade reconstruction using multiview spaceborne TomoSAR point clouds. IEEE Transactions on Geoscience and Remote Sensing, 52 (6): 3541–3552.

Zolanvari SI, Laefer DF (2016) Slicing Method for curved façade and window extraction from point clouds. ISPRS Journal of Photogrammetry and Remote Sensing, 119: 334–346.

Curriculum Vitae

Name	Yusheng Xu
Geburtstag und -ort	07.04.1989 in Henan, China
Staatsangehörigkeit	chinesisch
Postanschrift	Reutterstr.68B,80689, München, Deutschland
Phone	+ 49-1639457326
E-mail	yusheng.xu@tum.de
Familienstand	Verheiratet

Ausbildung/Tätigkeit

1995 – 2001	Grundschule in Zhengzhou, Henan, China.
2001 – 2007	Gymnasium in Zhengzhou, Henan, China. Abschluss: Allgemeine Hochschulreife
2007 – 2011	Studium der Geographische Informationswissenschaften an der Tongji Universität, Shanghai, China. Abschluss: Bachelor of Science
2011 – 2014	Studium der Geographische Informationswissenschaften an der Tongji Universität, Shanghai, China. Abschluss: Master of Engineering
2014 – 2015	Wissenschaftlicher Mitarbeiter am Fachgebiet Vermessung und Geoinformatik der Tongji Universität, Shanghai, China
2015 – 2017	Doktorand und Wissenschaftliche Hilfskraft am Fachgebiet Photogrammetrie und Fernerkundung an der Technischen Universität München
2017 – 2018	Wissenschaftlicher Mitarbeiter am Fachgebiet Photogrammetrie und Fernerkundung der Technischen Universität München

Acknowledgment

First and foremost, I would like to convey my sincere thanks to Prof. Uwe Stilla, who supervise my doctoral work at the Technische Universität München, for providing me the opportunity to be a part of his research team. The optimal atmosphere and conditions for researching within his team contributed significantly to my successful completion of the doctoral study. As always, I feel that I was really lucking to choose such a stimulating and fascinating research environment, because, during the four years of my research, Prof. Uwe Stilla was always available for a discussion regarding both scientific and formal, administrative matters, for which I am very grateful. I want to thank him for many useful tips regarding the publishing process of scientific manuscripts, including responding to reviewer comments, properly structuring the paper, etc. In particular, I am really grateful that Prof. Uwe Stilla provided me the chance to get the lecturer position at TUM, for gaining teaching experience, which also encourages me to continue pursuing my academic career in the future.

Research in a great team is always beneficial, here, I would like to convey my deepest thanks and best wishes to my friends and colleagues from the Chair of Photogrammetry and Remote Sensing for great and fruitful collaboration. In particular, my special thanks go to Dr. Ludwig Hoegner, Dr. Przemyslaw Polewski, and Dr. Sebastian Tuttas for their support during my researches. Also, I would like to express my gratitude to Prof. Wei Yao for many helpful discussions and directions, without his help and support, I cannot finish my work within the planned time interval of four years. All this contributed to an unforgettable four years. Last, but not least, I would like to thank my wife Ms. Chenguang Li and my parents for always believing in me and for their continual support.