Urban Modeling from Point Clouds

Dipl.-Ing. William Nguatem
Institute for Applied Computer Science
Visual Computing

2020

Urban Modeling from Point Clouds

William Nguatem

Vollständiger Abdruck der von der Fakultät für Informatik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktor-Ingenieur (Dr.-Ing.)

genehmigten Dissertation.

1. Gutachter: Univ.-Prof. Dr.-Ing. habil. Helmut Mayer

2. Gutachter: apl. Prof. Dr.-Ing. Norbert Haala

Die Dissertation wurde am 29.10.2020 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Informatik am 13.02.2021 angenommen. Die mündliche Prüfung fand am 29.03.2021 statt.

Acknowledgements

My research and education would not have been possible without the help and support of advisors, colleagues, friends, and family.

I am extremely grateful for the advice and support of Professor Helmut Mayer, who has guided me on both academic and personal decisions and whose influence can be seen throughout this thesis. Thank you Helmut for your devotion. Your enthusiasm kept me going in the lows and your insight guided me to exciting intellectual journeys. I will never forget your exceptional generosity with your time and energy. I had the privilege to have Professor Norbert Haala as second referee of this thesis. Thank you Professor Norbert Haala for your support.

I already miss very much the visual computing research group (led by Professor Helmut Mayer) and the AGIS research group (led by professor Wolfgang Reinhard), where I was surrounded by exceptional individuals, with big brains and big hearts. I have the fortune to have interacted with Professor Wolfgang Reinhard, Hai Huang, Mario Michelini, Winhard Tampubolon, Kujtim Rahmani, Imran Hossain, Admire Kandawasvika, Gisela Pietzner, Lukas Roth, Matthias Schmitz, Wolfgang Brandenburger, Martin Drauschke, Andreas Kuhn and Patrick Reidelstürz. These interactions were not restricted to the lab environment.

Finally, this thesis is dedicated to the memory of my late mum and dad, and to my entire family and friends, for the boundless love, encouragement and belief in me, and for the constant reminders of the truly important things in life. I especially want to thank my sister Pamela Tazi for her endless support and my late friend and in-law Kurt. To my kids Alemji and Awungafac, who will be happy to learn that I managed to finish the writing of this thesis and now have more time for our weekly baking experiments (without any recipes), which almost always end in a disaster. To my wife Emma; I can't be happier for having her on this journey and on the road ahead.

Kurzfassung

Diese Arbeit stellt eine vollautomatische Pipeline zur Modellierung von Gebäuden in einer urbanen Umgebung aus 3D-Punktwolken vor. Im Gegensatz zu bestehenden Ansätzen ist die Pipeline robust, skalierbar und bietet eine vollständigere Beschreibung, indem keine Manhattan-Welt-Annahme gemacht wird und sowohl Gebäude (mit Polyedern) als auch der nicht ebene Boden modelliert werden, wobei die natürliche inhärente Glattheit des Bodens erhalten wird. Sie beruht auf Wahrscheinlichkeitstheorie und skaliert auch für Datensätze mit Hunderten von Millionen verrauschten 3D-Punkten abgeleitet aus Bildern oder LiDAR-Scans. Die Pipeline basiert auf einer genauen, für große Flächen geeigneten Segmentierungsmethode, die eine Kombination aus RANSAC (RANdom SAmple Consensus) und nichtparametrischem Bayesschem Clustering nutzt, einem grundlegenden Satz von Regeln, die eine semantische Interpretation der Szene ermöglichen, der Detektion von Priors für die geometrische Form von Gebäuden, Polygon-Sweeping und Muster- sowie Oberflächenanpassung zur Modellierung sowohl von natürlichen (Boden) als auch künstlichen Oberflächen (Gebäuden). Die Pipeline ist nicht nur robust gegen starkes Rauschen, effizient und skalierbar, sondern arbeitet auch vollautomatisch und es müssen nur sehr wenige Parameter eingestellt werden, was eine einfache Nutzung ermöglicht. Die Wirksamkeit der Pipeline wird anhand einer Vielzahl von sowohl aus der Forschungsgemeinschaft stammenden als auch selbst generierten Datensätzen aufgezeigt.

Abstract

We present a fully automatic pipeline for modeling buildings in an urban environment from 3D point clouds. In contrast to existing approaches, our pipeline is robust, scalable and provides a more complete description by not making a Manhattan-World assumption and modeling both buildings (with polyhedra) as well as the non-planar ground, while maintaining the natural inherent smoothness of the ground. It relies on probability theory and scales to datasets with hundreds of millions of noisy 3D points derived from images or LiDAR scans. The pipeline is based on an accurate large-scale segmentation method using a combination of RANSAC (RANdom SAmple Consensus) and nonparametric Bayesian clustering, a set of basic rules enabling semantic scene interpretation and detection of shape priors for buildings, polygon-sweeping and template as well as surface fitting for modeling both natural (the ground) and man-made surfaces (buildings). Besides being robust against substantial noise, efficient and scalable, our pipeline operates fully automatically and has very few parameters to tune, hence, enabling an easy usage. We demonstrate the effectiveness of our pipeline on a wide variety of datasets from both the research community as well as our acquisitions.

Contents

Chapter 1

Introduction

Although the real world is three-dimensional (3D), humans perceiving their environment in two-dimensional (2D) images are nevertheless able to semantically interpret it in 3D. This is largely enabled through retinal disparity (Faugeras 1993, Qian and Qian 1997). The reconstruction and generation of accurate 3D object models from visual perception is a topic of great interest across several disciplines (computer graphics, computer vision, remote sensing, photogrammetry, signal processing) and in multiple domains including the military as well the public and private sector. In computer graphics and computer vision for example, recent advances in 3D depth cameras such as Microsoft Kinect, Figure 1.1 (left), have created new opportunities in the way humans interact with a computer—3D machine perception (Fossati et al. 2012). These developments have paved the way for novel applications in the consumer game and entertainment industry, e.g., enabling interaction in "true" virtual worlds by using 3D displays or glasses such as Oculus Rift, Figure 1.1 (right). Furthermore, the advent of autonomous driving vehicles and the corresponding demand for high definition semantic maps such as Baidu Maps, HERE Maps, Google Maps and Microsoft Bing, demonstrate the current interest in, and tremendous potential of methods to generate 3D models of cities as well as urban scenes. Similarly, in the construction industry, architects as well as public institutions responsible for city planing and housing also see huge benefits arising from accurate 3D models of urban scenes in Building Information Modeling (BIM) systems. These systems contain building models at various Levels of Detail (LOD) and can also be of great help, e.g., for rapid intervention and evacuation by the military or for rescue purposes in emergency situations by the fire brigade (Wu et al. 2015).

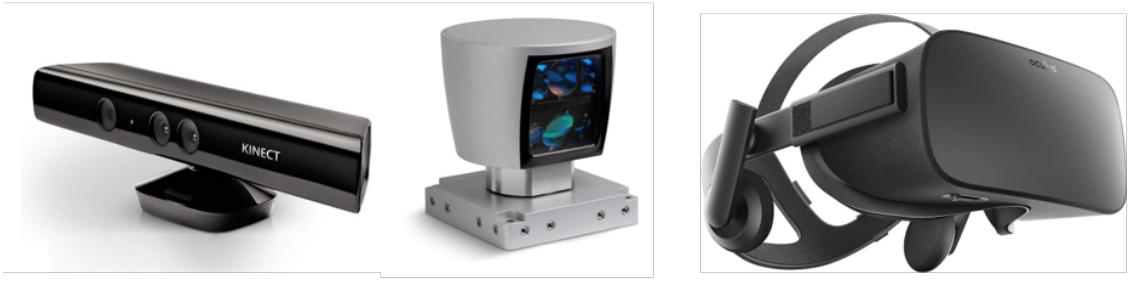

Figure 1.1: Example 3D Sensors (left and center) and 3D display (right)

Nevertheless, the creation of compelling 3D content as a prerequisite for BIM systems,

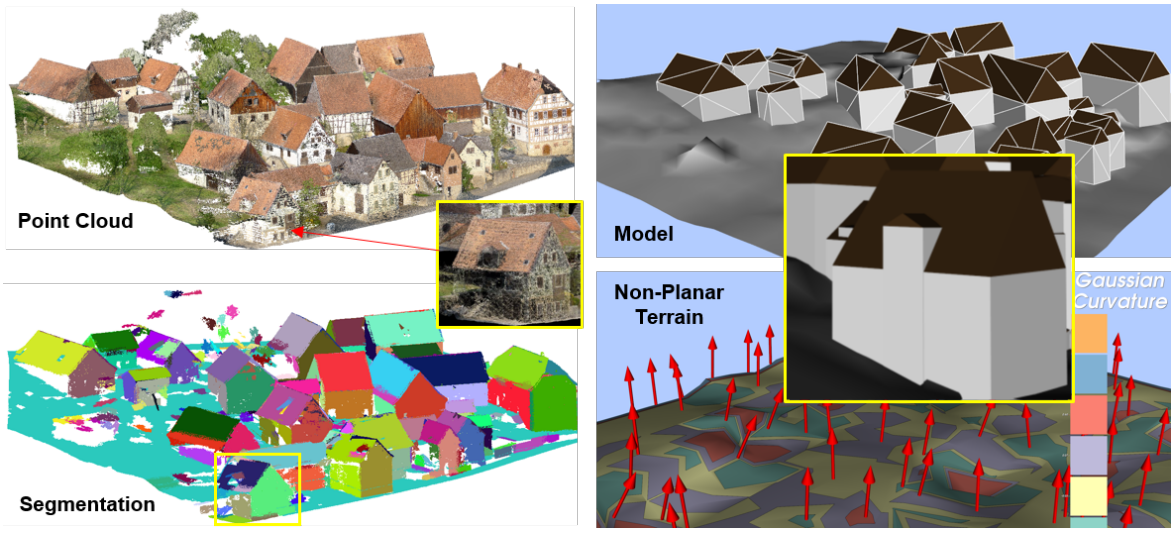

Figure 1.2: The proposed pipeline has converted 35 million raw 3D points (top left) of a residential scene containing an irregular arrangement of buildings on a non-planar terrain (bottom right depicts curvature profile) through consistently segmented patches (bottom left: random colors are assigned to the segmented patches) into a watertight semantic polygonal model (right).

immersive games, virtual worlds and movies is a notoriously difficult task. It has traditionally been solved by 3D content creation artists, often requiring hundreds of hours of skilled manual labor. Furthermore, the problem is exacerbated in models of buildings and their immediate surroundings since architectural scenes often exhibit fine detail at multiple scales. Approaches which automate this task of "hand crafting" 3D content would be highly beneficial for all anticipated applications.

1.1 Problem Statement

This thesis addresses the problem of creating accurate 3D models of urban scenes from point clouds. The major computational tasks that are investigated are shown in Figures 1.2 and 1.3. In both cases, the input data consist of 3D point clouds of an urban scene containing buildings as viewed from outside. The point clouds are derived by either Light Detection and Ranging scanners (LiDAR) or from images by Structure from Motion combined with Multiview Stereo (SfM/MVS). The input is a set of points with 3D coordinates which are estimates from the laser beams or pulses (for the LiDAR), or of the triangulation of intersecting rays (for SfM/MVS) for object surfaces in the urban scene. In most cases, the points are close to the real surfaces. However, they only give approximations of the surfaces due to the complexity of the real world as well as challenges during data acquisition which we present in the following.

The first computational task can be formulated as follows using the example presented

in Figure 1.2: Given a 3D point cloud $\mathcal{D}$ (top left), estimate an accurate semantic 3D surface model $\mathcal{S}$ (top right), that represents major geometric information of the objects in the scene with associated semantics such as:

1. The scene depicts 20 buildings.
2. All 20 buildings have gable roofs.
3. All buildings reside on non-planar terrain, in Figure 1.2 (bottom right) represented by the direction of the surface normals by red arrows as well as estimates of the surface curvature value coded in six distinct colors.

In the context of 3D city models, this form of 3D representation for architectural scenes is known as Level of Detail 2 (LOD 2) (Kolbe and Bacharach 2006). The second computational task can be seen in Figure 1.3. Given a 3D point cloud and a corresponding LOD 2 shape model $\mathcal{S}$, deduce a set of shape coefficients $\mathcal{T}$ that captures all the individual geometric entities and the associated semantic. Therefore, this thesis aims at producing surface models $\mathcal{M}$ consisting of the shape model and the corresponding coefficients $\mathcal{T}$, given an input point cloud $\mathcal{D}$. The output is, thus given by

$$\mathcal{M}=\mathcal{S}\cup\mathcal{T} \tag{1.1}$$

These two tasks pose serious challenges for state-of-the-art algorithms for 3D modeling from point clouds. They are due to the (physical) uniqueness of urban scenes during data acquisition and hence the characteristics of the resulting 3D point clouds but also because of the difficulties in developing novel algorithms that capture these peculiarities. In particular, the process of automatically creating scalable and compelling 3D models of urban scenes from point clouds needs to deal with the following major sources of problems:

1. **Noise and reconstruction artifacts in the input data**: While a Gaussian noise model can basically be assumed for 3D point clouds derived from LiDAR scans, this is not the case for point clouds acquired from a SfM/MVS pipeline. Here, the level of noise is substantial and does not conform with any known single probability distribution particularly in combination with gross errors such as reconstruction artifacts. Therefore, the noise and artifacts can be seen as a complex stochastic process.
2. **Incomplete data**: Missing data occur due to occlusions, objects whose surfaces do not return measurements from LiDAR scanning, or are textureless when using SfM/MVS leading to incomplete data.
3. **Clutter**: Scene clutter and variations in building placement, shape and scales as well as differences from one building to another (object diversity) makes the algorithmic design extremely difficult.

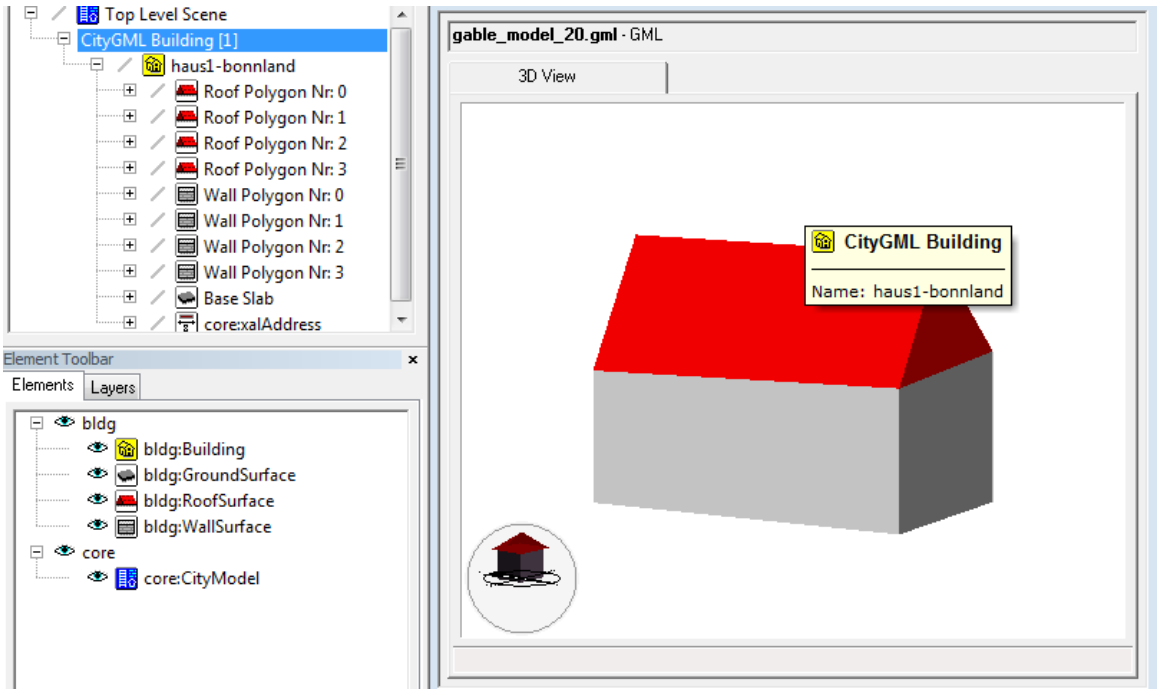

Figure 1.3: Geography Markup Language (GML) viewers enable the visual assessment of both geometric and semantic attributes of our results in LOD 2. This example illustrates a GML representation for a simple gable roof building model derived from our pipeline.

4. **Point Density Variation**: Surfaces in urban scenes exhibit a high diversity, hence a high point density variation. This is particularly severe for point clouds derived from SfM/MVS pipelines.

5. **High Runtime**: Processing capabilities are finite due to hardware limitations. Algorithmic design has to take this into consideration to avoid unacceptably high runtimes.

1.2 System Architecture

The proposed system architecture for modeling buildings from 3D point clouds is depicted in Figure 1.4. It shows the flow of the input data (unorganized 3D point cloud $\mathcal{D}$) through the intermediate stages of the system to the output surface models ($\mathcal{M}$) made up of shape models $\mathcal{S}$ and associated coefficients $\mathcal{T}$. The system is divided into three major stages:

1. Segmentation extracts patches,

2. Semantic scene interpretation assigns labels to patches and detects unique geometric shape primitives for buildings, and finally, the

3. Surface modeler creates geometric shapes and surface models from segmented and labeled objects in the scene.

The three major goals which lead to this architectural sub-division are:

1. Robustness,
2. Modularity, and
3. Efficiency and scalability.

To further increase the performance of the proposed pipeline, we transform the input data $\mathcal{D}$ to metric scale and a world coordinate frame with the $z-$axis pointing towards the vertical (up) direction.

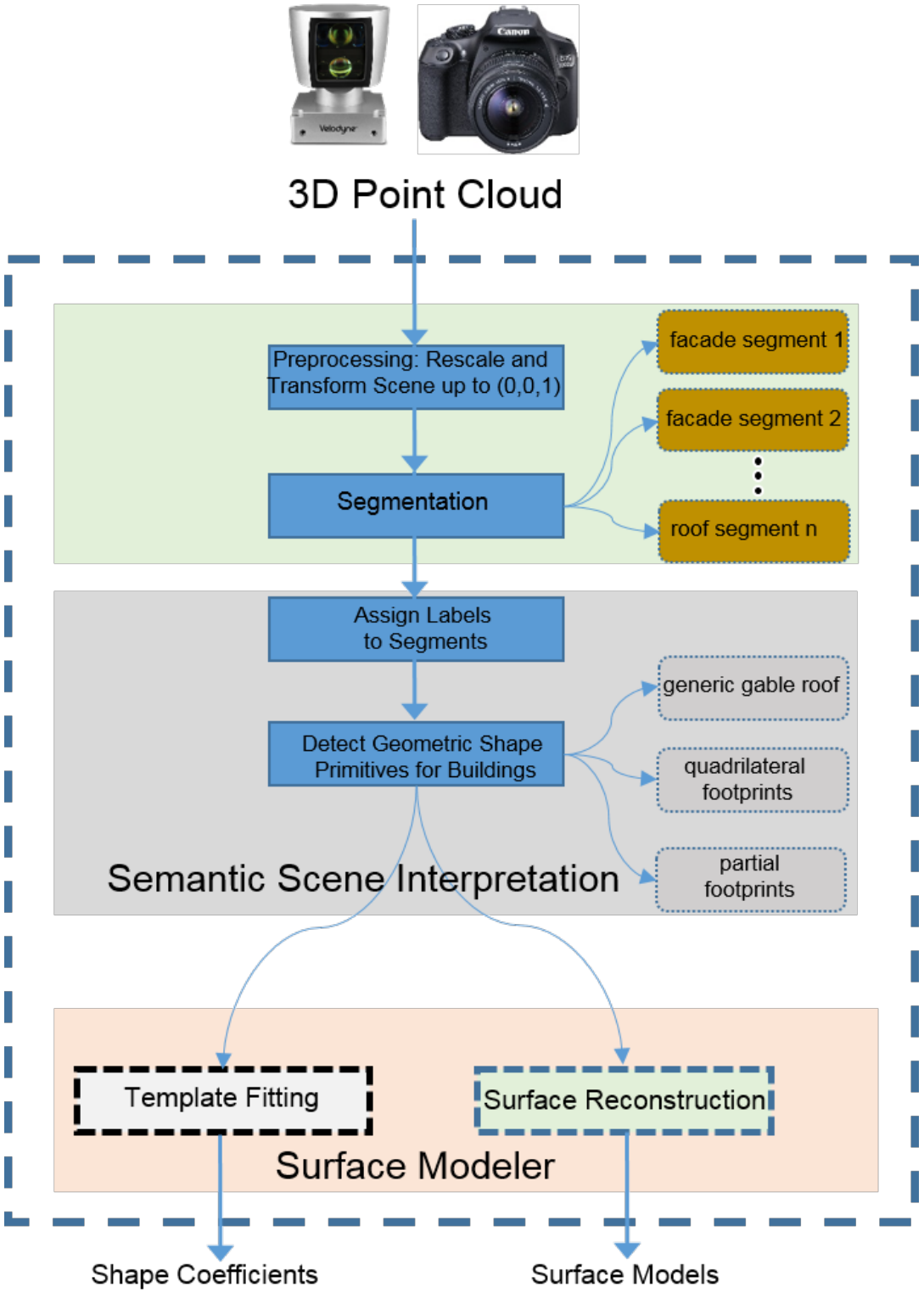

Figure 1.4: The architecture of the proposed pipeline for modeling from point clouds as presented in this thesis. The input data consist of unorganized 3D point cloud $\mathcal{D}$, and the output is a set of surface models $\mathcal{M}$, made up of shape models $\mathcal{S}$ and an associated set of shape coefficients $\mathcal{T}$.

1.3 What This Thesis Is and Isn't

This thesis describes a scalable system for modeling urban scenes from 3D point clouds and provides algorithmic solutions to the above problems with a focus on scalability, applicability to point clouds from both SfM/MVS and LiDAR as well as robustness against substantial noise and clutter. It works with 3D point clouds representing a single building, tens of buildings, or even a complete urban residential area and their immediate surroundings. It is designed for modeling urban scenes and the output consists of major architectural features such as the geometry of roofs, walls, and also the surrounding ground surface. Conversely, the proposed algorithms do not recover all the details of an architectural scene and its immediate surroundings. Vegetation is not modeled, as well as parked vehicles, pedestrians, just to name a few. Also, we have not tested the proposed algorithms on data sets of entire cities with hundreds of buildings including tall buildings. The reasons for this are twofold: Point cloud data acquisition at such a huge scale is time consuming and there are limitations of current algorithms for image as well as LiDAR scan registration. Additionally, the average desktop workstation is limited in memory size – in our case 64 GB – which is suitable only for moderately sized data sets.

1.4 Thesis Outline

This thesis focuses on the modeling of urban scenes from 3D point clouds. Major characteristics of the proposed solution are scalability, robustness and use of probability theory. The latter has been the natural framework underpinning the impressive success achieved while solving many difficult real-world problems in categorization, learning etc. (Hastie et al. 2009), which substantiates our choice. Our proposed solution is based on stochastic processes and combines existing estimation algorithms in a unique, yet unifying way that ensures robustness against noise and overall scalability. The thesis consists of the following chapters:

Chapter 2: Prerequisites

We begin with a description of the input data, the point cloud, as a 3D signal representation and briefly explain how it can be directly acquired using active techniques (e.g., a LiDAR scanner) or indirectly inferred from 2D imagery via a passive technique called Structure from Motion in combination with Multiview Stereo (SfM/MVS). We compare these two approaches for 3D point cloud data acquisition and present their pros and cons for our use case. In the second part of this chapter, we review robust parametric model fitting based on the random sampling consensus paradigm (RANSAC) and discuss the Gibbs sampling algorithm. These two algorithms (RANSAC and statistical sampling) have become the basis of a plethora of successful applications in photogrammetry, remote sensing and computer vision and our subsequent chapters are based on them.

Chapter 3: Related Work

We first review the broad range of contributions towards creating object models from 3D point clouds. They have many aspects in common, hence are difficult to divide into

non-overlapping groups. We focus our review on the core theme of each contribution and also associate it to one or more of the following categories: Semi-Automatic Modeling, Point Cloud Classification, Procedural Modeling, Fitting Geometric Primitives, Modeling Beyond a Manhattan-World and Shape Detection in Point clouds. We conclude the chapter by presenting the strengths and weaknesses of these algorithms, hence, substantiating the need for our proposed pipeline.

Chapter 4: Clustering as a Mixture of Gaussians
This chapter first describes finite mixture of Gaussians model-based clustering. Next, we introduce the Dirichlet process mixture of Gaussians (DPM), which is widely used in non-parametric Bayesian statistics, review the statistical assumptions underlying this representation and provide an efficient inference method based on Gibbs sampling. Our presentation focuses on the application of DPM as a tool for data clustering within the challenging scenario when "k", the number of clusters present in the data is unknown and has to be jointly inferred alongside the parameters of the clusters. This is often referred to as infinite mixture modeling.

Chapter 5: Large-Scale Patch Segmentation in Point Clouds
The fifth chapter applies a divide and conquer approach for scene segmentation. It is based on a representation for surface normals that does not "overfit" for noisy data and is robust against point density variation, the "Clip Plane Normal" (CPN). We then employ DPM for clustering CPN on the unit sphere, followed by connected component analysis and smoothing. The result is a consistent scene segmentation algorithm which automatically infers an appropriate number of patches and estimates corresponding object boundaries.

Chapter 6: Semantic Scene Interpretation
This chapter describes a set of basic architectural rules which are used in combination with spatial relations for point cloud labeling. We employ four basic class labels and associate the results of segmentation to them, leading to a consistent categorization of scene structures. Furthermore, we introduce the concept of a shape primitive for buildings – combinations of labeled segments which signify the presence of a particular building type. These primitives are used as strong priors for building model fitting in the next chapter.

Chapter 7: From Geometric Shape Primitives to Building Models
In this chapter, we generalize the problem of parametric model fitting to the complex shapes of urban scenes using templates of shapes for buildings. The detected shape primitives for buildings from the previous chapter are used as a strong prior for data fitting. To account for uncertainty and the variability of the shapes, we employ model selection.

Chapter 8: Experiments
Here we describe our experiments beginning with a thorough description of the input data and the parameter set. We discuss the choice of default values, the implementation and operating environments and finally conclude by evaluating the algorithms where possible.

Chapter 9: Strengths and Weaknesses
In this chapter, we present the strengths and weaknesses of our modeling pipeline.
Chapter 10: Conclusion
Finally, we conclude by summarizing the contributions of this thesis and outline directions for future research.

1.5 Publications

Most of the material of this thesis has been presented at major conferences and workshops in the fields of computer vision, photogrammetry and remote sensing after successfully passing a double-blind peer-review process. An excerpt from this list is given below beginning with the most relevant publications:

- William Nguatem, Helmut Mayer. *Modeling Urban Scenes from Point clouds.* ICCV 2017, Venice, Italy.
- William Nguatem, Helmut Mayer. *Contiguous Patch Segmentation in Point clouds.* GCPR 2016, Hannover, Germany.
- William Nguatem, Martin Drauschke, Helmut Mayer. *Automatic Generation of Building Models with Levels of Detail 1-3.* Proceedings of the XXIII ISPRS Congress, Technical Commission III 2016, Prague, Czech Republic. **Best Poster Paper Award.**
- William Nguatem, Martin Drauschke, Helmut Mayer. *Roof Reconstruction from Point Clouds Using Importance Sampling.* City Models, Roads and Traffic (CMRT) 2013, Antalya, Turkey. ISSN 2194-9042.
- William Nguatem, Martin Drauschke, Helmut Mayer. *Finding Cuboid-based Building Models in Point Clouds.* Proceedings of the XXII ISPRS Congress, Technical Commission III 2014, Melbourne, Australia.

Chapter 2

Prerequisites

In this chapter, we present the input data, discuss mathematical and statistical tools and define the notation used in the subsequent chapters. First, we describe the general structure of the input three dimensional (3D) data, the 3D point cloud, with major attention on geometric representations (Section 2.1) and different acquisition modalities (Section 2.2). Then we review the concept of an inlier to a parametric model and introduce the statistical tools underpinning the algorithms developed during this thesis (Section 2.4). Finally, we provide concluding remarks.

2.1 The Point Cloud – A Three-Dimensional Data Representation

There exist many different representations for 3D multi-modal data: Surface meshes, Digital Elevation Models (DEM), Digital Terrain Model (DTM), 3D points in cartesian coordinates (x, y, z), just to name a few. These different representations are a result of convenience and suitability for 3D multi-modal data in various disciplines ranging from artificial intelligence, robotics (Rusu and Cousins 2011), machine perception (Faugeras 1993) to geometrical modeling (Ericson 2004). In this thesis, we focus on just one form of 3D multi-modal data representation, the "3D point cloud". Point clouds can be obtained from LiDAR as well as from images by a computer vision processing technique known as Structure from Motion in combination with Multiview Stereo (SfM/MVS) which we describe in the sequel. Example 3D point cloud data produced from images by SfM/MVS are shown in Figure 2.1 (left).

Figure 2.1: 3D point clouds produced from images by SfM/MVS are becoming very popular. These examples show two point clouds derived from a SfM/MVS pipeline for both an outdoor (top left), and an indoor (bottom left) scene. In both cases, each 3D point is associated with a corresponding color value r, g, b. This allows for a visual comparison with a 2D image (right) from the image sequence.

2.1.1 3D Points and Point Clouds

A 3D point $\mathbf{P}$, as described by 3D range sensing systems (Siciliano and Khatib 2007) and in geometry (Faugeras 1993, Gross and Pfister 2007) is represented in its simplest form using the cartesian coordinates x, y, z. Usually, $\mathbf{P}$ is defined with respect to a given origin, direction of the coordinate axes and scale. Two choices of origins are widely used: the origin of the world coordinate system or a local origin defined by the data centroid, $\hat{\mathbf{P}}$. A major advantage of the local origin is its invariance to translation, hence, is often chosen. Point cloud data, $\mathcal{D}$, in its basic form, is an ensemble of 3D points of the same type, i.e., sharing similar attributes. These could be normals, color, or intensity values. In general, the presence of point attributes extends the definition of a 3D point towards more generic n-dimensional data. Example representations with associated attributes are:

$$\mathbf{P}=(x,y,z,i) \rightarrow with\ intensity\ (i)$$
$$\mathbf{P}=(x,y,z,r,g,b) \rightarrow with\ color\ (red, green, blue)$$
$$\mathbf{P}=(x,y,z,n_x,n_y,n_z) \rightarrow with\ normals$$
$$\mathbf{P}=(x,y,z,c_x,c_y,c_z) \rightarrow with\ curvature$$

Therefore, the generic 3D point is an $n-$dimensional tuple defined by

$$\mathbf{P}=(x,y,z,r,g,b,i,n_x,n_y,n_z,\cdots), \tag{2.1}$$

where x, y, z represents the cartesian coordinates r, g, b the associated color, i the intensity and n_x, n_y, n_z the normals. Throughout this thesis, unless otherwise stated, we assume that the input data is represented as a 3D point cloud translated so that the origin corresponds to the data centroid, $\hat{\mathbf{P}}$. Furthermore, we assume that the input data is transformed such that the vertical (up) direction of the scene is aligned with the $z-$axis.

2.1.2 Basic Features of 3D Point Clouds and the Concept of Neighborhood

In their simplest form, 3D point clouds of objects are represented by cartesian coordinates x, y, z with respect to an origin. Salient features are those describing the geometry of the objects in the scene. The most prominent is the normal of individual 3D points which indirectly quantifies the surface geometry of objects in the scene. Traditionally, normals are estimated using eigenvectors of covariance matrices (Rusu and Cousins 2011). Ideally, these are good approximations of the underlying object's surface and are sometimes referred to as object's surface normals (Ericson 2004, Siciliano and Khatib 2007).

Unfortunately, the accuracy and precision of the estimated normals in comparison to the true normals of object's surface strongly depends on the size of the immediate surrounding considered for a specific point. Therefore, the choice of this size is crucial for accurate normal and surface curvature estimation. In this respect, the notion of a local neighborhood – a kernel describing the immediate surrounding of a point – becomes important. Usually, the kernel is defined by a local radius r whose value is manually set or inferred. All points within r have equal probability, hence are responsible for the local geometric characteristics of a point.

Unfortunately, this definition of normals can lead to a strong bias in the case of sudden discontinuities often in the proximity of sharp edges. Furthermore, it is only correctly applicable to surfaces with a homogeneous point density. In point clouds derived from urban scenes, this is hardly the case due to differing acquisition distances and surface properties of the objects, leading to object surface diversity. This problem is exacerbated for point clouds of urban scenes derived from images by SfM/MVS. Choosing a small value for the kernel size, results in normals of individual 3D points capturing many tiny little surface undulations which aren't representative of the underlying true surface geometry. This problem can be remedied by post processing via normal smoothing or by

increasing the kernel size. In both cases, the additional computational cost can becomes bottleneck. Figure 2.2 shows a point cloud of a building alongside the surface normals estimated using eigenvector analysis with two different values for r. The effects of an inappropriate choice of r can be noticed in the red rectangle. Usually, the orientation of the normals is ambiguous when estimated by eigenvector analysis of the covariance matrices. However, with additional information, e.g., about the sensor locations and viewing directions, this can be readily resolved.

In contrast to this form of surface geometry representation via normals of individual 3D points, we present in Section 5 an alternative and robust approach that captures the scene geometry for 3D point clouds of urban scenes in a better way.

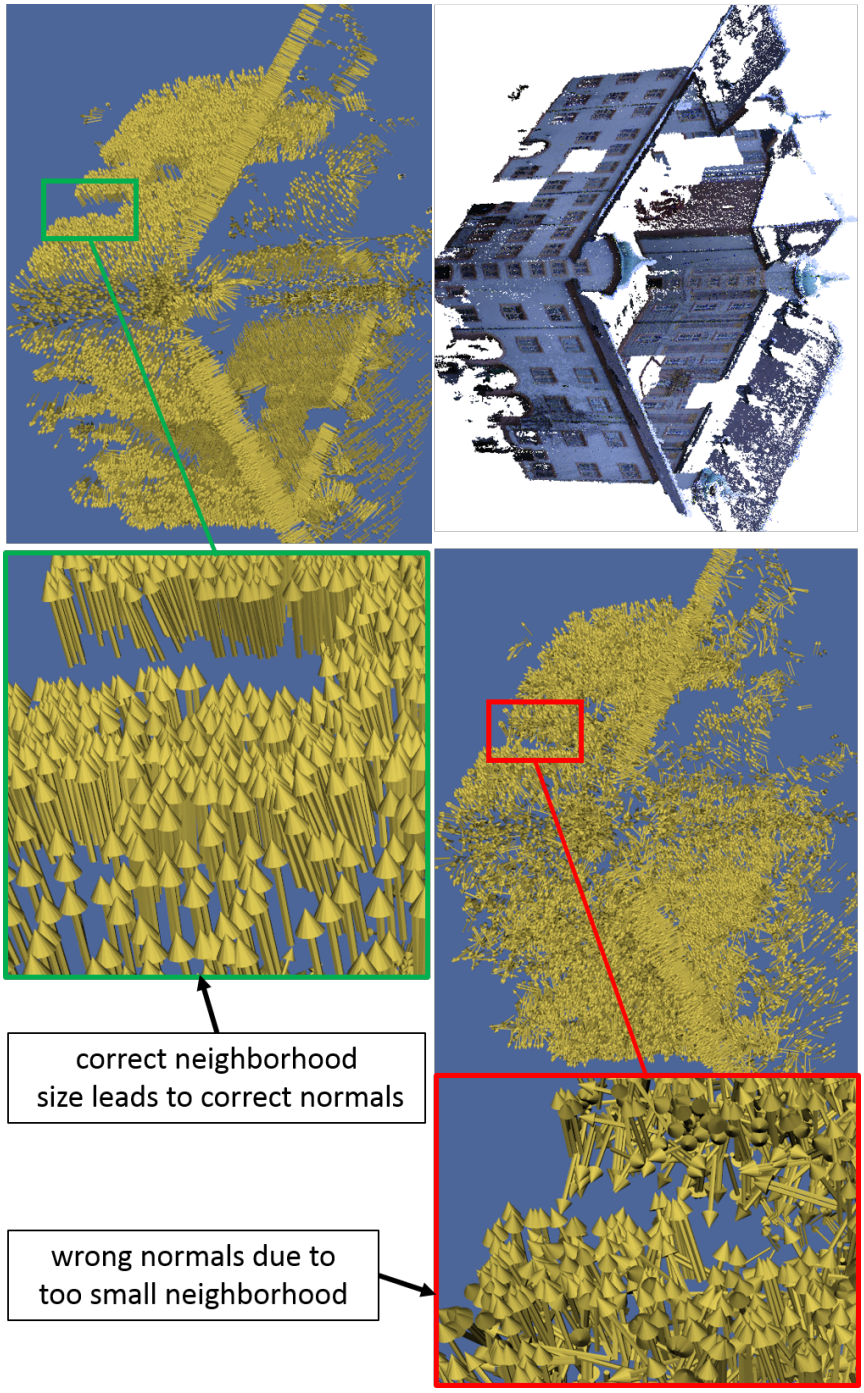

Figure 2.2: Surface normal estimation can be very sensitive to the size of the neighborhood when using traditional approaches. Choosing the right value leads to well estimated normals (arrows in green rectangle) as opposed to a wrong value resulting in incorrect normals (arrows in red rectangle).

2.1.3 Data Structures for Processing 3D Point Clouds

In the previous section, generic 3D points constituting point clouds have been defined as $n-$dimensional tuples. Processing such a high dimensional dataset naively can be computationally very expensive. However, there exist many data structures suitable to speed up computations with high dimensional data types. Trees and graph-based data structures are the most widely employed. We use *kd*-trees, octrees as well as adjacency graphs to speed up the search for neighbors of points without re-computing distances between points each time. These data structures employ different spatial decomposition techniques dividing $\mathcal{D}$ into chunks with well defined inter-chunk relationships. For queries within chunks, an octree is favored as compared to a *kd*-tree since the layout of the octree is volumetric enclosing all its points in boxes called "voxel". An example of an octree decomposition of a scene is given in Figure 2.3. Yet, for more generic queries with respect to distance, a *kd*-tree (Muja and Lowe 2014) is employed.

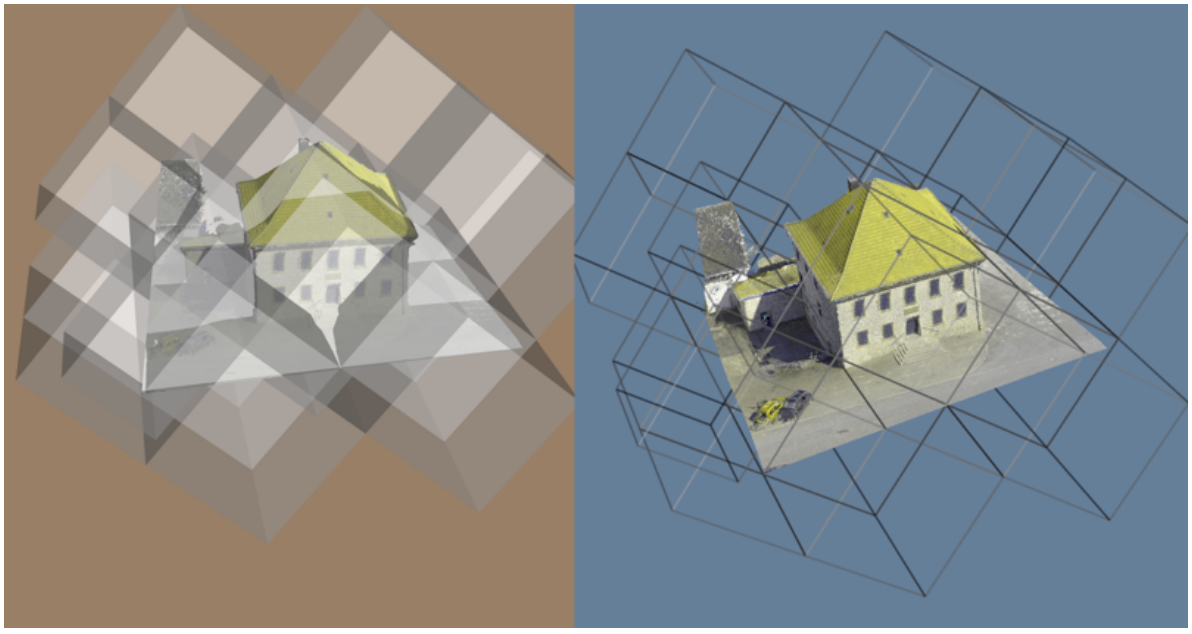

Figure 2.3: A basic data structure for 3D point cloud processing is the octree. In this example, we use a voxel of size 2m for illustration and render the same scene twice: with solid surfaces (left) and wire-frames (right). The octree decomposes the scene into boxes and speeds up queries within the proximity of a voxel since voxels further away will not be considered.

2.2 Point Cloud Data Acquisition Modalities

Point cloud data $\mathcal{D}$ is the input of the pipeline developed in this thesis. Many approaches exist to acquire accurate 3D point clouds of objects such as structured light scanning,

shape from shading, volumetric scans, LiDAR and SfM/MVS. While structured light is very popular for small objects and the key technology for point cloud acquisition in close-range indoor scenes, it is less suitable for scanning objects tens to hundreds of meters away. This is the domain of LiDAR and SfM/MVS which we have employed and for which we present more detail in the following.

2.2.1 Point Cloud from LiDAR

The basic principle behind 3D point cloud acquisition based on LiDAR is illustrated in Figure 2.4. Laser beams or pulses are emitted from a device. These are reflected by points on the surface of objects in the scene, and return to the device. The phase modulation of the beams or the time-of-flight of the pulses is combined with the speed of light to compute the distance to the points in the scene. This results in accurate measurements of objects. To capture a complete surface, the scanning device usually contains a fast spinning mirror and a rotating motor that jointly operates to sweep the laser spot across the scene in a defined grid pattern. As opposed to structured light, time-of-flight based LiDAR devices can measure distances of hundreds of meters (see Figure 2.4), while phase-based systems have a maximum range of about eighty meters (Radke 2012). Despite the long distances as compared to structured light, data acquired via LiDAR scanning devices can be very accurate, often within a few millimeters. Furthermore, the distance to points on objects on the scene is measured directly, as opposed to being estimated indirectly, e.g., from images by SfM/MVS. For these reasons, laser scanning is considered the de facto standard for 3D data acquisition for urban scenes.

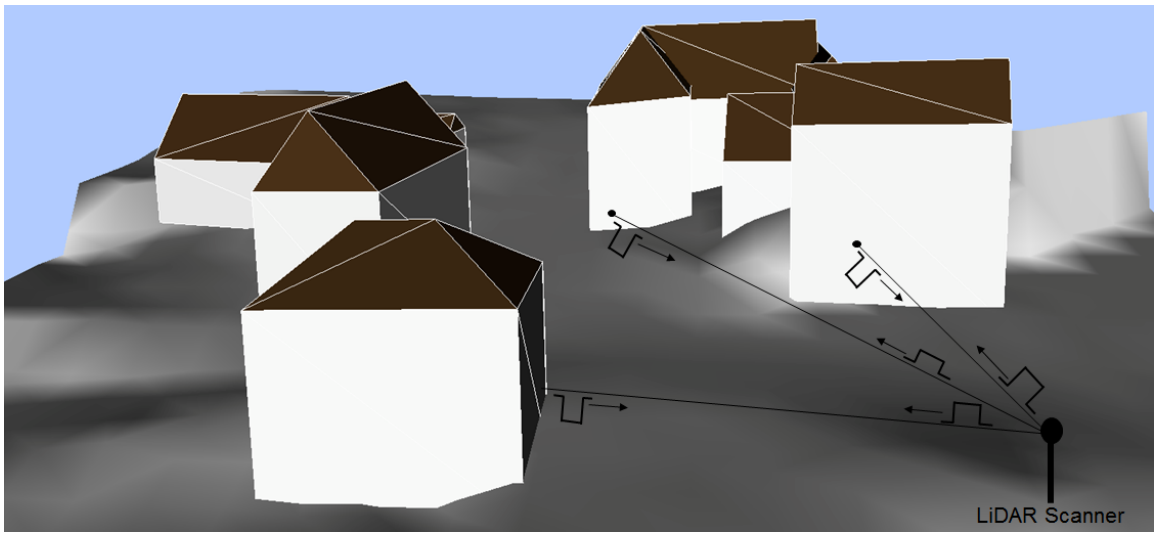

Figure 2.4: Distances from the LiDAR scanner to object surfaces in the scene can be measured using the time of flight of laser pulses.

The 3D point clouds acquired by LiDAR scanning are generally of very high quality.

Yet, it is important to note that some materials are inherently problematic for this technology. Highly reflective surfaces such as glass often result in missing distance measurements. Also, dark or highly absorptive surfaces are difficult to scan since an insufficient amount of light is reflected back to the scanning device. Therefore, a scene containing bright, matte surfaces that are all roughly at right angles to the laser beam works best for LiDAR scanning. Man-made surfaces, e.g., buildings usually lead to good results (except for the glass windows or doors). Building roofs are difficult to scan from the ground due to the small viewing angle and because LiDAR scanners are in general not portable, i.e., cannot easily be mounted on an unmanned aerial vehicle (UAV) as opposed to a camera. Furthermore, LiDAR scans often suffer from characteristic "shadows" of missing data produced by occluding foreground objects. For large scenes, e.g., buildings, shadowed regions can be filled with data from adjacent scans from different perspectives once they have been merged in a common coordinate system – a procedure called co-registration (Radke 2012). LiDAR scanning devices are often designed to return just distance measurements, hence the resulting point cloud data consist only of the three coordinates x, y, z. However, for an appealing rendering, they often include an integrated color camera whose pixels are precisely aligned to the scans, enabling the raw 3D points to be texture-mapped with color.

2.2.2 Point Clouds Derived from Images

Range sensing using LiDAR is a mature technology. Yet, the development of new algorithms driven by many novel applications in photogrammetry, remote sensing, computer vision, and robotics (Goesele et al. 2006, Strecha et al. 2008, Furukawa and Ponce 2010, Kuhn et al. 2017, Huang et al. 2019) combined with the availability of cheap consumer cameras and dedicated computing hardware – Graphics Processing Units (GPU) – makes point clouds generated from images an interesting alternative. The processing is often divided into two distinct stages: The first stage is called Structure-from-Motion (SfM). The ultimate goal is to estimate accurate camera locations and orientations from sets of images. Approximate internal calibration of the cameras is assumed or estimated either by matching salient features in a fully 3D natural scene or using a calibration board. Alternatively, EXIF information from images often provides a good basis to estimates an approximate internal calibration. Matching points or regions produces correspondences between pairs of images and later across sets of images. Since the cameras are internally calibrated, triangulating the correspondences leads to 3D measurements of scene points. The second stage is called Multi-View Stereo (MVS) reconstruction. Four major groups can be distinguished:

- Volumetric methods discretize the scene in 3D and, hence, represent the scene as a finely sampled set of colored voxels. A set of voxels is selected such that the shape and colors of the voxelized scene is consistent with all the images (Kolmogorov and Zabih 2001)

- Surface Deformation methods can be viewed as a generalization of the active con-

tours or snakes used for image segmentation (Kass et al. 1988), evolving the 3D surface (represented as a triangle mesh or a level-set) to tightly wrap the region containing an object (Esteban and Schmitt 2004)

- Patch Matching methods generate small 3D patches of geometric primitives (e.g., planes) in the scene by triangulating multi-image matches of salient features, and grow these patches in 3D to account for as much of the scene as possible (Furukawa and Ponce 2010)
- Depth Map Fusion-based approaches start from a disparity map (depth map). They try to incrementally fuse (hence the name) the latter into a unified set of 3D points and hence surface (Goesele et al. 2006, Kuhn et al. 2017) (see Figure 2.5)

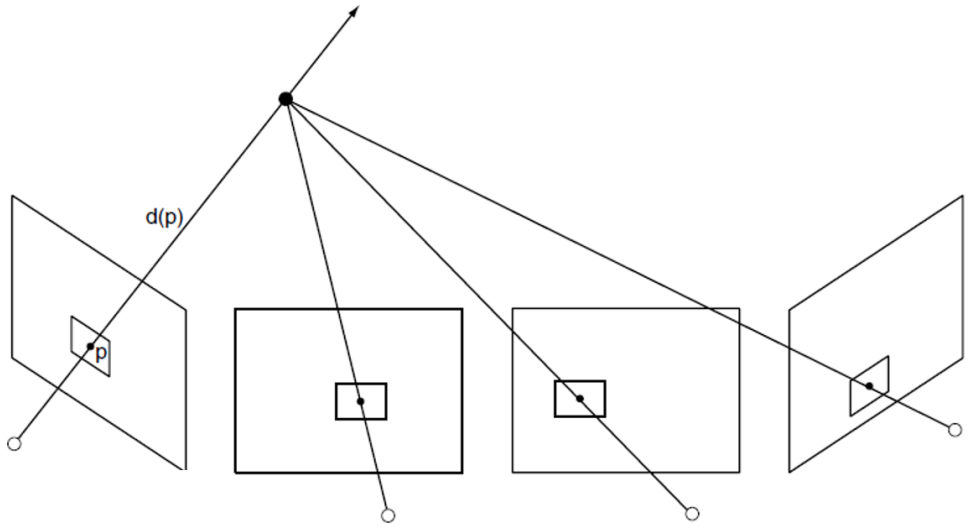

Figure 2.5: Depth $d(p)$ is estimated for a pixel p in the reference image from the image set employing, e.g., normalized cross-correlation of windows around the projected image locations.

It is important to note that while modern MVS results are qualitatively quite impressive, and quantitatively (i.e., millimeter or even sub-millimeter) accurate for small objects, 3D point clouds from purely image-based techniques suffer from a number of deficits: High point density variations resulting from differences in surface texture, from different distances to the cameras as well as variations in viewing angles. Illumination conditions during image set acquisition can also affect the quality of the point clouds. SfM/MVS algorithms can also be quite computationally expensive. All this notwithstanding, a major advantage of this form of data acquisition is the inherent flexibility. For example, a consumer camera can easily be mounted on a UAV as opposed to a LiDAR scanning device and it is much easier to take images from different perspectives (if necessary due to the 3D structure of the scene) than to move a LiDAR scanning device.

2.3 Robust Model Fitting

Point cloud data can be approximated by replacing the 3D points with parametric models. Ideally, databases of models (e.g., from CAD tools) for all objects present in the real world, e.g., buildings, chairs, cars, would exist. In this case, the creation of accurate models from point clouds could be reduced to replacing the points in $\mathcal{D}$ sampled from the object's surfaces with their corresponding models from the database. The problem remains to search and fit appropriate models. Supposing these algorithms would work very well, the task of modeling from point clouds data would be solved. Unfortunately, this is rarely the case because:

- Real world structures are often complex and difficult represent by a database of parametric models. This results in unexpected outcomes for the search of certain structures in point cloud data.
- Noise in the input point cloud data makes search and fitting difficult.
- Holes and clutter lead to incomplete as well as additional "ghost" models.

Therefore, instead of fitting complex models of objects, one of the most popular approaches for data simplification is to search and replace points with basic geometric primitives. Examples include planes, cylinders, spheres, higher order bivariate polynomials, or Non-Uniform Rational B-Splines (NURBS). The ultimate goal is to approximate the sensed data from the real world represented as point clouds within given error bounds with these parametric models.

The major reasons for the choice of this form of data simplification can be better understood in the context of integrating point clouds or parametric models within an application such as rendering in computer graphics or collision detection in mobile robotics. In the latter, a mobile robot moves around in the world and checks for collisions between its body parts and the surrounding. If the environment is represented just with 3D points, the robot would need to estimate a lot of distances from all its body parts to all 3D points in the world in order to compute whether a collision has occurred. Replacing the points belonging to flat surfaces with, e.g., large quads or bounding polygons can significantly reduce the number of collision checks, hence, reducing the computational load. Similarly, in a renderer, computing the z-buffer test for a large parametric model, e.g., a quad is computationally less expensive than for many hundreds of 3D points, even when using modern GPUs.

The RANdom SAmple Consensus paradigm (RANSAC) (Fischler and Bolles 1981) family of algorithms has been widely employed to speed up the robust search. It offers a generic method for robustly determining a consistent subset of points, the "concensus set", for a given model. The algorithm operates iteratively and is illustrated below using the case of the search for a plane:

1. Randomly select three non-collinear unique points p_i, p_j, p_k from the point cloud $\mathcal{D}$,

2. Compute the coefficients of a plane model from the three points: $\mathcal{P} = ax + by + cz + d$. $\mathcal{P}$ defines a hypothesis,

3. Iterate through all the points in $\mathcal{D}$ and compute the distances to the plane model $\mathcal{P}$,

4. Score $\mathcal{P}$ according to a cost function $\mathcal{C}_{\mathcal{P}}$.

$$\mathcal{H}\left(\boldsymbol{C}_P\right) = \exp\left(-\sum_j \rho\left(e_j\right)\right) \tag{2.2}$$

with

$$\rho\left(e_j\right) = \begin{cases} e_j & e_j < T \\ T & e_j \geq T \end{cases} \tag{2.3}$$

and e_j the Euclidean distance from point p_j in the point cloud to the surface $\mathcal{P}$. Alternatively, one could use the "vanilla" RANSAC cost function by counting the number of points p_j whose distance to the plane model falls within the user specified threshold. After the algorithm has terminated, the set with the smallest cost $\mathcal{H}\left(\boldsymbol{C}_P\right)$ or alternatively, the largest number of points (inliers) is selected as the support for the best planar model found. Often, the planar model coefficients are refined by a least-squares estimation using only the inliers. Figure 2.6 presents the set of inliers to a planar model and the resulting model fit for a facade using a point cloud derived by a SfM/MVS pipeline.

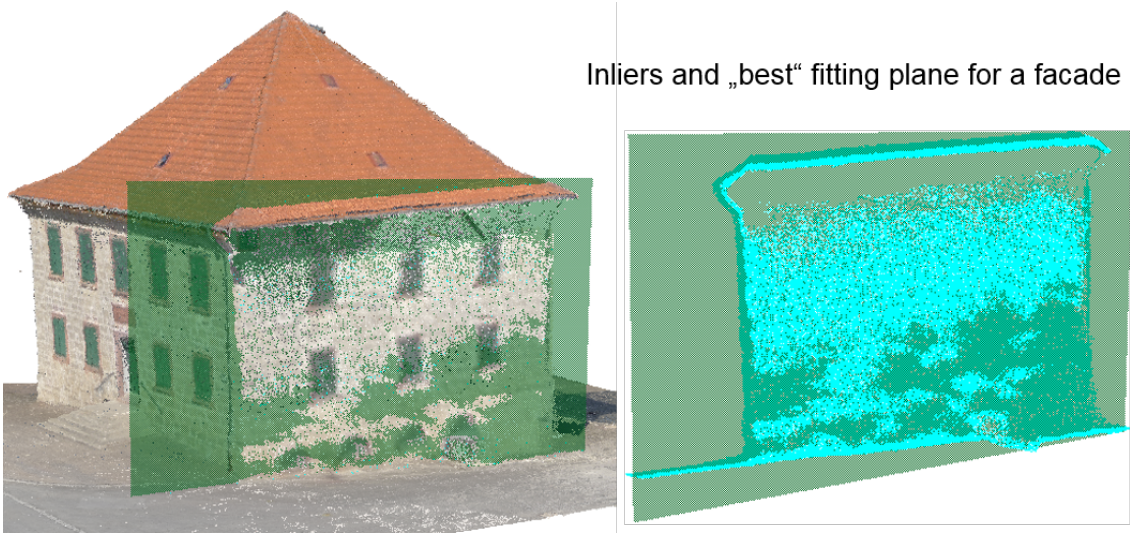

Figure 2.6: Left: point cloud and the resulting best fitting plane for a wall, right: the inliers of the wall denoting the supporting points of this hypothesis defined by the plane model.

2.4 Gibbs Sampler

Probability theory and statistics play a key role in the design and analysis of many highly successful high-dimensional machine perception and modeling systems. This is largely facilitated by the availability of high performance computers, enabling extensive statistical analysis to capture the underlying nature of complex real world phenomena. The major goal in these applications are amongst others, information extraction, inference and decision making. In this section, we present the Gibbs sampler and illustrate its use in Bayesian inference, considering the problem of parameter estimation. The primary objective is to choose parameters of a statistical model such that the data is "best" explained by a chosen instantiation of the model. In the context of this thesis, we consider the specific problem of sampling from the posterior distribution of the parameters in a mixture of Gaussian distributions, given a random sample from the mixture. Let the parameter vector be given by $\boldsymbol{\theta}$ and assume that $\boldsymbol{\theta}$ can be partitioned into B blocks, i.e. $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_B)$, such that the complete conditional posterior densities for each block

$$\begin{gathered} P\left(\theta_1 \left|\theta_2,\theta_3,\cdots,\theta_B,X_n\right.\right) \\ P\left(\theta_2 \left|\theta_1,\theta_3,\cdots,\theta_B,X_n\right.\right) \\ \vdots \\ P\left(\theta_B \left|\theta_1,\theta_2,\cdots,\theta_{B-1},X_n\right.\right) \end{gathered}$$

can be drawn, where X_n is the input data.

The Gibbs sampler directly draws samples iteratively from the conditional posterior distributions. To begin, we initialize the parameters to be estimated with the value $\boldsymbol{\theta}^{(0)} = \left\{\theta_1^{(0)},\theta_2^{(0)},\cdots,\theta_B^{(0)}\right\}$. The sampler then generates a sequence of random variables as follows

$$\begin{gathered} \textit{Step 1. Draw } \theta_1^{(1)} \sim P\left(\theta_1 \left|\theta_2^{(0)},\theta_3^{(0)},\cdots,\theta_B^{(0)},X_n\right.\right) \\ \textit{Step 2. Draw } \theta_2^{(1)} \sim P\left(\theta_2 \left|\theta_1^{(1)},\theta_3^{(0)},\cdots,\theta_B^{(0)},X_n\right.\right) \\ \vdots \\ \textit{Step B. Draw } \theta_B^{(1)} \sim P\left(\theta_B \left|\theta_1^{(1)},\theta_2^{(1)},\cdots,\theta_{B-1}^{(1)},X_n\right.\right) \end{gathered}$$

When arriving at Step B, we go back to Step 1 and repeat for a predefined number of iterations, L. Notice that a draw is made based on the current values of the parameter θ in the next loop and the input data X_n is known. A Gibbs sampler sequentially generates a set of random variables, $\boldsymbol{\theta}^{(j)}, j = 1, 2, \ldots, L$. After a certain number of iterations (L), we can refer to the samples from the Gibbs sampler as a sample from the joint posterior distribution of $\boldsymbol{\theta}$. Hence a set of samples drawn from $P\left(\boldsymbol{\theta}|X_n\right)$ is finally used for Bayesian inference. Specifically, Gibbs sampling is the most widely employed sampling algorithm for solving Bayesian estimation problems dealing with multivariate parameter sets since these are naturally sampled component-wise. In this thesis, we employ Gibbs sampling

to solve the problem of clustering assuming underlying multivariate normal distributions via sampling from conditional posterior distributions as we show in Chapter 4.

In this chapter, we have introduced the input data as well as basic mathematical and statistical tools. We have also given a brief review of RANSAC for parametric model fitting as well as Gibbs sampling as a parameter estimation and inference algorithm in a Bayesian setting.

Chapter 3

Related Work

The most accurate 3D assets are still obtained nowadays from skilled and talented visual effects artist professionals who expertly use tools like Nuke, Maya, and After Effects, to create real world scenes that fulfill, e.g., a film director's vision. Even though a good modeling quality is ensured, it may, however, be rather time consuming especially for large scenes, and often require many hours of manual labor. Consequently, the desire for fully automated modeling approaches employing the strength of modern computing and the advances in algorithmic development has increased. An interesting alternative concerning both efficiency and cost is to use 3D point clouds obtained by LiDAR scanning or SfM/MVS as input to algorithms that generate 3D content.

Processing 3D point clouds of urban scenes is an active area of research focusing on (a) a significant amount of data reduction for, e.g., real-time rendering applications, (b) the automation of object segmentation, detection and reconstruction from unstructured 3D point clouds, (c) the integration of semantics into buildings in Building Information Modeling (BIM) systems and 3D city models, and (d) modeling and 3D content creation (Poullis and You 2009, Lin et al. 2013, Pham et al. 2014, Straub et al. 2014, Monszpart et al. 2015).

Classical modeling workflows for creating 3D content from point clouds, employing triangular meshing algorithms, e.g., Poisson reconstruction (Kazhdan et al. 2006), Marching cubes or greedy projection (Marton et al. 2009), typically over-fit the data and produce overly complex triangulated meshes. They are not suitable for real-time rendering and maintain little or no semantics of the original real world scene (Figure 3.1). Mesh decimation algorithms can be used to partially remedy these problems. However, this usually results in the loss of important geometric detail often decisive for the semantics. Moreover, meshing algorithms often fail in the presence of noise and for high point density variation, e.g., as is often the case for the point clouds of urban scenes derived from images by SfM/MVS. As alternative to meshing, many works rely on specific characteristics of the input datasets, e.g, assume Gaussian noise, a low noise level or moderate clutter. The processing has mainly focused on the analysis of 3D point clouds from terrestrial and airborne LiDAR datasets. The two major reasons for this choice are (1) the maturity of the LiDAR technology as opposed to SfM/MVS and (2) the resulting high accuracy and very low noise level of LiDAR point clouds of large urban scenes.

Nevertheless, due to the general availability of digital cameras, there is a huge interest in the generation and processing of 3D point clouds from images by SfM/MVS (Snavely et al. 2006, Frahm et al. 2010, Huang et al. 2019). The pipeline is being continuously improved by the development of new algorithms, e.g.,

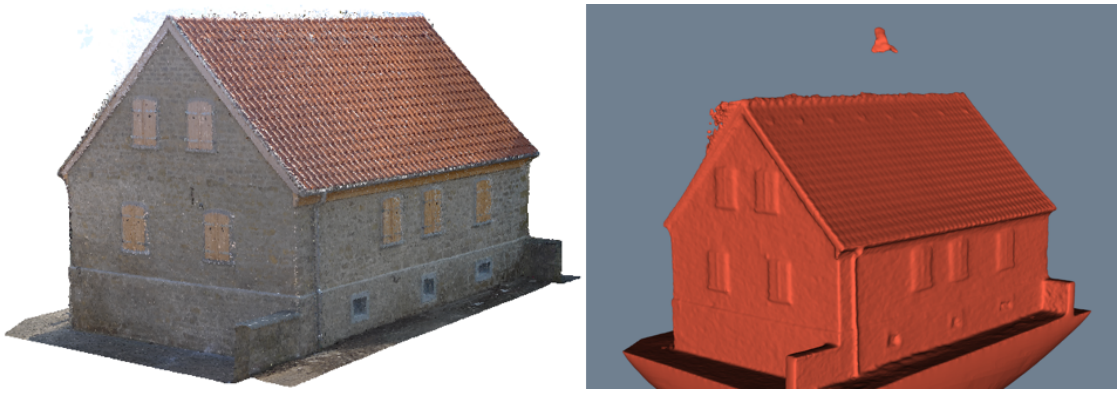

Figure 3.1: Automatic model creation from point clouds using classical meshing algorithms has been widely employed for small objects in indoor scenes. In this example, we applied Poisson reconstruction to point clouds from an SfM/MVS pipeline. The point cloud was manually filtered and everything but the immediate surrounding (left) cut-off. Besides lack of semantic information, the reconstructed triangular meshes (right) over-fit the scene capturing all fine details.

for MVS (Strecha et al. 2008, Goesele et al. 2006, Hirschmüller 2008, Kuhn et al. 2017). Currently they show qualitatively quite impressive results and quantitatively they can achieve sub-millimeter accuracy for small objects. Yet, purely SfM/MVS-based techniques are arguably not yet ready to replace LiDAR systems for highly accurate, large-scale 3D data acquisition for urban scenes. However, the 3D data are good enough to reliably recognize objects and other structural details present in the scene via visual inspection. Furthermore, the algorithms also benefit from the inherent flexibility by using a standard consumer camera. Few recent efforts have been devoted to create 3D models of urban scenes using 3D point clouds derived from images by SfM/MVS that are devised for cluttered outdoor scenes, especially for high-resolution images or a large number of images (Verdie et al. 2015, Nguatem and Mayer 2016, Nguatem and Mayer 2017, Zhu et al. 2018). We see a trend towards an increasing quality of the 3D point clouds derived from images by SfM/MVS and, thus, a necessity for devising methods to create models from point clouds, which work well irrespective of data acquisition technique. Previous efforts for processing 3D point clouds and creating 3D models can be divided into four categories: Semi-Automatic Modeling, Procedural Modeling, Geometric model fitting, and Classification which are discussed in the following.

3.1 Semi-Automatic Modeling

Efforts have been made to improve the creation of 3D models using semi-automatic approaches (Demir et al. 2015, Nishida et al. 2016). The goal is to bridge the gap between traditional carefully handcrafted 3D models from content creation artists and

fully automatic systems. However, the creation of high-quality accurate 3D models for large urban scenes still remains a notoriously difficult task that often requires many hours of skilled labor using dedicated software tools.

Figure 3.2: Derivations from the city grammar targeted at skylines of a whole city block (Talton et al. 2011)

3.2 Procedural Modeling

Though not directly using 3D point clouds as their input, procedural modeling of urban scene is often employed. Together with shape grammars and reversible jump Markov Chain Monte Carlo (rjMCMC) sampling, procedural modeling algorithms have been widely used to generate 3D shapes of man-made structures as well as vegetation (Dick et al. 2004, Müller et al. 2006, Ripperda and Brenner 2006, Talton et al. 2011, Hou et al. 2016, Nishida et al. 2016) in urban scenes. The approach is very flexible, with the following major ideas:

1. Develop a set of basic shapes which act as primitives of objects in the scenes.
2. Define production rules which further shapes can evolve from the primitives.

With an assumption of strong structural regularity, this approach can produce very impressive results, e.g., for city type skyscrapers scenes (Talton et al. 2011) as can be seen in Figure 3.2. However, for large-scale scenes with a rich diversity of buildings and an irregular building placement (as is often the case in urban areas), these algorithms suffer from a lack of scalability. Additionally, the convergence of the Markov chain is difficult to ascertain in finite computational time, because the basic shapes might not have captured the diversity present in the scenes. It is often difficult to derive good proposal distributions for the Markov chain that generalize and are suitable (well calibrated) for the target distribution, i.e., avoid the phenomenon called persistent rejection

within rjMCMC. Furthermore, increasing the set of predefined primitive shapes to produce shapes not within the vocabulary results in an exponential increase in the sample space to be explored by the Markov chain as well as computational time.

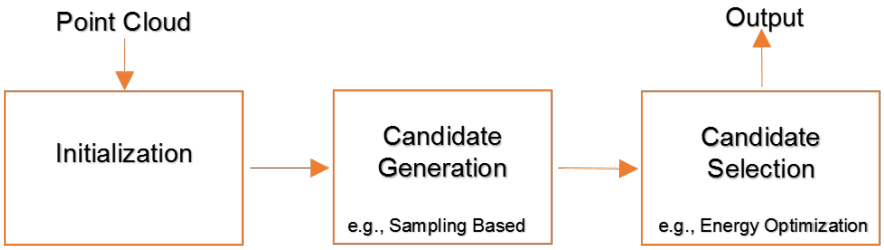

Figure 3.3: The processing pipeline behind most approaches for fitting geometric primitives to 3D data often works in three stages: (i) Initialization, (ii) Candidate. Generation; and (iii) Candidate Selection.

3.3 Fitting Geometric Primitives

This alternative to procedural modeling is often used for urban scene segmentation and modeling from LiDAR point clouds (Poullis and You 2009, Zhou and Neumann 2009, Lafarge and Mallet 2012, Zhou and Neumann 2013, Oesau et al. 2015). A cost function is formulated to assess the quality of the fit (see Figure 3.3). The algorithms mostly operate in three different modes: Region growing (Pu and Vosselman 2009, Musialski et al. 2013, Poullis 2013), RANSAC (Fischler and Bolles 1981, Chum and Matas 2005, Schnabel et al. 2007, Raguram et al. 2013) and energy-based (Isack and Boykov 2011). However, point cloud segmentation alone only gives a partial solution, as it does not address the generation of a well-behaved and watertight 3D surface model required, e.g., for real-time rendering.

A few recent approaches simultaneously address geometric primitive detection and regularization for modeling 3D data (Chauve et al. 2010, Zhou and Neumann 2012, Lin et al. 2013, Verdie et al. 2015, Nan and Wonka 2017). The major idea is very similar to procedural modeling and is illustrated in Figure 3.4: The scene is segmented using RANSAC. Assuming structural regularity many shape primitives, e.g., planes or boxes are generated to represent hypotheses of building parts and using a predefined energy function, the optimal fit to the data is assessed. The latter is used to select which model fits best to the data. However, these approaches only handle LiDAR or synthetic 3D data of moderate size and do not incorperate the ground, let alone the possibility of non-planar terrain. Some approaches even require hardware acceleration units (GPUs) (Oesau et al. 2015, Verdie et al. 2015) or already triangulated 3D points (Verdie et al. 2015, Zhu et al. 2018). Structural regularity also plays a vital rule in this line of research. In its simplest form, only strictly parallel or orthogonal primitives are considered. However, custom constraints can also be considered (Monszpart et

al. 2015). In addition of being mostly suitable only for single isolated buildings, it is important to note that the output of these algorithms is without semantics.

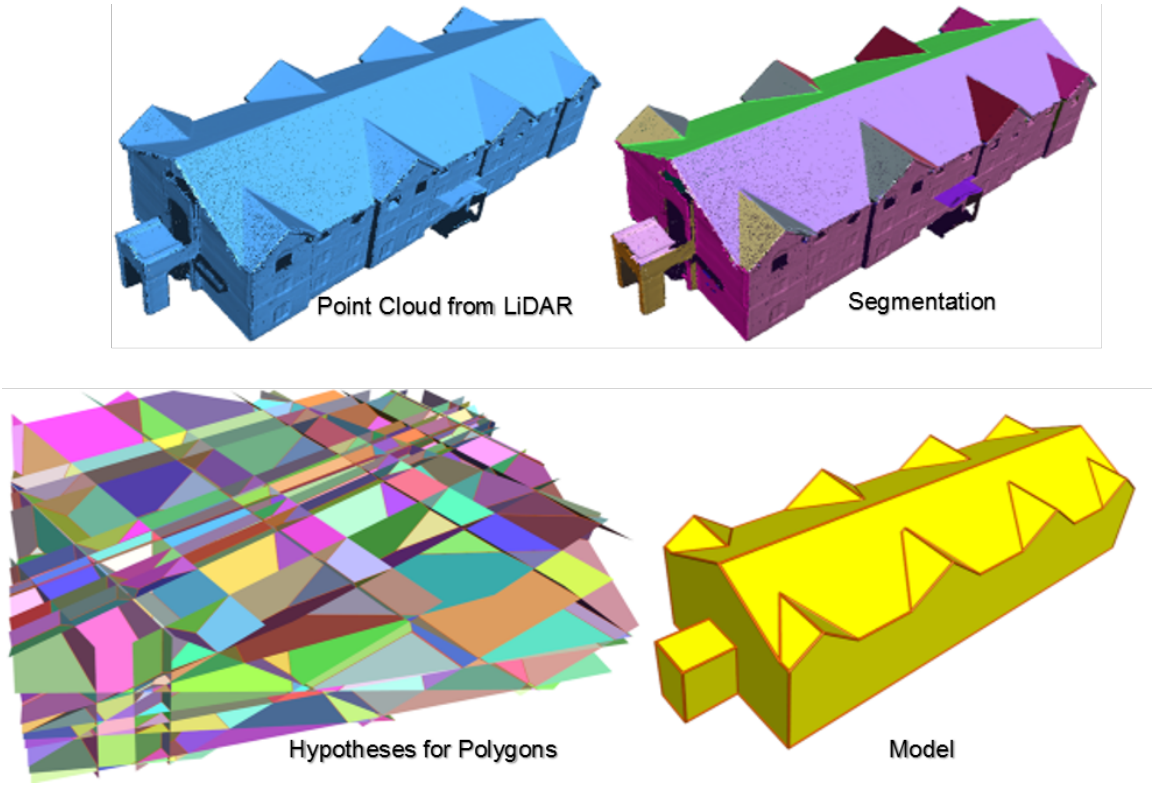

Figure 3.4: Modeling workflow described in (Nan and Wonka 2017). The input point cloud is segmented into planes using RANSAC. Hypotheses for polygons are generated using pairwise intersection of faces and assessed by an energy function. The optimal model is chosen. Planar faces and segments are randomly colored.

3.4 Point Cloud Processing Using Deep Learning

Deep learning has recently become very successful in solving many machine perception problems. Correspondingly, efforts have been made to process 3D point clouds with deep networks. Besides a focus on classification, algorithms to model point clouds directly also have been pursued. The idea is to transform an unstructured point cloud to structured data on which neural networks can be trained similarly to 2D image grids. (Zeng et al. 2018) combines deep neural networks with a procedural algorithm. In standard procedural modeling, randomization plays a key factor to produce new shapes. In contrast to this, after data transformation, the major idea of (Zeng et al. 2018) is to infer fixed rules for shapes defining a shape grammar and classification determines the rule to be applied to produce CAD-type models. Geometric attributes for the CAD

models are regressed from the input 3D points producing the coefficients of the models. A key disadvantage of this approach is the huge processing time as well as the lack of suitable training data to capture the diversity of urban architectures, particularly as deep learning has a need for a large amount of training data to achieve a high accuracy.

3.5 Point Cloud Classification

Assigning labels to individual 3D points is widely used for point cloud data processing (Rusu and Cousins 2011, Weinmann 2016, Landrieu and Simonovsky 2018). These approaches usually take a supervised learning pipeline, i.e., rely on the availability of correctly labeled data, computation of discriminative features as well as a learning algorithm. Most of the employed discriminative features are based on estimated surface normals or curvature values of the individual 3D points as well as intensity values if present. Often, a planar ground is assumed. However, a label for every single point in the cloud, does not necessarily lead to contiguous patches and watertight polygonal models constituting objects in the scene. Furthermore, classifying 3D points of large scale urban scenes derived from images by SfM/MVS is itself a non-trivial task.

3.6 Modeling Beyond a Manhattan-World

The assumption of a mixture of Manhattan-Frames is widely used both implicitly (Monszpart et al. 2015, Armeni et al. 2016, Li et al. 2016b) and explicitly (Straub et al. 2014, Li et al. 2016a) for processing 3D point clouds. A major topic is the number of frames present in the scene. In the simplest case, there is just one frame present, however, even if the number of frames is known, the goal remains to search and associate points to the respective frames. Work based on the Manhatten-World assumption can only handle planar surfaces including the ground, hence, there is a strong planarity assumption. Therefore, it is suitable only for indoor scenes (Mura et al. 2016, Armeni et al. 2016). A flexible approach offering free-form surfaces by fitting NURBS to pre-segmented patches of LiDAR point clouds is proposed in (Dimitrov et al. 2016). The major challenges remain, namely, how to robustly segment and how to infer unique and consistent control points of the NURBS surfaces in the presence of substantial noise, large point density variations and missing data as is often the case for point clouds derived from images by SfM/MVS.

We see a need for algorithms that directly model urban scenes from unstructured 3D point clouds, which are robust, scalable and independent of the data acquisition modality. In addition, the expected output from these algorithms should

1. Maintain the scene semantics, thus, achieving a compact yet rich representation for further processing and

2. Generalize the much too restrictive planar ground assumption underlying

the Manhattan-World as well as planar patch segmentation and regularization (Monszpart et al. 2015, Oesau et al. 2015).

Chapter 4

Clustering as a Mixture of Gaussians

A major problem for which a solution is presented for the purpose of this thesis is to group data items in such a way that all items in the same group are in some sense more similar to one another than those in other groups—a common problem called clustering. Clustering is a very popular problem in computer science and data mining. The text in this chapter loosely follows (Nguatem and Mayer 2016, Nguatem and Mayer 2017) and introduces data clustering under a mixture of Gaussians assumption. Our approach uses Gibbs sampling introduced in Section 2.4 to cluster data items. This bridges the gap between the basic algorithms described in Chapter 2 and their application to normal clustering on the unit sphere presented in Chapter 5.

4.1 Finite Mixture Clustering

Clustering is a major problem in statistical data analysis with many algorithms adopting a model-based approach, assuming a certain model for clusters and attempting to optimize the fit between data and model. In practice, each cluster is mathematically represented by a parametric distribution, e.g., a Gaussian or a Poisson distribution. This leads to modeling the entire dataset by an ensemble—a mixture of these distributions. An individual distribution used to represent a specific cluster is referred to as a component distribution of the mixture. We use a Gaussian mixture model-based approach to cluster data items by attempting to optimize the fit between the data and Gaussian distribution components. A component distribution can be fully specified by its mean and covariance denoted by μ_a and Σ_a, respectively. If we assume that the number of components k is known and the data has equal variance and uncorrelated $\Sigma = \sigma I_d$, where I_d denotes the d-dimensional identity matrix, the data generation process can be represented as follows

$$x_i \sim \frac{1}{k} \sum_{a=1}^{k} \mathcal{N}\left(\mu_a, \sigma I_d\right)$$

where $k \in \mathbb{N}$, $\sigma > 0$ and the mixture components are normal distributions with means μ_a and covariance matrices σI_d. Since k is known and equal variance as well as no correlations are assumed, the above setup can be solved using k-means clustering which iteratively associates data items to cluster centers. While the k-means algorithm is simple and very efficient, it is not able to handle overlap as well as spherical clusters, i.e.,

data items must have identical variances. This is because the data association considers Euclidean distances which do not prefer one direction over another when $d > 1$.

An obvious next step is to employ the more flexible finite mixture of Gaussians model that learns a Gaussian distribution per cluster explaining how a cluster looks like. In addition to a mean, a finite mixture model of Gaussians learns covariances which explain the spread of the data items in different directions. The data generation process for a finite Gaussian mixture model with k components can be written as:

$$x_i \sim \frac{1}{k} \sum_{a=1}^{k} \mathcal{N}\left(\mu_a, \Sigma_a\right)$$

Finite Gaussian mixture models provide a number of advantages for clustering:

- They are flexible because they can account for non-spherically shaped clusters.
- They provide a means for choosing k.
- They are less sensitive to data scaling.

This approach to data analysis is comprehensive, yet still very limited for many real world scenarios. For example, it is impossible to know the number of facades, roof elements, as well as their orientations in point cloud data of an urban scene. This implies that it is not possible to know a-priori the number of clusters present in the data, i.e, the structure underlying the data. Various information criteria have been developed such as Akaike Information Criterion (AIC), Bayes Information Criterion (BIC) and their modifications such as quadratic AIC/BIC, the Integrated Classification Likelihood criterion (ICL), Normalized Entropy Criterion (NEC) and Minimum Information Ratio criterion (MIR). They are used to test different competing structures—a process called model selection. In general, these methods can be easily implemented, but share one shortcoming: It is difficult to obtain a meaningful comparison of the model fit from one situation to another. An alternative is to use rjMCMC algorithms. These are however computationally expensive, difficult to set up, and usually hardly explore the sample space of possible solutions.

4.2 Infinite Mixture Clustering

In the previous section, we assumed that the number of mixture components is fixed and finite. An approach addressing the challenges posed by a lack of complete information about the number of components is to include it as a latent variable. We illustrate this approach to clustering in a Bayesian setting. The main aim is to generate samples from the posterior distributions of the means or the means and the covariance matrices in Equation (4.2) from the conditional distributions of $\mu_1, \ldots, \mu_k$ and $\Sigma_1, \ldots, \Sigma_k$ respectively. This target distribution has the density

$$P\left(\Theta | x_i\right) = \frac{P\left(x_i | \Theta\right) P\left(\Theta\right)}{P\left(x_i\right)}$$

where $\Theta=\mathbb{R}^d\times\mathcal{S}_d^+$ with $\mathcal{S}_d^+$, the space of positive definite d-dimensional symmetric matrices and x_i are the data items to be clustered represented in $\mathbb{R}^d$ dimensional space.

The Gibbs sampling algorithm described in Section 2.4 can be used to generate samples from this density. It updates each variable in turn from the conditional distribution, given all estimates for other variables in the system.

In contrast to the finite mixture models presented above, an infinite number of underlying structures, i.e., $k \to \infty$ can potentially better explain the data. For Bayesian statistical analysis, how can we place prior distributions on an infinite number of probability values? The Dirichlet process plays a major role in answering this question. In the following section, we describe the Dirichlet process and present its application to clustering as used within this thesis. We also discuss an approach to do inference with the posterior in this setting.

4.3 Dirichlet Process and Mixture Model Clustering

The Dirichlet Process (DP) introduced by Ferguson (Ferguson 1973) is a probability distribution on the space Θ of discrete probability distributions. It can be used as a prior distribution within a Bayesian setting for data clustering. In this thesis, we use the DP to cluster unit normals. Hence, this section gives a brief introduction to this form of data analysis – the clustering of data items in a Bayesian nonparametric setting (Mueller et al. 2015). The Dirichlet Process (DP) is parameterized by α, a positive real number called scaling parameter, and G_0, the base probability distribution. To draw a sample from the DP on a given probability space, say Θ, we use,

$$G\sim DP\left(\alpha,G_0\right). \tag{4.1}$$

Because realizations from the DP are discrete, they are usually convolved with a smoothing kernel. This leads to a (non-parametric) prior over smooth probability densities (the kernel) and is called the Dirichlet process mixture (DPM) model (Lo 1984). The following hierarchical equations describe a **DPM**,

$$\begin{aligned} &G\sim DP\left(\alpha,G_0\right)\\ &\theta_i\sim G\\ &x_i\sim f\left(x,\theta_i\right), \quad i=1\cdots n, \end{aligned} \tag{4.2}$$

where $f\left(x,\theta_i\right)$ is a non-negative smoothing kernel on the space $\mathcal{X}\times\Theta$, with $\int_{\mathcal{X}} f\left(x,\theta\right)dx=1$ for all θ. There exist many different priors widely used to represent the DP—stick-breaking process (Ishwaran and James 2001), Chinese restaurant process (Aldous 1985), the Polya urn scheme (Hoppe 1984), etc. In the the following, we discuss the prior used within this thesis to represent a DP, and substantiate our choice for this prior. Usually, in order to make the mathematics more explicit, the base distribution, G_0, is chosen as the conjugate prior of the kernel.

The **DP** induces a clustering effect on data items, with the data items in the eth cluster having parameter θ_e. The cluster parameters define central tendency values (e.g.,

location, spread, skewness) of the kernel and are themselves independently and identically distributed (i.i.d.) samples drawn from G_0, which characterizes the spread of the clusters in parameter space. Figure 4.1 gives a simple example of the clustering effect for 1000 data items in $\mathbb{R}^2$ induced by using two realizations (left and right plot) of a **DPM**. In this example, Gaussian kernels have been used, thus, resulting in a Dirichlet process mixture model of Gaussians in $\mathbb{R}^2$. The five clusters of the left plot have zero covariance. Therefore, the location of the centroids, i.e., the mean, defines the latent variable (the cluster parameter) for each cluster. Hence, both Θ and $\mathcal{X}$ are in 2D Euclidean space, $\mathbb{R}^2$. Conversely, the right plot shows three clusters with varying covariance matrices as well as varying means across clusters. Therefore, $\Theta{=}\mathbb{R}^2{\times}\mathcal{S}_2^+$, where $\mathcal{S}_2^+$ is the space of positive definite 2D symmetric matrices. Here, the conjugate normal-inverse Wishart (NIW) distribution is often used as the base probability distribution, G_0, (Görür and Rasmussen 2010). Similarly, the algorithm presented in the next chapter also uses NIW as the base probability distribution, G_0. The task is to cluster surface normals (our data items) in $\mathbb{R}^3$. We use the Chinese Restaurant Process (CRP) (Aldous 1985) because of its simplicity and plausibility as a prior to represent a **DP**. It assigns data items to clusters in a way that clusters grow in proportion to their sizes[1]

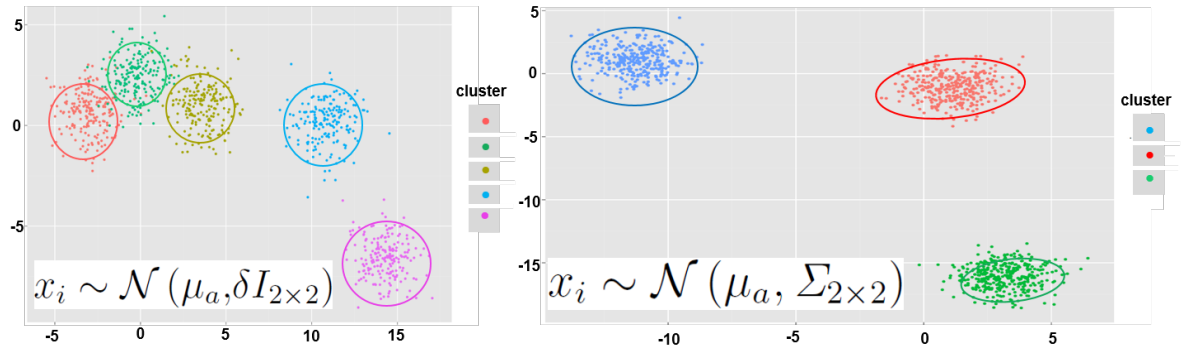

Figure 4.1: Two realizations of a DPM model of Gaussians in $\mathbb{R}^2$. The latent variables are the means (left), and the means and the covariances (right).

Mathematically, this assignment is given by,

$$p(z_n{=}e\,|z_{1:n-1}) = \begin{cases} \frac{m_e}{n-1+\alpha} & \text{if } m_e{>}0 \quad \text{(i.e., } e \text{ is old)} \\ \frac{\alpha}{n-1+\alpha} & \text{if } m_e{=}0 \quad \text{(i.e., } e \text{ is new)} \end{cases}, \tag{4.3}$$

where m_e is the number of data items assigned to cluster θ_e, $\alpha{\geq}0$, the positive real number called scaling parameter of the DP and n the total number of data points. The latent variable z_n defines our cluster assignment[2]. Since we assume that our data items

[1] In analogy to a customer choosing a table at a Chinese Restaurant. Those tables with larger number of persons are more likely choices for the nth customer to arrive at the restaurant.

[2] Traditionally, z, is widely used as membership assignment variable (Basu et al. 2008, Airoldi et al. 2014, Hennig et al. 2015)

(surface normals in $\mathbb{R}^3$) are from a multivariate normal distribution with mean vector μ, and covariance matrix Σ, the detailed data generation process revealing the specified priors for the DP, the CRP, and the G_0 is defined as follows,

$$\begin{aligned} &\theta_{1,2,\ldots} \sim G_0 = NIW\left(\mu_0, \kappa_0, \Psi_0, \upsilon_0\right) \\ &z_{1:n-1} \sim CRP\left(\alpha\right) \\ &x_{i=1,2,\ldots n-1} \sim \mathcal{N}\left(\mu, \Sigma\right), \end{aligned} \tag{4.4}$$

where $NIW\left(\mu_0, \kappa_0, \Psi_0, \upsilon_0\right)$ is the NIW distribution with its four parameters and $\mathcal{N}\left(\mu, \Sigma\right)$ the normal distribution. For a specific parameter estimation problem, e.g., data clustering, the ultimate goal is to infer the cluster assignments given the data items and the four parameters of the NIW, i.e., $p\left(z_{i:n} \,|x_{i:n}, \mu_0, \kappa_0, \Psi_0, \upsilon_0\right)$. To this end, we use Gibbs sampling (Neal 2000). The algorithm updates each variable in turn, i.e., a variable-at-a-time approach, by drawing samples from its approximate posterior predictive distribution conditioned on all other variables. In the next chapter, we present our complete solution for clustering unit normals in Algorithm (1).

Chapter 5

Large-Scale Patch Segmentation in Point Clouds

The first step of our system for modeling urban scenes from point clouds partitions the input point cloud into distinctive units, hence point cloud segmentation. The text in this chapter is weakly based on (Nguatem and Mayer 2016, Nguatem and Mayer 2017). There exist many different definitions of object segmentation in point clouds, differing in how the location of an object in the point cloud is represented, e.g., by a bounding box, a plane with associated inliers or by a set of contiguous groups of points representing patches. In general, the segmentation of a single planar object in a point cloud can be done in a brute-force manner, for example by simply picking three random points, estimating the parameters of a plane from them, and then scoring points according to this model using a suitable threshold metric in a RANSAC framework (cf. Section 2.3) as presented in (Schnabel et al. 2007). This needs no data preprocessing and works in a straightforward manner on simple datasets with mostly isolated object instances. Yet, it is bound to fail on moderate to more complex environments, since a resulting model might contain points located on different objects. Furthermore, the approach underlies a stringent planarity constraint which is often not adequate, e.g., for the natural ground.

Figure 5.1: Building walls, roof elements, and the non-planar ground (right) segmented by our algorithm in a point cloud (left: 46 million points) derived from images by Structure from Motion/Multiview Stereo (SfM/MVS). The problem is formulated as probabilistic inference that robustly segments all patches in the scene.

.

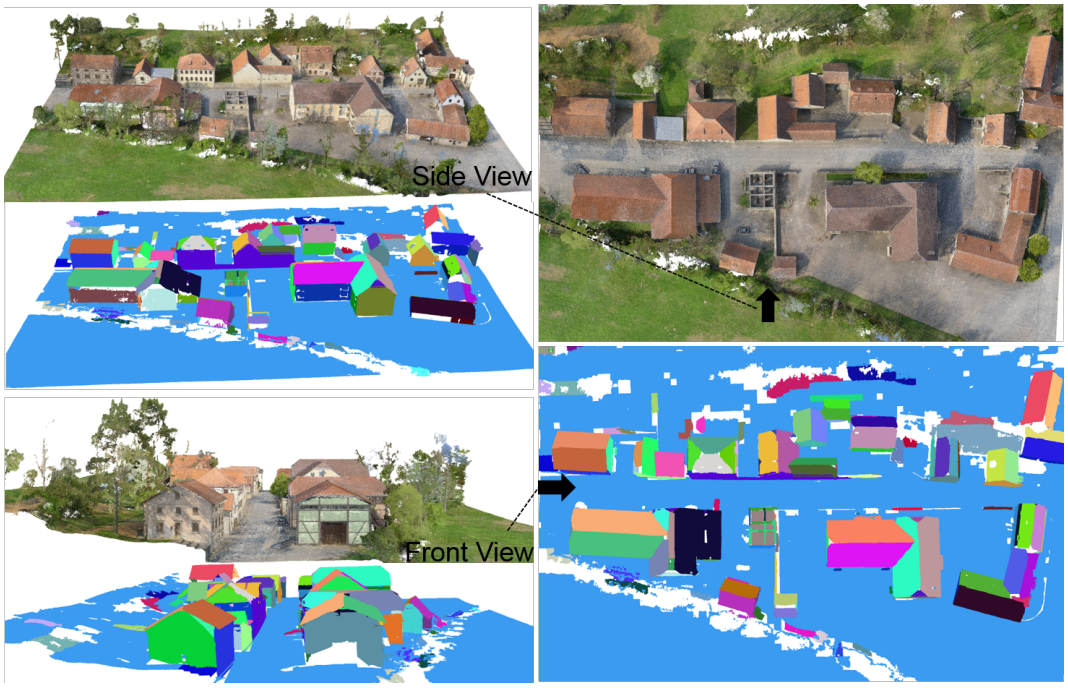

Figure 5.2: Besides accurate scene segmentation into patches of different sizes, our proposed algorithm is robust against noise, e.g., vegetation or reconstruction artifacts from SfM/MVS, and estimates accurate boundaries for each patch. Here, 76.7 million points acquired from images by SfM/MVS reconstruction of a residential area are accurately segmented (patches are randomly colored).

We therefore, present the large-scale contiguous patch segmentation algorithm (Nguatem and Mayer 2016) that resolves many of the above problems. It takes the complexity of the environment into account and considerably speeds up the segmentation and aims at finding patches in the point cloud $\mathcal{D}$, which exhibit contiguous local planarity, i.e., are planar within a predefined small radius. Algorithms based on region growing, RANSAC and energy optimization have been widely used to solve similar problems (Schnabel et al. 2007, Isack and Boykov 2011, Oesau et al. 2015). Yet, these approaches to patch segmentation in point clouds have major deficits: They lack scalability, robustness against substantial noise and are tailored only for point clouds from LiDAR scans. Furthermore, some of them underlies a Manhattan-World assumption (see Chapter 3). Conversely, our algorithm uses a probabilistic approach to segmentation that combines the strength of RANSAC and robustness of Bayesian statistics to segment challenging real world scenes such as the ones shown in Figures 5.1 and 5.2.

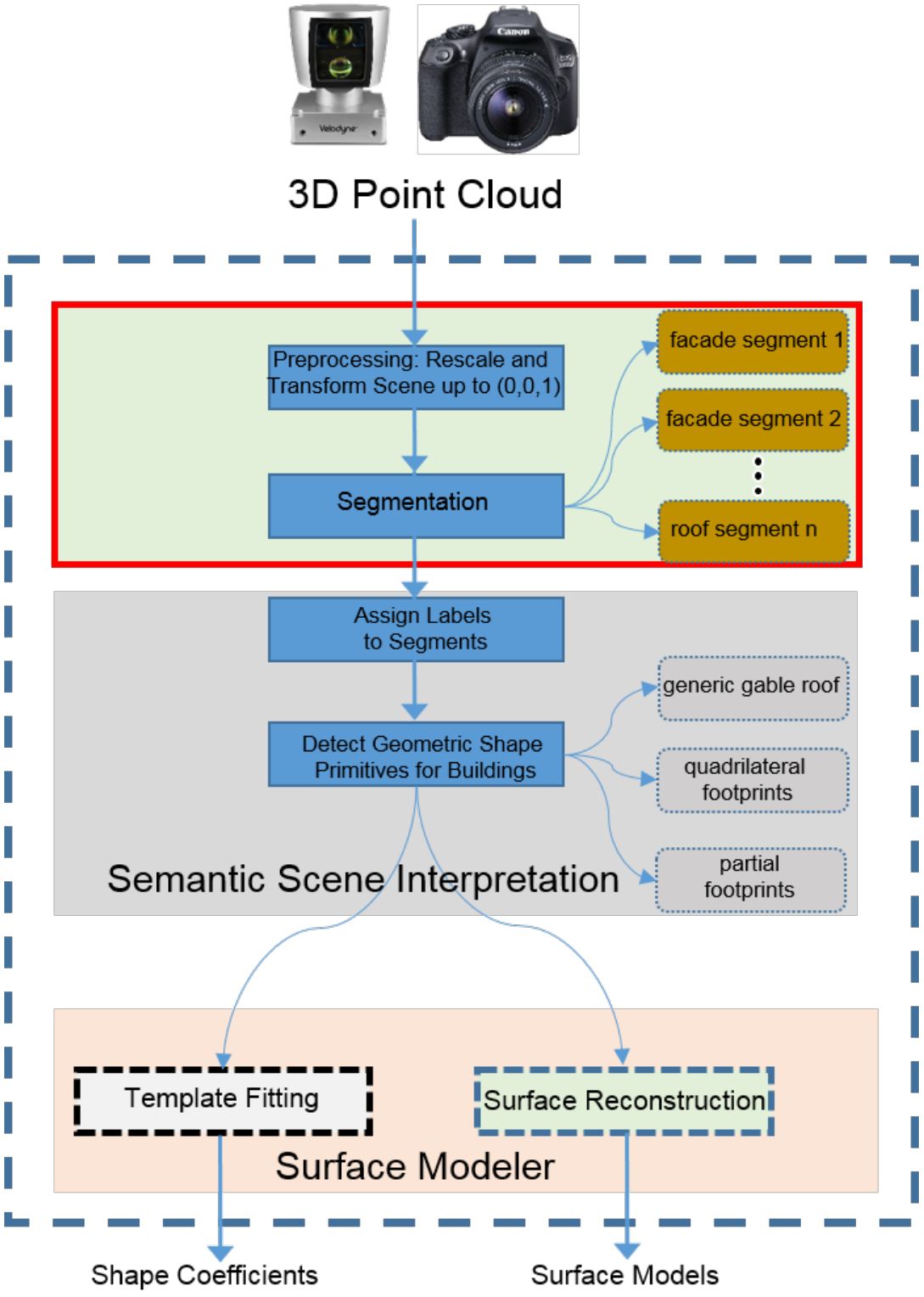

Figure 5.3: Architecture: Framework – Segmentation

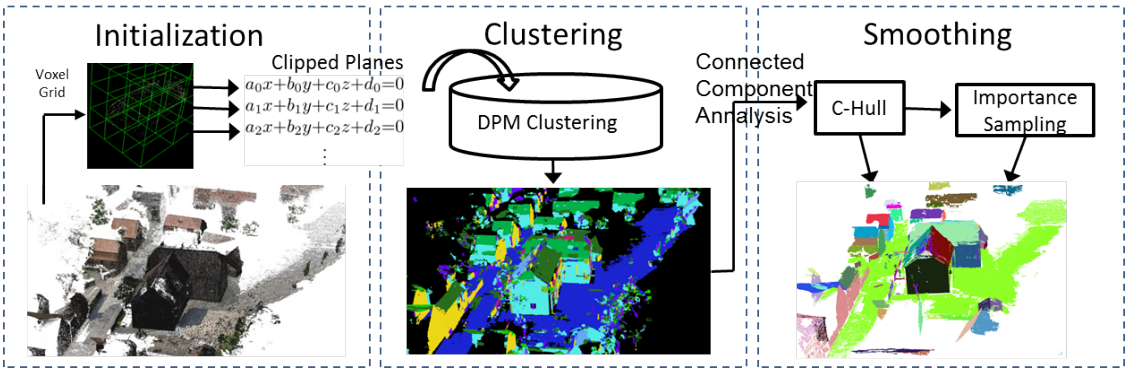

Figure 5.4: Algorithm Overview – The proposed system segments point cloud data by extracting contiguous patches as well as the boundaries of objects present in the scene. It consist of three sub-modules namely, (i) Initialization, (ii) Clustering, and (iii) Smoothing (Nguatem and Mayer 2016).

Patch segmentation in point clouds is formulated as an inference problem. We use non-parametric Bayesian methods, represented through a Dirichlet process, to cluster a mixture of Gaussians followed by stochastic smoothing to resolve the problem of patch segmentation. In Chapters 2 and 4, a brief introduction to statistical theory has been presented with a focus on data clustering using Dirichlet Process Mixtures (DPM). With this as foundation, we sketch our large-scale patch segmentation algorithm in its entirety in the next Section.

5.1 Overview of the Segmentation

Segmentation is the first stage of our modeling pipeline as presented in Figure 5.3. It starts from an unstructured 3D point cloud $\mathcal{D}$ as input and aims at partitioning $\mathcal{D}$ into a set of contiguous patches with associated object boundary estimates. Here, we defined contiguous patches as areas of $\mathcal{D}$ which are locally planar, e.g., roof elements, walls, table tops, the ground. While, the natural ground can be highly non-planar when perceived globally, it is usually planar within a local neighborhood. Our segmentation algorithm assumes that the metric scale of the input point cloud $\mathcal{D}$ is known. It is based on a divide-and-conquer paradigm, and consist of three main sub-modules depicted in Figure 5.4: (i) Initialization, (ii) Clustering, and (iii) Smoothing. During initialization, the input point cloud $\mathcal{D}$ is divided into non-overlapping fixed-size voxels and RANSAC is used to fit a plane within each voxel. In the clustering sub-module, clusters of neighboring voxels are built using the DPM model of Gaussians. For each inferred cluster, connected components are estimated to produce contiguous regions. These regions are smoothed using Importance Sampling (IS) as an optimization. The smoothing produces our final output which does not only give the locations of contiguous patches, e.g., table tops, roof elements, walls and the ground, in point clouds, but also estimates their boundaries (see Figures 5.1 and 5.2). Next we present the three sub-modules of the proposed

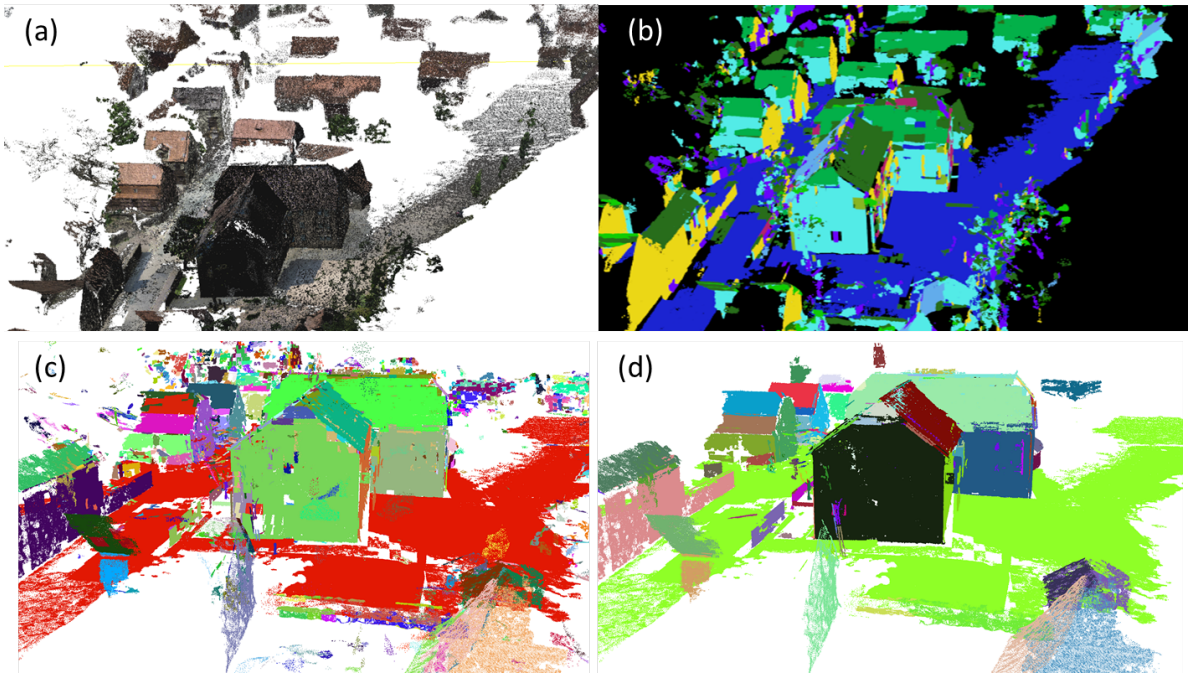

Figure 5.5: The input data in (a) contains 47M 3D points from SfM/MVS reconstruction. Clustering ambiguity when not using the directionality can be seen in (b). Parallel surfaces belong to the same cluster and have the same color. Connected component analysis resolves this ambiguity (c). Finally, the resulting regions are smoothed (d).

segmentation algorithm in detail.

5.2 Initialization

This sub-module acts as a form of data preprocessor where the ultimate goal is to estimate consistent unit normals describing the surface geometry present in the scene. Eigenvalue analysis of covariance matrices has been widely employed for similar tasks as explained in Section 2.1.2. However, it is computationally expensive for point clouds of large-scale urban scenes and produces surface normals which are not a good representation of the underlying surface geometry particularly in the presence of substantial noise, as is often the case in point clouds from images by SfM/MVS. Thus, we make use of the fact that urban scenes are largely planar within a local neighborhood. We divide $\mathcal{D}$ into non-overlapping voxels of fixed size and compute an adjacency graph of voxel (spatial neighborhood) relations, $\mathcal{G}_{adj}$. The voxel leaf size, l_s, is chosen such that the smallest object of interest, e.g., roof superstructure, is greater than l_s. This ensures that every patch of interest is captured by at least one voxel. We use RANSAC to fit a plane within each voxel. The planes have local support limited to the voxel bounds,

they are thus clipped planes.

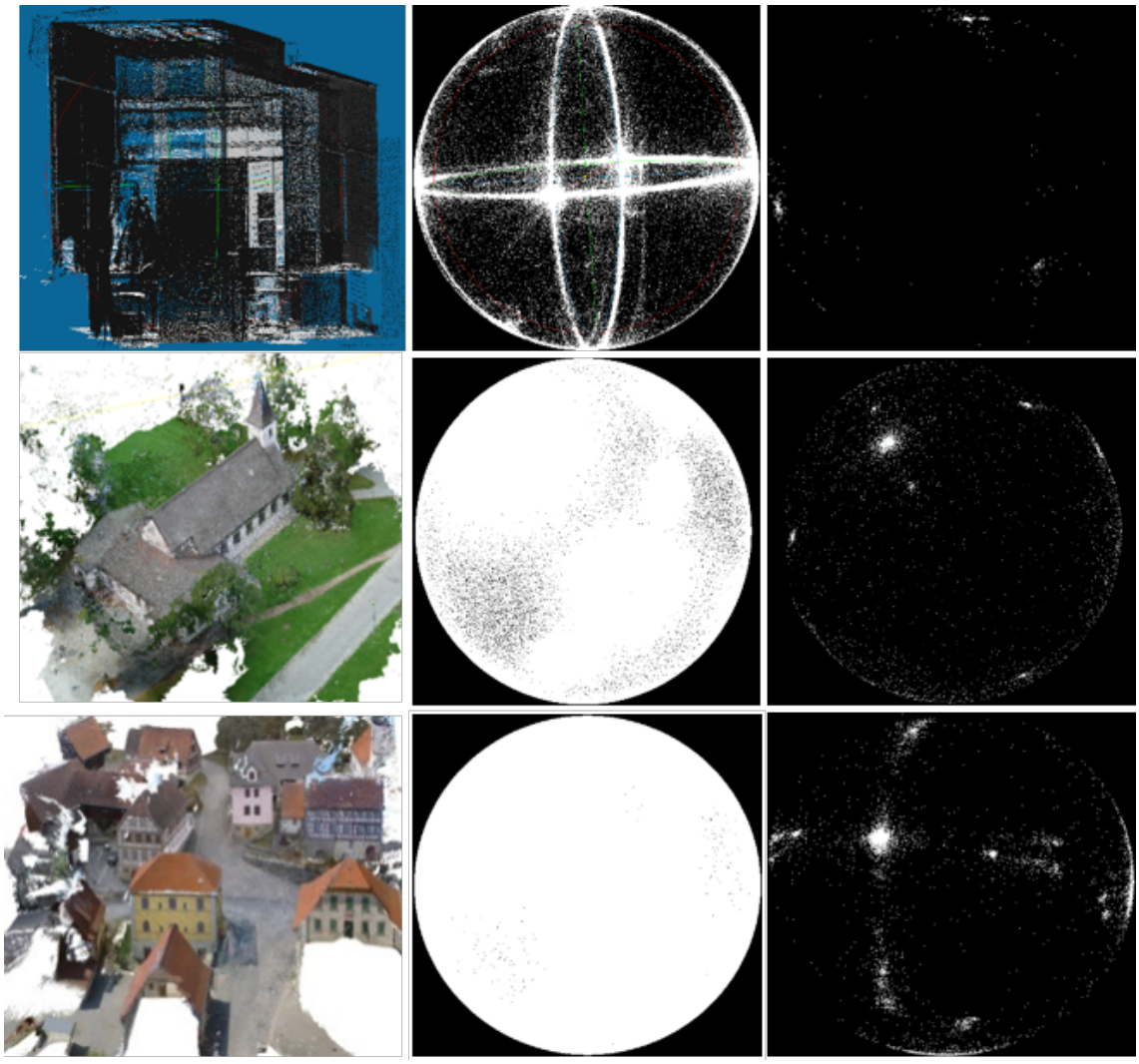

Figure 5.6: Comparison of normals of point clouds from the Kinect sensor of an office scene (first row), and from images by SfM/MVS of urban scenes (second row) and (third row). Normals of individual 3D points can suffer from a phenomenon called wrap-around (directions wrong by 180^0) and can overfit for urban scenes (middle column). Opposed to this, clipped plane normals ($3^{rd}column$) form distinct clusters (Nguatem and Mayer 2017).

Compared to normals of individual 3D points, clipped plane normals (CPNs) offer a number of advantages (cf. also Figure 5.6):

- Surfaces in urban scenes exhibit many small local variations, e.g., roof tiles, which aren't a good representation of the underlying geometry. Therefore, normals of

individual 3D points often overfit. On the other hand, due to the robustness of RANSAC, CPNs are stable and do not tend to overfit.

- CPNs are robust against point density variations and noise, again because of the robustness of RANSAC.
- Normals of 3D points can wrap-around the unit sphere (i.e., orientation is wrong by 180^0) when estimated by traditional eigenvector analysis of covariance matrices. This is due to the influence from neighboring points of adjacent surfaces on a query point. This may be even worse in point clouds with substantial noise, such as those derived from images by SfM/MVS of urban scenes (see Figure 5.6 middle column). In contrast, CPNs do not wrap-around (cf. Figure 5.6 right column), even for large-scale scenes (cf. Figure 5.7), since RANSAC searches and fits a single "best" plane to the inliers, i.e., without influence of points from adjacent surfaces.
- CPN computation scales for large scenes, because it is far less expensive computationally than normal estimation of individual 3D points. Here, RANSAC mostly converges within a few iterations due to the strong local planarity property within most voxels. Furthermore, the number of CPNs is many magnitudes lower than the number of 3D points due to scene voxelization.

The output of the initialization is a set of consistent unit vectors, one for every voxel, representing the "true" underlying surface geometry of the scene. We proceed with clustering CPNs, thus, combining voxels consistently. The clustering algorithm models the distribution of CPNs on the unit sphere as a mixture of multivariate Gaussian distributions. The resulting clusters give information about the number of principal axes in the scene. If the scene is made up of a single building with a flat roof, we can be sure that there will be three clusters showing the three major directions. However, for large-scale urban scenes, with an irregular arrangement of buildings, the number of major axes denoted by the different orientations of walls, roofs and the non-planar ground is unknown. Hence, we employ inference for Gaussian mixture models with an unknown number of components and structure—the parameters of the individual mixture components—using Bayesian non-parametric statistics as introduced earlier in Chapter 4.

5.3 Clustering

The aim of this sub-module is to cluster neighboring voxels into contiguous regions. To this end, a cluster will represent a principal axis in the scene, and can be made up of voxels representing for example, all walls aligned parallel to one another. The input to the clustering algorithm are the estimated CPNs from the initialization sub-module. Unit normals in $\mathbb{R}^3$ are multivariate vectors and directional data. In order to consistently model directional data in a probabilistic framework, it is necessary to employ some form of directional statistics (Straub et al. 2015, Kent 1982, Fisher 1953, Evans et

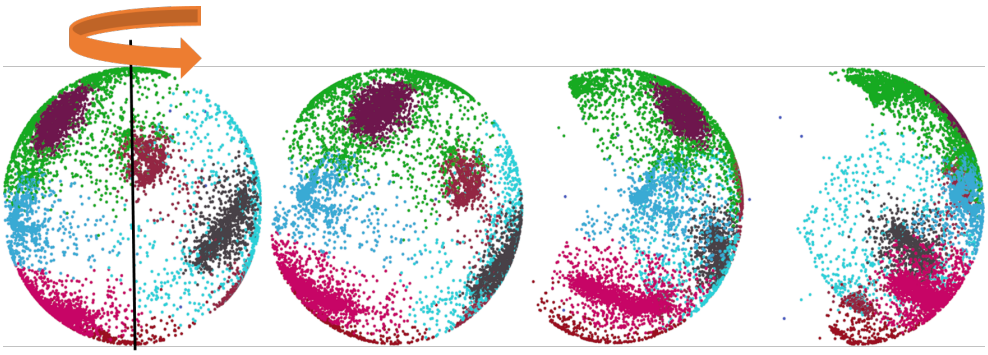

Figure 5.7: Even for large scale urban scenes containing hundreds of millions of 3D points and many buildings, CPNs do not overfit and do not suffer from wrap-arounds. Here, we show 950,000 clustered CPNs of the scene of Figure 8.16. Clustered CPNs are rotated at four viewpoints showing that there are no wrap-arounds.

al. 2000). Yet, as CPNs do not wrap around the unit sphere (e.g, see Figure 5.7), they can be clustered without considering the directionality. Our clustering algorithm is based on Gibbs sampling and presented in Algorithm 1.

Gibbs sampling is fundamentally sequential, i.e., usually requires a good initialization to guarantee fast convergence. Many initialization strategies exist in state-of-the-art clustering algorithms based on Gibbs sampling inference for a mixture model (Mitra and Müller 2015, Phadia 2016):

- Initialization with one data item per cluster
- $log(N_{cpn})$ clusters with random assignments for initialization, where N_{cpn} is the number of data items, i.e., clipped plane normals
- Use of the output of an alternative, yet sub-optimal clustering algorithm, e.g., K-means, as initial cluster assignments

While one data item per cluster seems least likely to be biased based on random early decisions, for large datasets with hundreds of thousands of CPNs it is also the prohibitively slowest. To reduce the computational complexity in the initialization problem, we employ the two level approach to clustering presented in Figure 5.8. On the first level, a subset of the CPNs are clustered using random assignments of N_{init} clusters for initialization. This subset is chosen such that it preserves the salient mathematical relationships within the CPNs, hence we call it the "core set". The inferred underlying primary structure, $\boldsymbol{\theta}_1^0, \boldsymbol{\theta}_2^0, \cdots, \boldsymbol{\theta}_{K^0}^0$, is then used as the initialization for clustering the full set of CPNs containing N_{cpn} data items. A naive way of choosing the elements of the "core set" is to use random uniform downsampling of the full set of CPNs. Since CPNs are not evenly distributed on the unit sphere, this is rather biased. We thus employ stratified resampling to obtain the "core set":

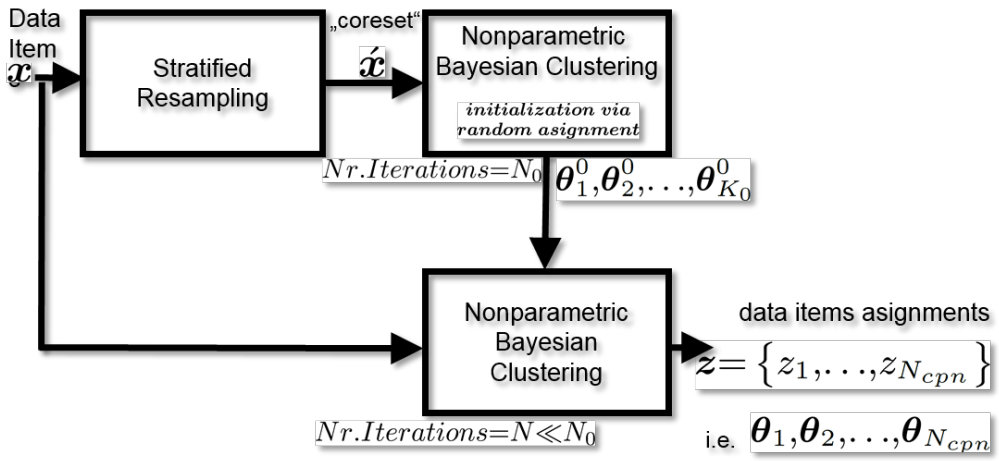

Figure 5.8: Two level clustering of CPNs: The first level clusters only the "core set" – a subset of the data using initial random assignments. The second level takes into account the full data using the output of level one as initial assignments. On both levels, the same clustering Algorithm 1 is employed.

1. Divide the unit sphere into N_p polygons of approximately equal size.
2. Sample $N_{coreset} \ll N_{cpn}$ proportionally to the number of CPNs per polygon.

This ensures a variance reduction in the downsampled data as compared to a naive simple uniform random sampling strategy (Arvo 1995, Yershova and LaValle 2004). The number of polygons, N_p, is empirically set to 300. An example "core set" for the point cloud depicted in Figure 5.1 is shown in Figure 5.9.

The output of the clustering algorithm is ambiguous since parallel surfaces belong to the same cluster (see Figure 5.5b – Parallel surfaces have the same color). Yet, in architectural scenes of urban environments, parallel surfaces within the same cluster are usually far apart, i.e., not connected. To make use of this knowledge, we employ the voxel adjacency graph $\mathcal{G}_{adj}$, taking spatial proximity as a metric to denote connectivity, and perform connected component analysis – CCA (Vosselman 2013) within every cluster to resolve the ambiguities. This two-step approach—probabilistic clustering and deterministic CCA – has two major benefits: It provides a clear distinction between the stochastic and the deterministic part of the clustering algorithm and it speeds up the clustering as a whole, since all parallel surfaces in the scene will share the same θ, cf. Equation (4.4). I.e., the effective sample space of θ is much smaller compared to analyzing the clustering problem with directional statistics (Straub et al. 2015). The result of the CCA is a set of contiguous regions, see Figures 5.5 and 5.10.

Before we proceed to the final sub-module of our large-scale contiguous patch segmentation algorithm, it should be noted that Algorithm 1 leaves several parameters unspecified—the concentration parameter, α, used during the setup of DPM model of

Input : $x_{i:n}$, $G_0{=}NIW\left(\mu_0,\kappa_0,\Psi_0,\upsilon_0\right)$, with CPNs as data points x_i

Output : $p\left(z_{i:n}\left|x_{i:n},\mu_0,\kappa_0,\Psi_0,\upsilon_0\right.\right)$, a set of new cluster parameters $\hat{\theta}^*$

initialize $z_{1:n-1}$, i.e., assign data items to initial clusters θ^*

foreach *Gibbs sampling iteration j* **do**

  foreach *data point i* **do**

    Discard the cluster assignment z_i of the ith data item

    If i belongs to its own cluster, discard θ_{z_i} from θ^*

    Sample a new assignment for i using CRP prior (cf. Eq. (4.3)) as follows

$$p\left(z_i{=}e\left|\theta^*\right.\right)=\begin{cases}\frac{m_e}{n-1+\alpha}f\left(x_i,\theta_e^*\right) & \text{if } m_e{>}0 \text{ (i.e., } e \text{ is old)}\\ \frac{\alpha}{n-1+\alpha}\int f\left(x_i,\theta\right)G_0\left(\theta\right)d\theta & \text{if } m_e{=}0 \text{ (i.e., } e \text{ is new)}\end{cases}$$

    If i is assigned to a new cluster, sample this cluster's parameters from

$$p(\hat{\theta}_{z_i}^*\left|x_{i:n}\right.)\propto G_0\left(\hat{\theta}_{z_i}^*\right)f\left(x_i\left|\hat{\theta}_{z_i}^*\right.\right)$$

  end

end

Algorithm 1: Gibbs sampling using Chinese Restaurant Process (CRP) Representation.

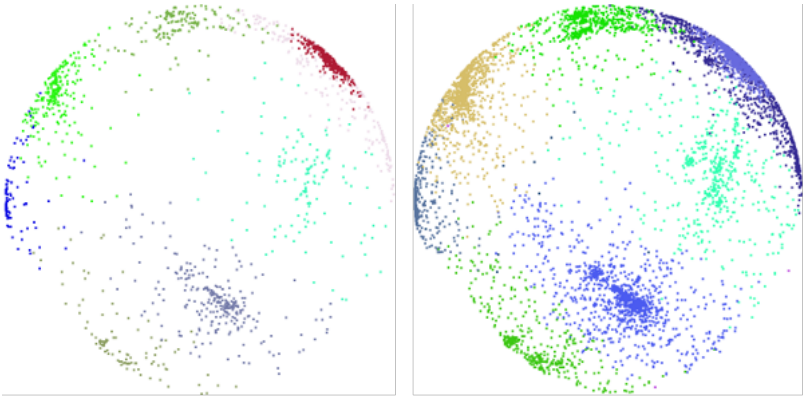

Figure 5.9: The clustered “core set” (left) captures the salient structure of the CPNs, yet contains only 10% of the data items compared to the full CPNs (right). All colors where randomly chosen.

Gaussians, the parameters accounting for the initial assignment of CPNs to clusters $x_{i:n}$, $G_0{=}NIW\,(\mu_0{,}\kappa_0{,}\Psi_0{,}\upsilon_0)$ as well as the number of iterations of the Gibbs sampler. All parameters were set manually, and maintained their default values throughout this thesis apart from the concentration parameter α. The choice of α will affect the clustering performance as will be shown in Chapter 8.

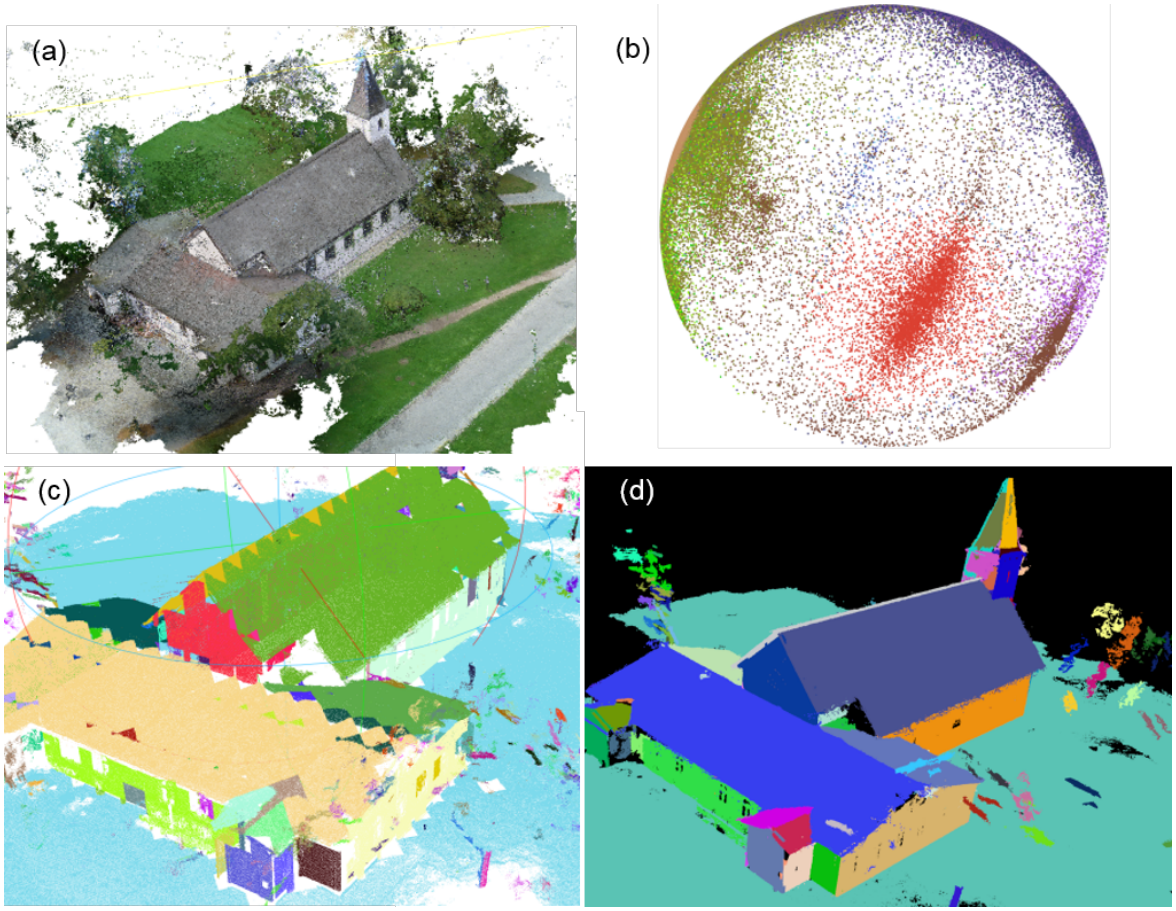

Figure 5.10: (a): The point cloud of a church acquired from images by SfM/MVS is segmented into contiguous patches. The following stages of our algorithm are illustrated: (b) Clustering of clipped plane normals on the unit sphere, (c) Connecting voxels from the same cluster to form regions. Boundaries of regions suffer from "zig-zag" effect (d) Smoothing is employed to remedy this effect caused by voxelization.

5.4 Smoothing

Our segmentation workflow employs fixed size scene voxelization during initialization. This leads to smoothness problems along object boundaries. Hence, connected components from clustering voxels suffer from discontinuities ("zig-zag" effect) at the component boundaries (cf. Figure 5.10). We resolve this problem by stochastic region growing extending the support of the boundary voxels of each region as explained below. The extent to which the regions are grown is constrained by the predefined size of a voxel, ls. It should be noted that the probabilistic clustering provides a good initial segmentation which, combined with the fixed voxel size, enables stochastic region growing to smooth

away the "zig-zag" effects. Without this strong prior segmentation, it would be difficult to develop good criteria that generalize well for merging neighboring voxels consistently. A deterministic cost-function based approach for merging voxels has been presented in (Papon et al. 2013). We compare and contrast the limitations of such an approach with ours in Chapter 8.

The stochastic region growing algorithm employed for smoothing, removing the "zig-zag" effects is developed as an importance sampling-based procedure and is summarized as follows: Let convex-hull-voxels for the jth contiguous region generated by CCA be defined as the set of voxels which lines connecting the elements of the convex hull pass through. The importance of the cth voxel in this set is given by the likelihood function $\mathcal{L}(\pi_c)$ which is the M-Estimator SAmple Consensus (MSAC) score of the clipped plane for the cth voxel. The aim of the smoothing is to get a better estimate of the underlying object boundary. For the jth region,

- `compute the convex-hull-voxels`
- `repeat` N `times`
  - `randomly perturb the normals of the clipped planes`
  - `randomly increase the size of the convex-hull-voxel by drawing from a simple stochastic diffusion process given by` $l_s + rand(0.5 * l_s)$ `to ensure missing boundary points become potential inliers`
  - `compute the sum of` $\mathcal{L}(\pi_c)$ `for these voxels`
- `select inliers of solution with the smallest` $\mathcal{L}(\pi_c)$ `value`

The convex hull of the inliers is a good estimate of the object boundary. Example results from segmenting building facades, the ground as well as roof elements from point clouds derived from images by a SfM/MVS framework, but also point clouds from LiDAR are shown in Figure 5.11.

5.5 Summary

In this chapter, we have presented our algorithm for large-scale segmentation of noisy unstructured 3D point clouds into contiguous patches with associated object boundaries. The algorithm incorporates sample space division and RANSAC plane fitting to reduce the variance in surface normal estimation in 3D point clouds due to noise and point density variations in large-scale scenes avoiding over-fitting. Furthermore, by combining DPM model clustering with connected component analysis the speed of the algorithm is increased by solving parts of the clustering problem deterministically.

Algorithmically, the workflow first divides the input data into a fixed sized voxel grid and then robustly generates surface normals using RANSAC search and least-squares plane fitting within every voxel. Voxels are clustered into regions using a DPM model of Gaussians combined with connected component analysis and refined by a stochastic region growing based on importance sampling. The resulting algorithm works independently of the origin of the input data as well as the point density.

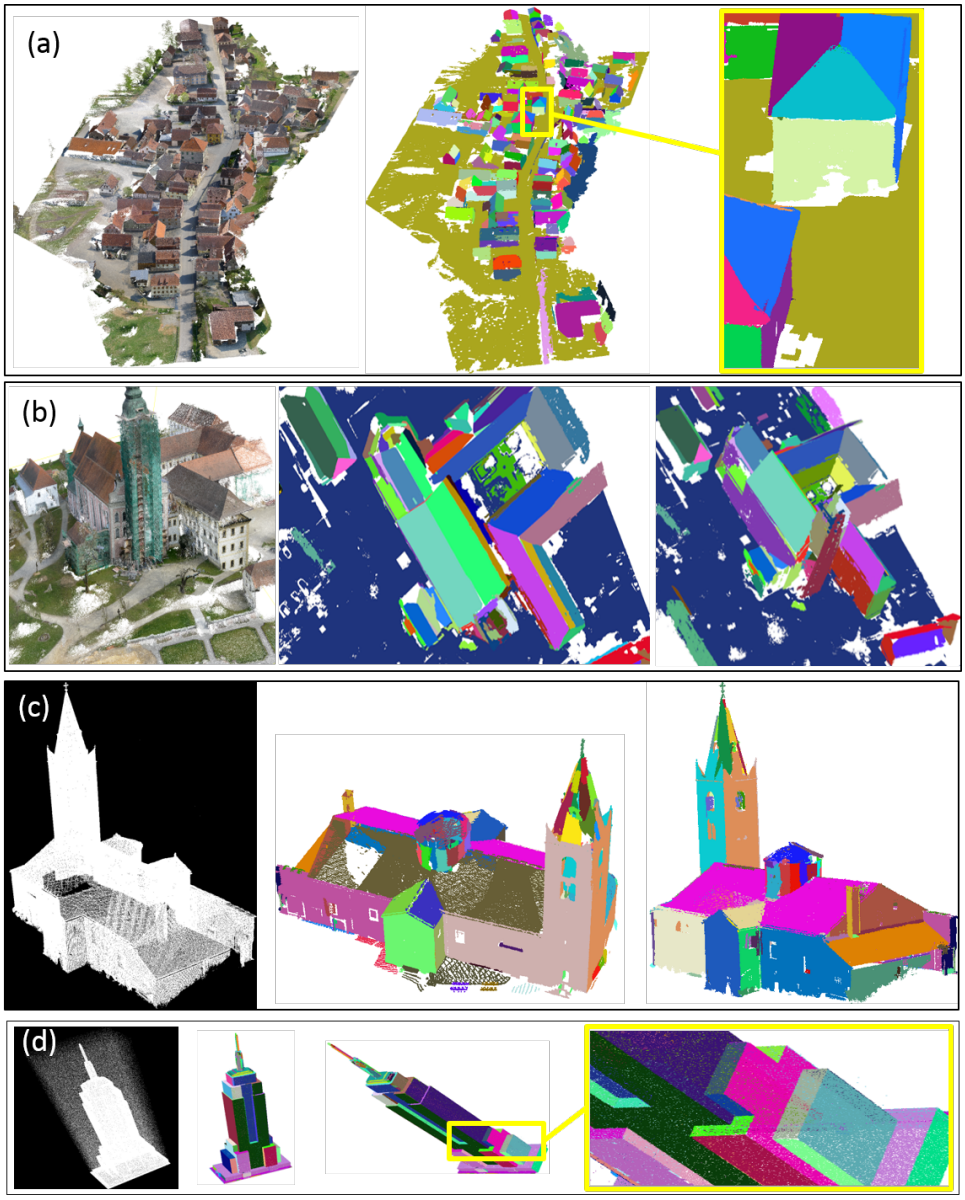

Figure 5.11: Output of segmentation for four data sets. Point clouds "Bonnland" (a) and "Adelsbach", (b) are from SfM/MVS with 107M and 50M 3D points, respectively. The "church" (c) and "empire" (d) data sets are courtesy of (Lafarge and Alliez n.d.). Segmented patches are randomly colored (Nguatem and Mayer 2016).

Chapter 6

Semantic Scene Interpretation

The previous chapter has presented a large-scale patch segmentation algorithm for noisy 3D point clouds which detects contiguous patches corresponding to scene objects such as roof elements, facades and the ground. Yet, the patches have no direct association to semantic labels of the scene objects. Therefore, it is not possible to say that, e.g., segments one, two, and three represent roof elements or that segments four and five belong to the ground. Classical supervised learning using point-wise classification algorithms have been widely used to resolve similar problems of 3D scene labeling (Lim and Suter 2009, Brodu and Lague 2012, Weinmann 2016). Recently, approaches employing deep learning have also targeted this problems (Charles et al. 2017, Boulch et al. 2017, Landrieu and Simonovsky 2018). Nevertheless, interpretation remains a challenge for point clouds of urban scenes derived from images by SfM/MVS and LiDAR due to the presence of substantial noise, point density variations as well as possibly non-planar terrain (see Chapter 3). In this chapter, we tackle this problem at the second stage of our modeling pipeline (see Figure 6.2) using a combination of heuristics and basic architectural rules. The text in this chapter loosely follows (Nguatem and Mayer 2017). Figure 6.1 shows an example output of scene interpretation and labeling (after segmentation, cf. Chapter. 5) using the algorithm described in the sequel.

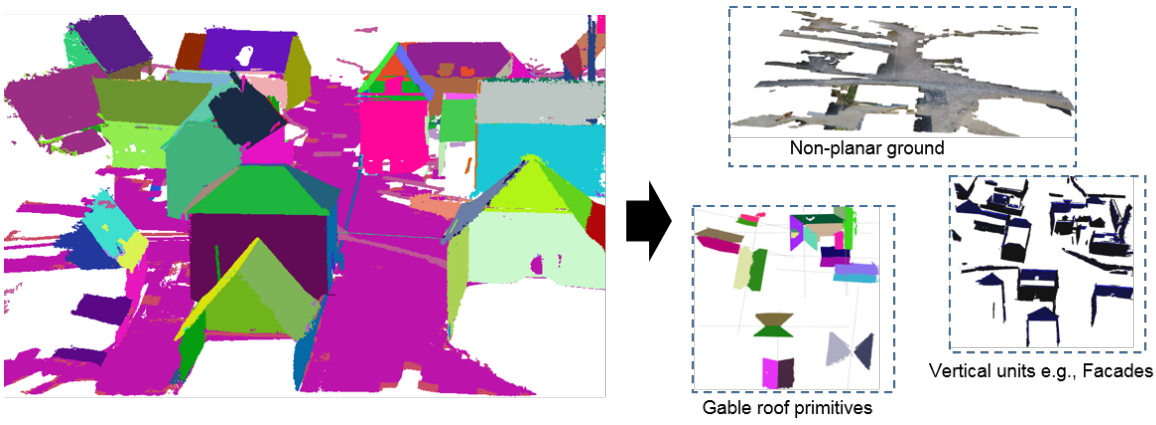

Figure 6.1: Scene Interpretation: To the output of the segmentation of a dataset derived from images by SfM/MVS containing many patches (left) we apply simple heuristics and decompose the scene semantically into its constituent units (right), the ground, roof elements and facades.

There are two major objectives of this chapter:

1. Classify the output of the large-scale patch segmentation algorithm into one of the four classes, the ground, facade, roof element and the rest using basic, yet robust architectural rules and heuristics.

2. Detect combinations of facade and roof segments that form unique shape primitives for buildings.

While simple geometric relations underpin the basic architectural rules and heuristics employed, we assign four distinct labels (ground, facade, roof element, and rest) to the segments derived in the previous chapter. This achieves a consistent semantic interpretation of the scene and hence, labeling of the individual 3D points. Amongst the four class labels, we presume that segments associated with the class label "rest" predominantly correspond to vegetation. They will not be considered for further processing within this thesis.

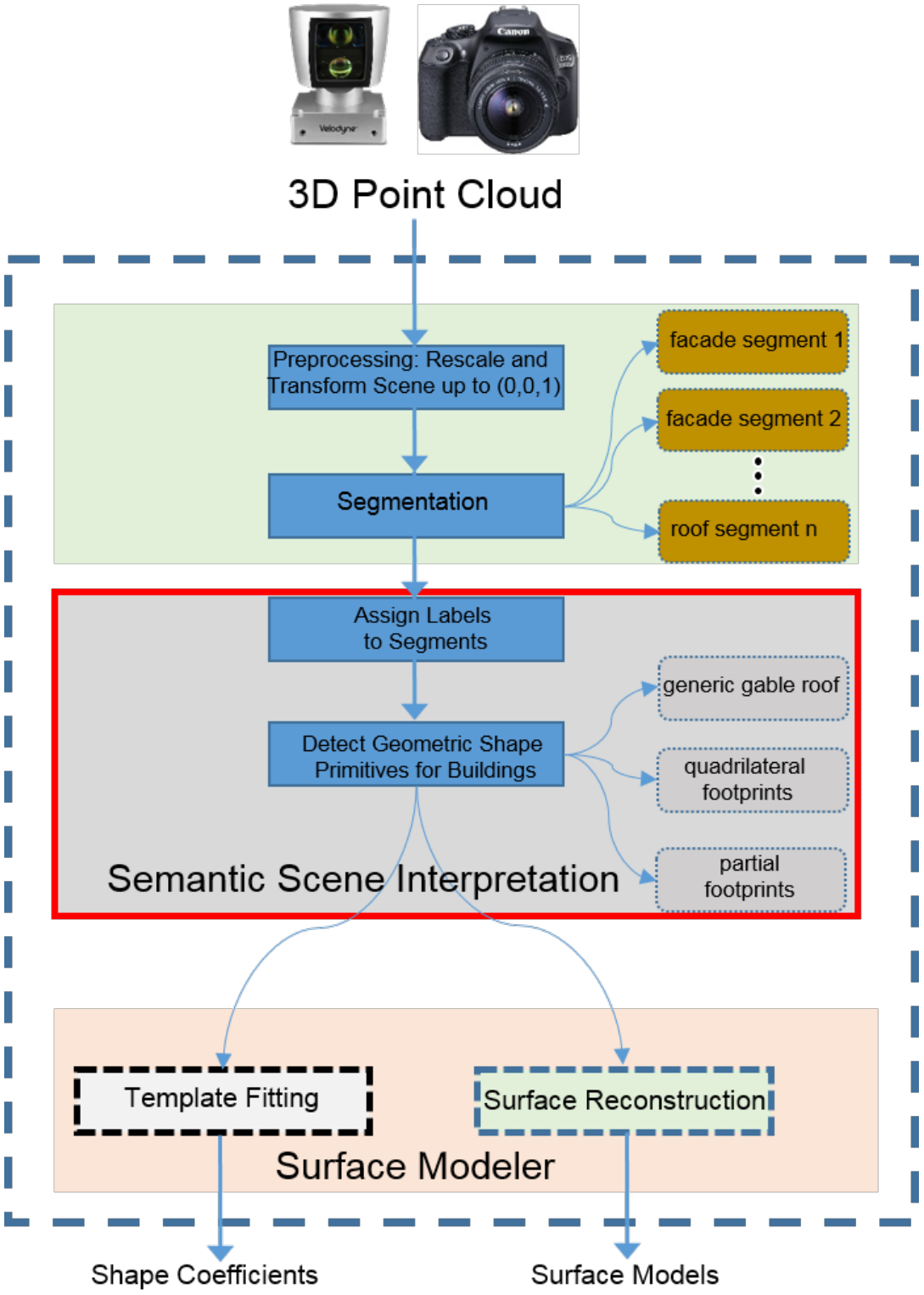

Figure 6.2: Architecture: Framework – Semantic Scene Interpretation

Compared to state-of-the-art 3D semantic labeling algorithms employing classical or neural networks-based point-wise classification, patch labeling offers a number of advantages:

- Fast and scalable since a single class label is assigned to a contiguous patch—a segment containing many hundreds or thousands of 3D points.
- Based on simple geometric relationships and well understood basic architectural rules that generalize well for urban scenes.
- Robust against noise due to the use of robust segmentation.
- Enables multi-label assignment, e.g., points along the line of intersection of facades and ground segments have both labels as can be seen in the example depicted in Figure 6.3.
- Simplifies the creation of relations between labeled segments which are the basis for constructing geometric shape primitives for buildings as described in the sequel.

Figure 6.3: Results of segmentation overlapping at segment boundaries. 3D points at regions overlapping are assigned multiple labels. In this example, inliers on the lines of intersection of the two facades, as well as of the facades and the ground have multiple labels.

Two major geometric properties which are the basis for architectural rules for appropriate scene interpretation are proximity and spatial orientation. Proximity quantifies how close segments are to one another in $\mathbb{R}^3$. Neighboring labeled segments that are close to one another are assumed to be connected building parts, e.g., adjacent facades. The measure of proximity adopted throughout this thesis is the shortest Euclidean distance from a segment's bounding contour to that of its neighbors.

A segment's spatial orientation in its simplest form is represented by the direction of the segment's surface normal, estimated using RANSAC plane search and least squares fit. Yet, spatial orientation alone is not distinctive enough to, e.g., differentiate vertical up-right segments from horizontal ones. However, in combination with the known vertical (up) direction, a segment's "relative" spatial orientation can be fully quantified, hence enabling the estimation of a segment's semantic label, e.g., for facades as we present in the next section. At this point, we suppose that the input point cloud data $\mathcal{D}$ has been consistently transformed such that the vertical (up) direction is parallel to the z-axis (see Chapter 2). In the rest of the chapter we present our basic architectural rules for scene interpretation. We first identify all segments associated with the label facade in the scene, then the ground and finally combinations of segments belonging to unique building parts, hence creating geometric shape primitives of building parts.

6.1 Facade Detection

When there are no buildings with curved facades – as this is most often the case in urban scenes, facades typically consist of vertical upright structures. Geometrically, they can be well approximated with quadrilaterals in $\mathbb{R}^3$. Then, facade detection reduces to the detection of vertical segments. We use RANSAC together with least-squares plane fitting (Schnabel et al. 2007) on each segment from the output of Chapter 5. We impose orthogonality to the vertical (up) direction, $\boldsymbol{v}$, as a constraint to detect facades as follows: For a given segment $\mathbf{S}_1$, RANSAC is used to search and fit a plane with the normal vector given by $\boldsymbol{n}_1$. Then, $\mathbf{S}_1$ is a facade if

$$|\boldsymbol{n}_1 \cdot \boldsymbol{v}| \leq \epsilon, \tag{6.1}$$

where ϵ is a small tolerance angle accounting for architectural imperfection as well as the uncertainty of $\boldsymbol{v}$. The choice of ϵ can affect the detection result. However, for our experiments, we assume a fixed value for ϵ. We derive this value empirically as presented in Chapter 8. Examples of facades detected by applying this simple scheme on a point cloud derived from SfM/MVS are shown in Figure 6.4. While this reveals the robustness of this simple rule, it also unveils a peculiarity of the segmentation algorithm which over-segments curved surfaces if present in the scene. This is due to the bias towards planar structures segmented by fitting planes within voxels as described in Chapter 5. This effect is noted by over-segmentation of the cylindrical tower in the center of the castle depicted in Figure 6.4. Even in this unique case with atypical buildings with curved facades, our algorithm still detects facades correctly even though they are split incorrectly. We elaborate on this limitation in Chapter 8.

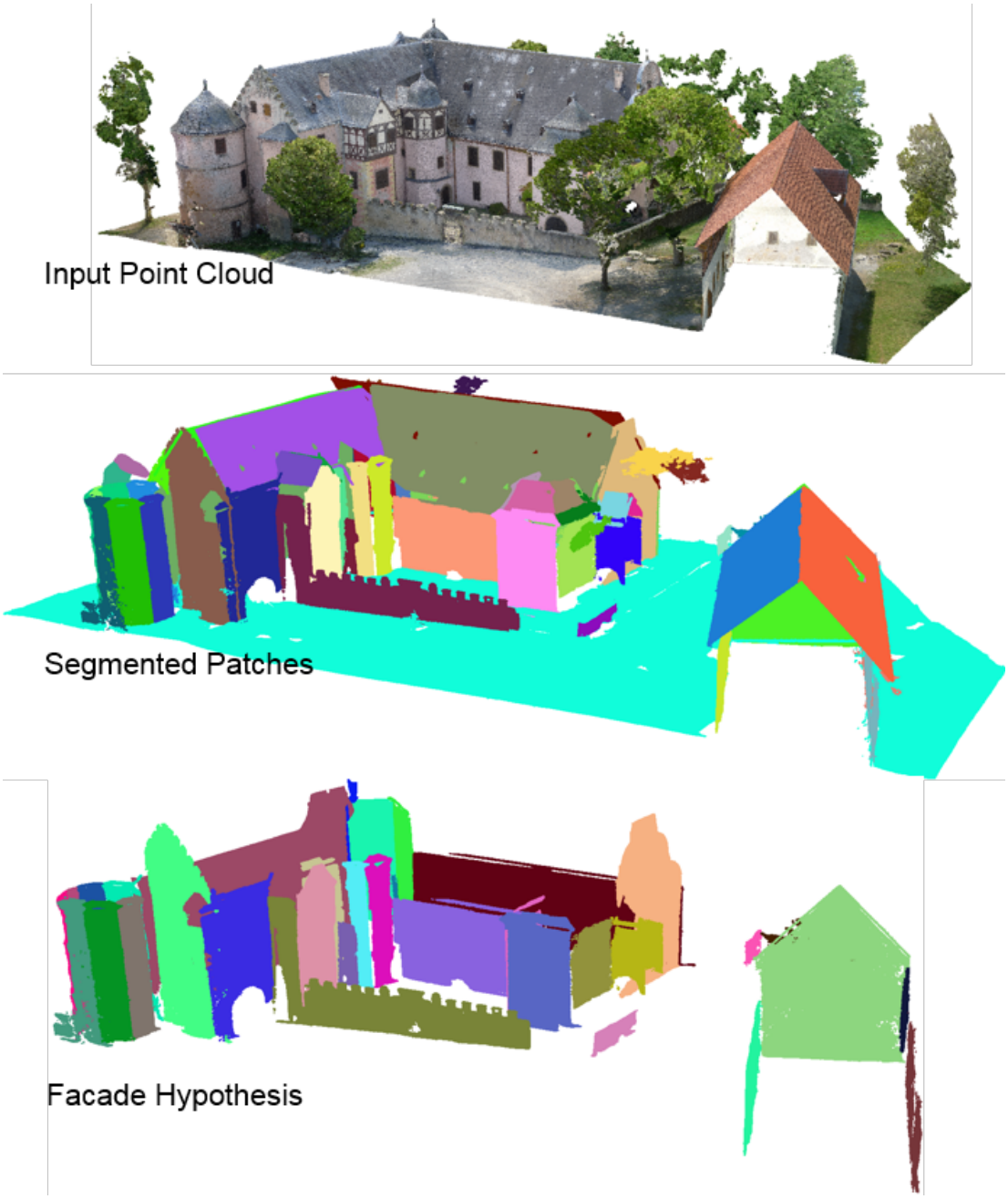

Figure 6.4: Simple architectural rules are used to assign labels to the output of the segmentation. In this example, the point cloud contains 44 million 3D points derived from images by a SfM/MVS (top). The segmentation result (middle) shows distinct segments for meaningful units such as facades, roofs and the ground. The segment with the largest surface area is the ground (here turquoise). Normals are used to label segments belonging to facades (bottom). All colors of individual segments before and after semantic labeling are randomly chosen.

6.2 Ground Detection

In contrast to facades, the geometry of the ground can be (highly) non-planar, cf. e.g., Figure 6.1. Developing features for non-planar ground detection that generalize well by means of traditional supervised classification algorithms (cf. Section 3.5) is a non-trivial task. Intensity values of individual 3D points are often included to remedy this (Weinmann 2016). However, intensity is only discriminative for natural ground with vegetation. The situation is even worsen in the presence of noise and 3D reconstruction artifacts. Yet, if we assume that buildings are enclosed by a surrounding ground, i.e., an envelope of 3D points, and that there are no very tall buildings in our urban scenes, the ground is the segment with the largest surface area. In the 3D point cloud shown in Figure 6.1, ground points are colored purple. We approximate the surface area by the convex hull. This approach for ground detection is simple, yet generalizes well even for large scenes with many irregularly and tightly placed buildings as well as in the presence of substantial vegetation and non-planar terrain (cf. Figure 6.7).

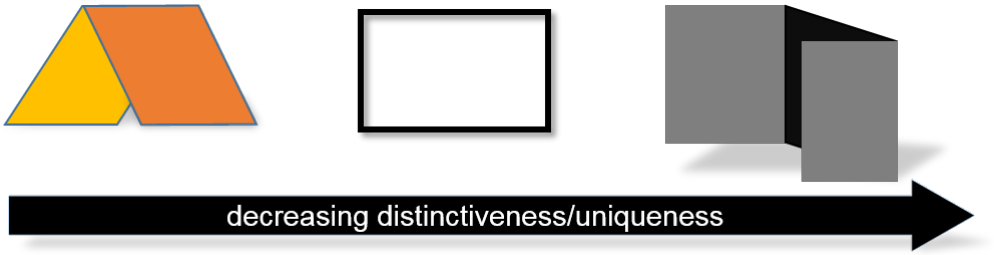

Figure 6.5: Geometric Shape Primitives (GSP) representing unique building parts

6.3 Detection of Building Primitives

Buildings in urban scenes are often composed of combinations of quadrilaterals. One labeled facade segment combined with the ground is not enough to define a building with a specific roof type (flat, gable, mansard, pyramid, etc.) in 3D point cloud data. Therefore, we detect distinct combinations of facades and roof segments that often signify the presence of buildings in urban scenes. These combinations define core building parts and we call them "**G**eometric **S**hape **P**rimitives" (GSPs) for buildings. A subset of GSPs considered in this thesis is shown in Figure 6.5. They enable, e.g., the reconstruction of buildings with a gable roof by detecting the presence of connected opposing roof segments (left) as well as connected facades forming a closed cycle (center), and finally connecting three facades forming a convex part of the footprint (right). GSPs are not mutually exclusive in describing a building with a specific roof type, yet some are more discriminative for a particular building and roof type than others. For example, two connected segments with almost the same inclination give strong evidence for a gable roof which is not the case for just four connecting walls. The latter only defines a quadrilateral footprint, and could be underneath a mansard or a pyramid roof as well. Therefore, it

is important to prioritize the GSPs for further processing within the overall pipeline for creating building models from point clouds of urban scenes. The more discriminative a GSP is, the higher is its priority. GSPs with higher priority are detected first and the corresponding building is modeled (cf. Chapter 7) before the GSPs further down the priority list are considered. This simple intuitive strategy is necessary to avoid ambiguous detections and, hence, modeling one and the same building multiple times. We now present the detection schemes for the different GSPs considered in this thesis.

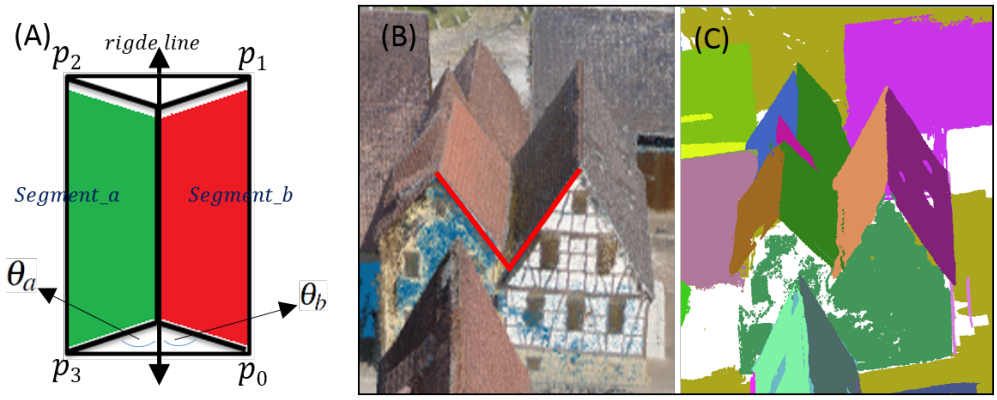

Figure 6.6: Roof hypothesis defined by intersecting segment pair, a and b (A). The point cloud from images by SfM/MVS (B) is segmented (C). An appealing Z-symmetry is present, because both buildings are close to one another, but in the wrong direction.

6.3.1 Detection of Generic Gable Roof

The majority of buildings we encounter in urban scenes have gable roofs. Combinations of this basic roof type often lead to more complex roof geometries. The salient feature of the GSP for the generic gable roof is the presence of two opposing roof planes with similar slope. We use the template shown in Figure 6.6(A) to detect this GSP which fully captures the geometry of a generic gable roof. Template-based detection has the major advantage that even in the presence of substantial noise, e.g., plenty of vegetation, robust detection is guaranteed. Furthermore, model completeness is ascertained in the presence of missing data, as is often the case in point clouds derived from images by SfM/MVS of close built buildings.

(Zhou and Neumann 2012) introduced the idea of using general rotational symmetry for generic gable roof detection. This has been further concretized through the notion of z-symmetry in (Verdie et al. 2015). Both approaches have the constraints that connected gable roof segments share a similar slope, i.e., segments forming a gable roof pair are strictly symmetrical about the vertical (up) direction as well as the ridge line. However, these constraints are very stringent and allow little or no architectural imperfections and asymmetry. They also do not permit substantial noise or point cloud

data acquisition artifacts as in point clouds derived from images by SfM/MVS. Furthermore, it is very common in urban scenes that neighboring buildings are too close to one another and thus, an appealing z-symmetry is present, but in the opposite direction as shown in the multi-gable roof example of Figure 6.6 (B and C).

The approach followed in this thesis robustly detects segment pairs ($\mathbf{S}_1$, $\mathbf{S}_2$), which form a good hypothesis for a GSP for single generic gable roof. Particularly, segments $\mathbf{S}_1$ and $\mathbf{S}_2$ form a generic gable roof GSP if the following propositions hold:

1. **Proximity**: Both $\mathbf{S}_1$ and $\mathbf{S}_2$ should be close to each other and to the ridge line, i.e., the line of intersection of the two planes formed by RANSAC plane search and least squares fit for $\mathbf{S}_1$ and $\mathbf{S}_2$.

2. **Concavity**: The ridge line is at angles to the segments $\mathbf{S}_1$ and $\mathbf{S}_2$ (see Figure 6.6).

3. **Downwards Concavity**: The centroid of both segments should lie below the ridge line.

4. **Size-Similarity**: There shouldn't be a substantial difference in the convex-hull area of $\mathbf{S}_1$ and $\mathbf{S}_2$.

The first three rules are in accordance with the spatial configuration and orientation of the roof. The size-similarity constraint is important to avoid false detections which may arise when a segment from a "true" roof is combined with a segment from a roof superstructure, e.g., an oriel. A major disadvantage of this relaxation as compared to the simple, yet more restrictive z-symmetry assumption (Zhou and Neumann 2012, Verdie et al. 2015) is that both symmetric and asymmetric (half)-hipped as well as pyramid roof buildings are equally detected using the GSP for generic gable roof (see detected roof segment pairs in the example depicted in Figure 6.1). Also, in the presence of substantial noise, e.g., plenty of vegetation, certain configurations of leaves may lead to false detections. Nevertheless, this can be readily resolved during fitting and regularization which will be described in the next chapter.

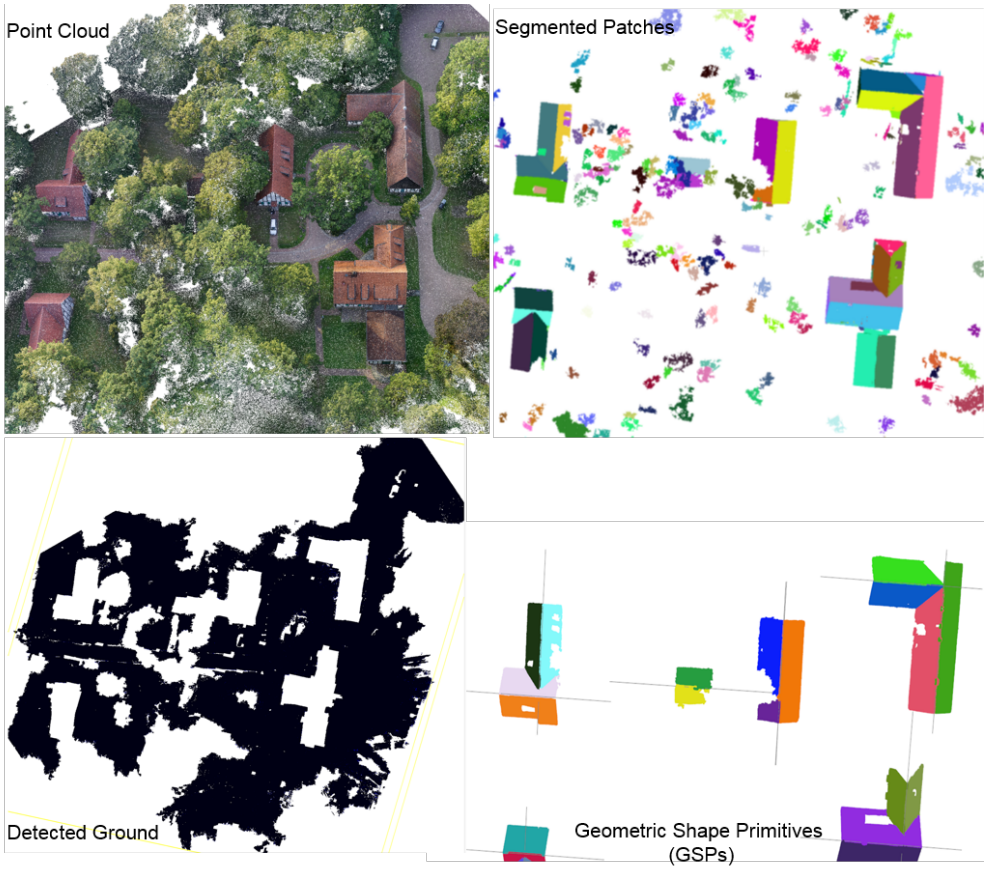

Figure 6.7: Besides accurate semantic scene decomposition into constituent units with possibly different labels, our algorithm is robust against noise, e.g., vegetation or reconstruction artifacts from images by SfM/MVS, and assigns correct labels to most patches. Here, 76.7 million points acquired from images by SfM/MVS reconstruction of a residential area are accurately segmented, patches are labeled and GSPs detected for gable roof (patches are randomly colored).

6.3.2 Detection of Footprints

The GSP for a generic gable roof buildings described above assumes the presence of enough 3D points for detecting and connecting gable roof segments. This is often the case for aerial data acquisition, e.g., with an unmanned aerial vehicle—UAV (Mayer et al. 2012). However, if the point cloud data is acquired using terrestrial laser scanning or from images captured by hand-held consumer cameras and SfM/MVS, 3D points associated with roofs, if at all present, are often very sparse and not adequate for detecting opposing roof segment pairs. On the other hand, the 3D points associated with the facades are mostly dense enough for robust facade detection. We, thus, devise the GSP describing the presence of buildings based on facades. Particularly, connected facades describe the

underlying footprint signifying the presence of buildings. Footprints are closed cycles. If they are complex, can often be decomposed into constituent quadrilaterals. However, we assume simple footprints made of quadrilaterals and detect them as follows:

1. Detect all connecting facade segments and build an undirected facade connectivity graph, with segments as nodes and edges formed between connected segments. Proximity is employed as additional constraint to ensure that adjacent facades segments connect to one another.

2. Detect all unique cycles present in undirected graph using the depth-first search as described in (SEDGEWICK 2002).

Figure 6.8 illustrates this example on a building with complex footprint.

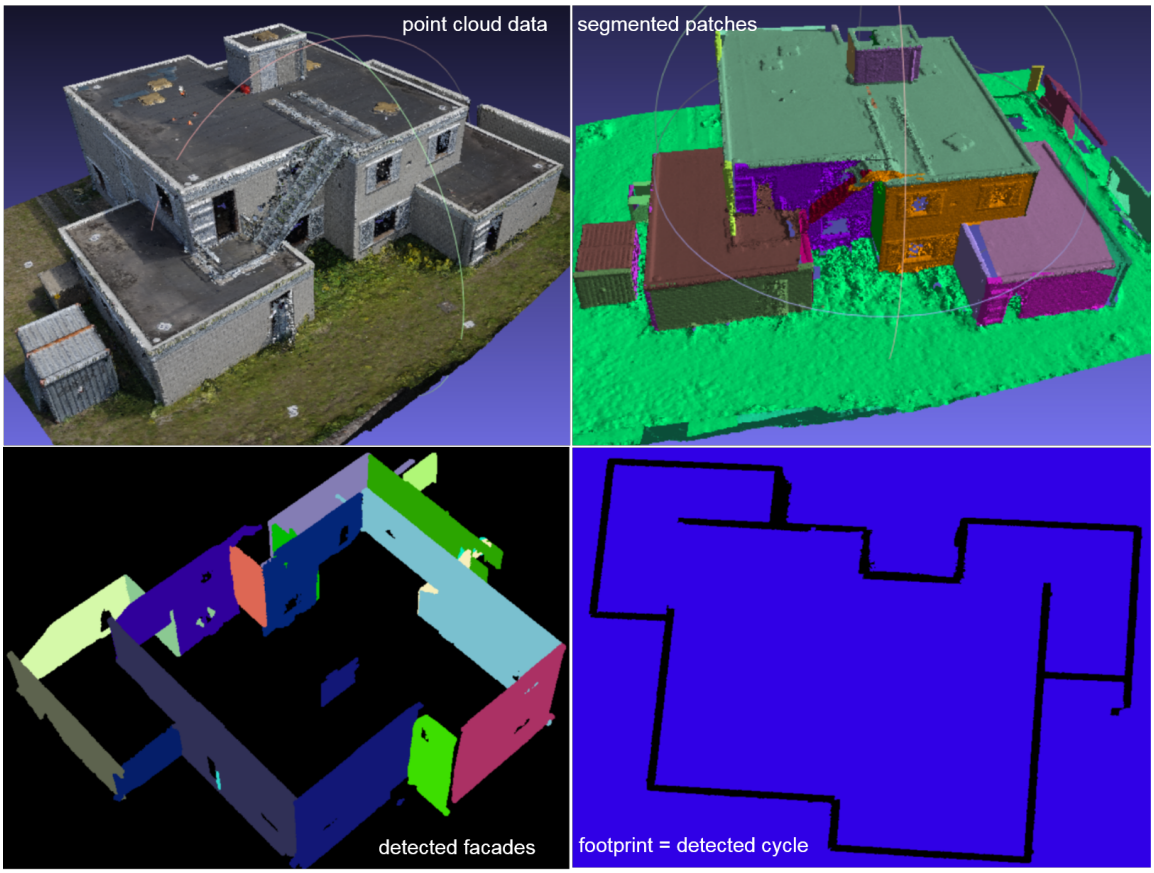

Figure 6.8: Point cloud data of a building with complex footprint. The detected footprint forms a closed cycle traversing along the walls.

6.3.3 Detection of Partial Footprints

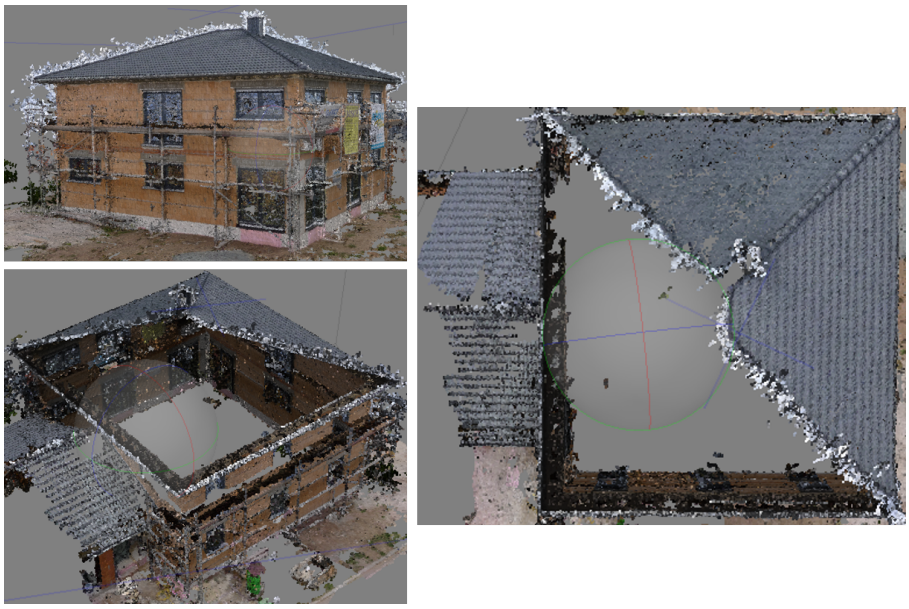

Figure 6.9: The completeness of the point cloud representing the scene can often not be guaranteed due to problems with data acquired from the ground.

3D point cloud data of urban scenes are often incomplete. In many cases, this is the result of occlusion, e.g., by vegetation or neighboring buildings. Here, the presence of a pair of connected roof segments highlighting a generic gable roof often cannot be ascertained. Furthermore, it is usually difficult to collect data associated with all facades to produce a complete cycle for the footprint as described in the previous section. To deal with this kind of situation, we assume a single missing facade and detect three connected facades forming the convex-hull to ensure model completeness. Figure 6.5 (right) shows the template for this GSP. It is the most flexible primitive in our library for buildings and consists only of three connected facades forming the footprint using the convex-hull. An example point cloud from an urban scene with incomplete data is shown in Figure 6.9. An attempt to detect a complete cycle forming the footprint will fail due to missing data as can demonstrated in Figure 6.10. We could manually extend the distance threshold deciding about if two facades connect, in order to resolve this problem and produce a complete cycle indicated. Yet, this semi-automatic approach clearly does not fit to our initial goals—automatic reconstruction of building models from point clouds. Thus, we impose proximity as a constraint between connecting facade segments and replicate the missing facade from the existing on the opposite side (cf. Figure 6.10).

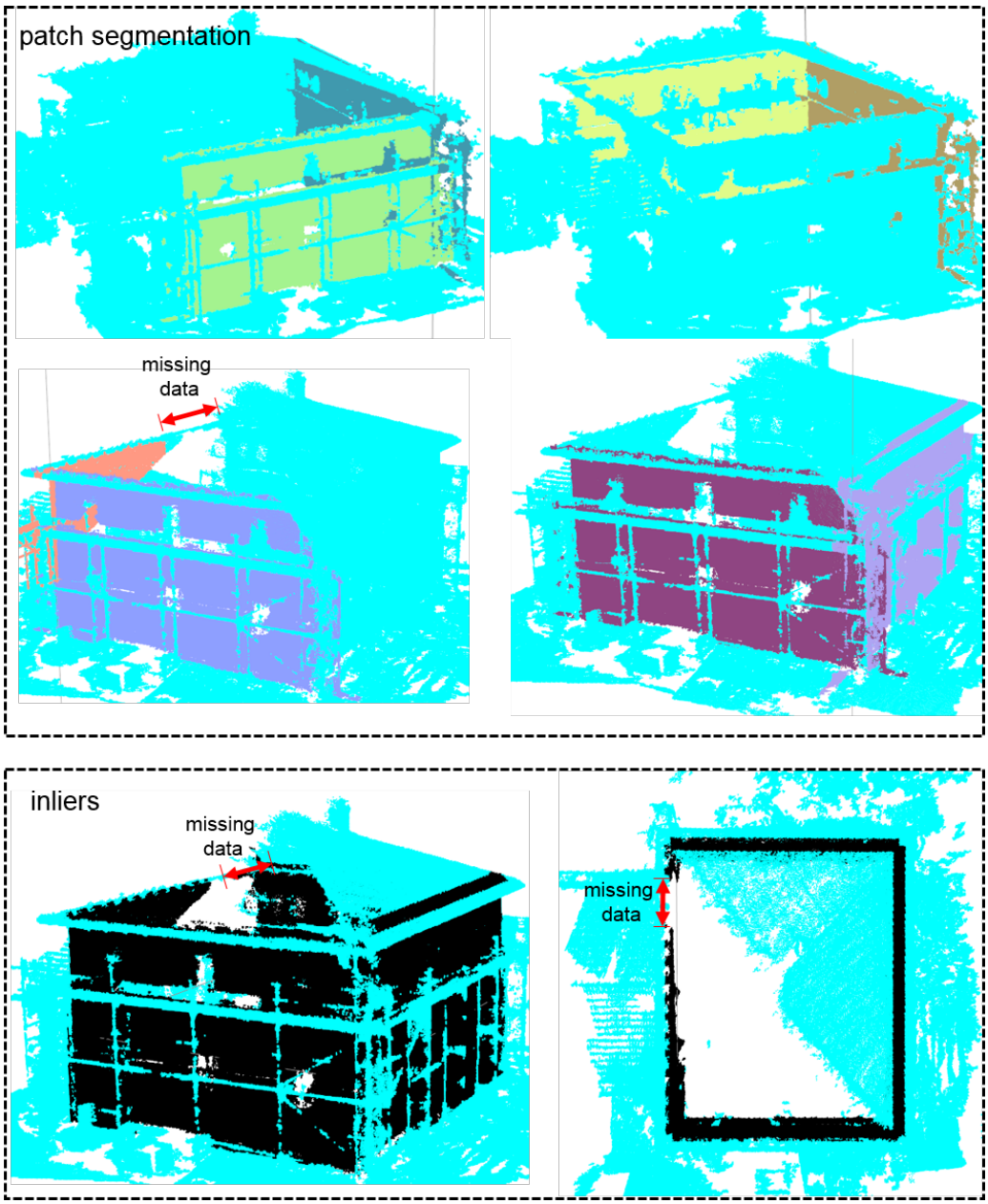

Figure 6.10: Robustness in the presence of missing data is difficult to achieve. Template-based modeling is our approach to resolve this difficulty. In this example, inliers of the segmented facades (randomly colored points) are missing on the fourth facade as a result of incomplete data. A cycle of connected facades, thus, cannot be determined. In this unique case, we replicate the missing facade with the facade on the opposite side, thereby forming a closed quadrilateral footprint as required by our algorithm.

6.3.4 Detection of Flat and Mansard Roof Primitives

The geometry of flat or the top of mansard roofs is simple and can be approximated with a quadrilateral in $\mathbb{R}^3$ (in analogy to the geometry of facades). Parallelism of the segment's normal to the vertical (up) direction, $\boldsymbol{v}$, is imposed as a constraint to detect these geometric shape primitives as follows: For a given segment, S_1, RANSAC is used to search and fit a plane with normal $\boldsymbol{n}_1$. Then, S_1 is a primitive of a flat or mansard roof building if $|1 - \boldsymbol{n}_1 \cdot \boldsymbol{v}| \leq \epsilon$.

6.4 Summary

This chapter presented a set of basic architectural rules and heuristics that enables the consistent association of semantic labels to segments, hence, creating a semantic labeling of the complete scene. Four labels are used, namely the ground, roof element, facade and "rest". Furthermore, we presented a set of geometric shape primitives (GSP) for buildings alongside their detection schemes which generalize even in the presence of substantial noise, vegetation and incomplete data. This can be seen, e.g., in the example for detected generic gable roofs given in Figure 6.7. In the next chapter, we use the detected GSPs as informative priors to guide a randomized search for building models by fitting predefined polygonal chain configurations to the 3D point cloud data.

Chapter 7

From Geometric Shape Primitives to Building Models

The Geometric Shape Primitives (GSPs) as described in Chapter 6 only highlight the presence of buildings. GSPs alone are not enough to infer coefficients of watertight polygonal models of buildings, let alone for different roof types, e.g., hipped, gable, flat or mansard. In this chapter, we present a stochastic framework to search and fit competing sets of polygons to point clouds that produces watertight (Ericson 2004) computer-aided design (CAD) models for buildings and the text is weakly based on (Nguatem et al. 2013a, Nguatem and Mayer 2017). The proposed framework is the final stage of the overall modeling pipeline presented in this thesis as shown in Figure 7.2. Its output are geometric object models that consist of shape coefficients for buildings as well as triangulated polygonal surface meshes for the ground. A result of the proposed framework is shown in Figure 7.1 below. In this example, watertight polygons (right) are fit to the point cloud associated to seven buildings of an urban scene. The segmented point cloud data (left) shows an irregular building distribution as well as non-planar ground. The fitted polygons of the model are colored according to the inferred semantic label.

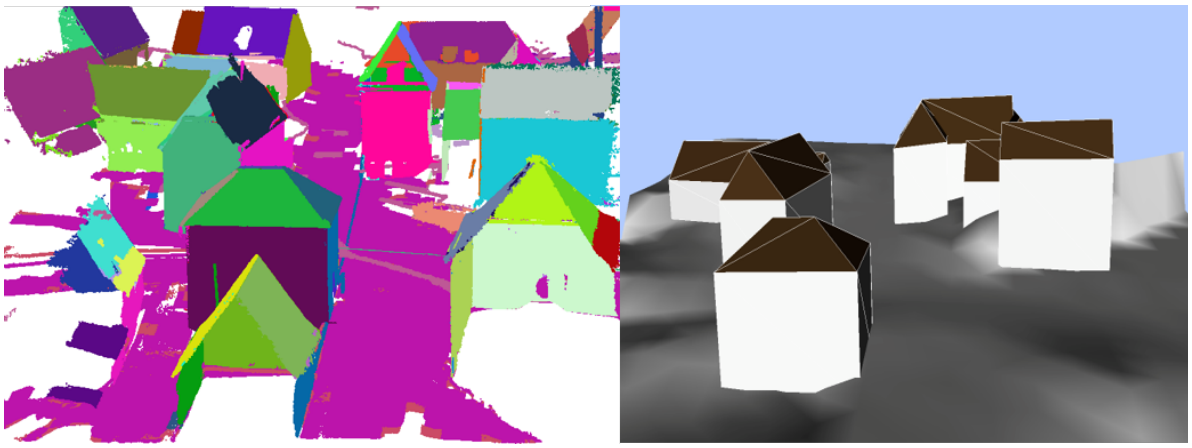

Figure 7.1: Polygonal model (right) from labeled segments (left) for point cloud data derived from images by SfM/MVS for an urban scene. Fitted polygons of the model are color-coded according to the semantic label, e.g., dark brown for roofs.

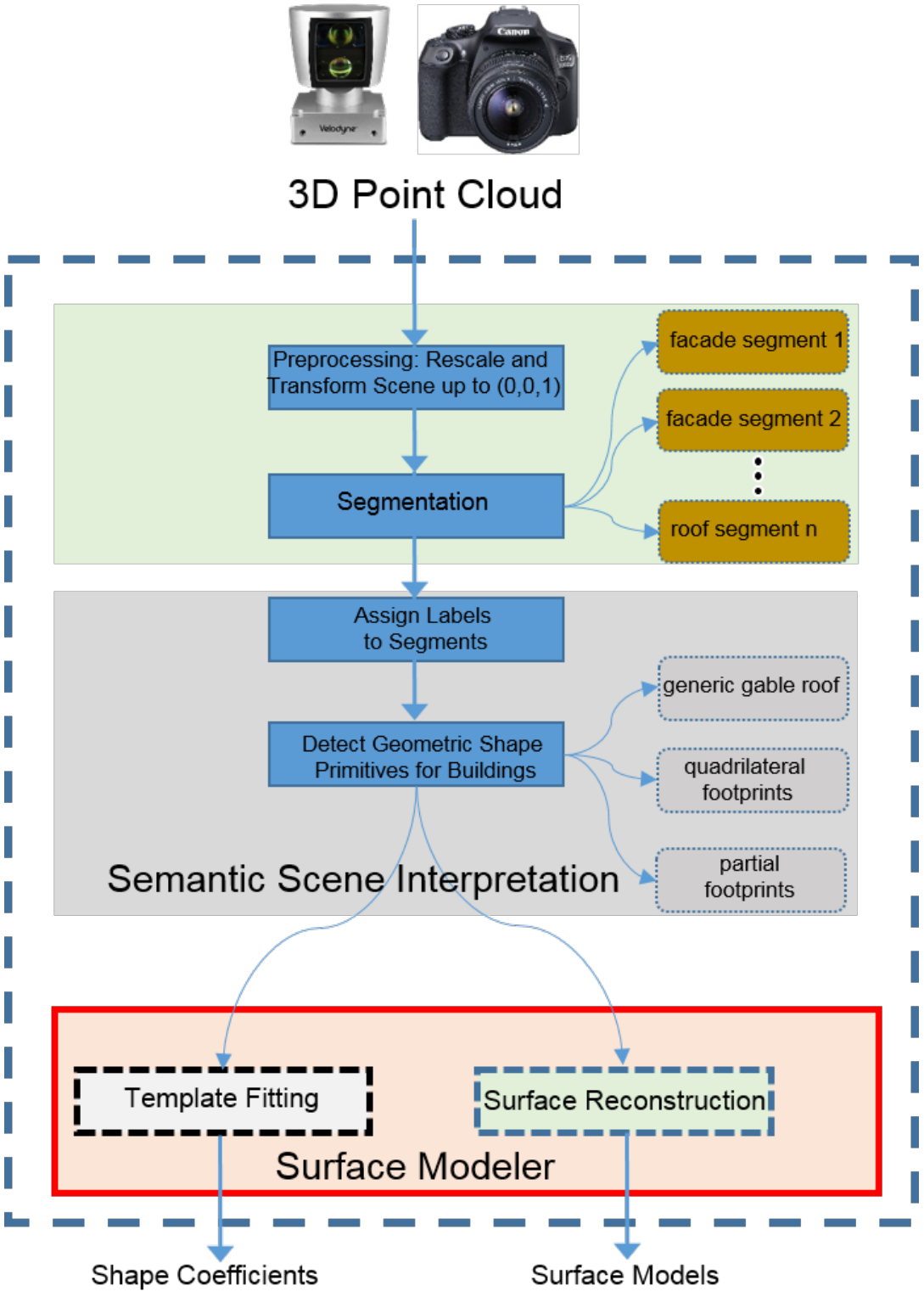

Figure 7.2: Architecture: Framework – Surface Modeler

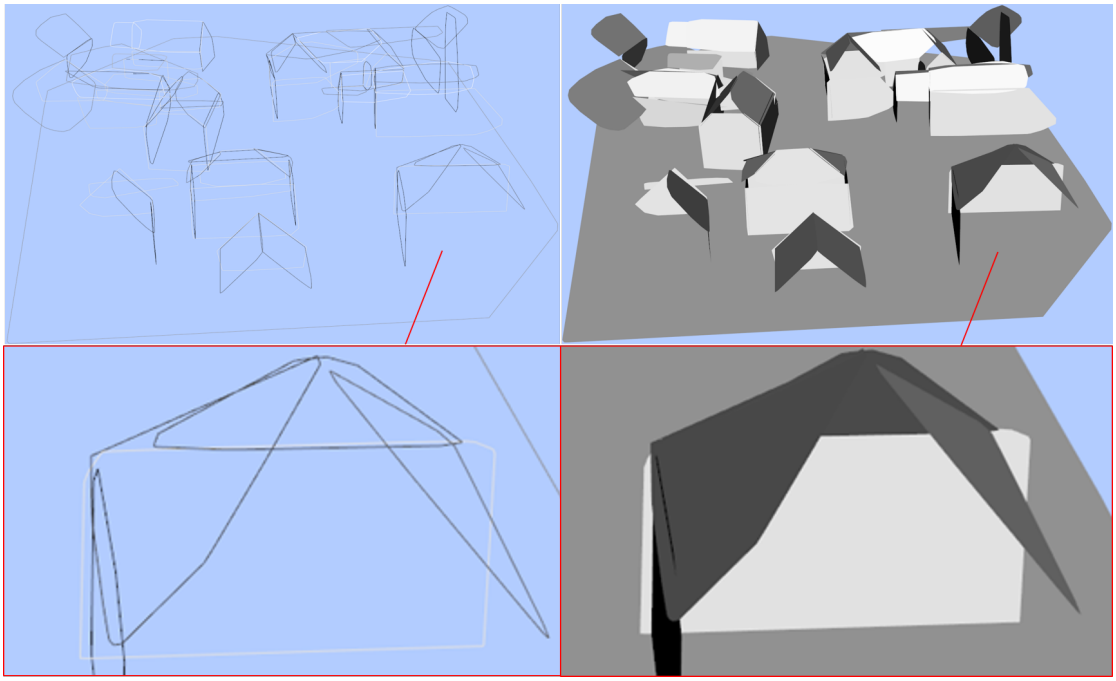

Figure 7.3: Sub-optimal fitting of polygons to convex hulls of segments produces non-watertight models.

Basically, one could fit polygons to the convex-hulls of all labeled segments derived by the large-scale segmentation algorithm presented in Chapter 5. Ideally, such a naive approach could produce the desired watertight polygonal models. However, the presence of substantial noise, non-planar ground as well as vegetation—as is often the case in point clouds of urban scenes derived from images by SfM/MVS—usually results in sub-optimal and non-watertight models (cf. Figure 7.3 for the data set of the example in Figure 7.1 left). These problems may exacerbate in the presence of missing data. To overcome them, we adopt a template-based approach to polygonal model fitting. Template-based fitting of GSPs to the data has two major advantages:

1. It guarantees model completeness in the presence of incomplete data, i.e., it is robust to incomplete data, and,

2. It ensures decent results in the presence of substantial noise, i.e., it is robust with respect to noise.

A major disadvantage of this approach is its inability to capture object shapes beyond the set of predefined shape templates. This, however, can easily be remedied by extending the set of templates when needed. In Chapter 9, we present a detailed discussion on the strengths and weaknesses of this approach. We propose an approach based on stochastic search to fit templates of polygonal models to 3D point cloud data. This is summarized in the algorithmic workflow shown in Figure 7.4. The algorithm randomly

generates competing configurations of a polygonal chain $\boldsymbol{c}_u$ in accordance with a predefined template, assesses them consistently with a likelihood function, and finally selects a single "best" polygonal chain configuration as the final watertight model according to a given criterion. GSPs detected in the previous chapter are used as informative priors to guide the stochastic search. The form of the likelihood function follows directly from the polygonal chain constituting a configuration. We adopt an MSAC—M-Estimator SAmple Consensus (Torr and Zisserman 2000) based likelihood to determine if a 3D point from the point cloud is within the close vicinity, i.e., is inlier of the polygonal chain defined by a configuration. Mathematically, this is defined as follows,

$$\mathcal{L}\left(\boldsymbol{c}_u\right) = \exp\left(-\sum_j \rho\left(e_j\right)\right) \tag{7.1}$$

with

$$\rho\left(e_j\right) = \begin{cases} e_j & e_j < T \\ T & e_j \geq T \end{cases} \tag{7.2}$$

T a given threshold and e_j the shortest Euclidean distance from point p_j in the point cloud to the surface of the polygonal chain defined by configuration $\boldsymbol{c}_u$. This form of assessing polygons using an MSAC-based likelihood is similar to RANSAC-based plane fitting (Schnabel et al. 2007). The peculiarity here is to ensure that inliers are just related to the polygons under consideration. Thus, inliers are only those 3D points that additionally pass a "polygon-hit-test" (Ericson 2004) for the polygonal chain of configuration $\boldsymbol{c}_u$ and all the remaining points are outliers.

Randomly generating and scoring competing geometric configurations that represent shapes of objects have been widely used in the literature within a Markov chain setting (Saatkamp and Schmittwilken 2007, Huang et al. 2011). These approaches, however, suffer from being computationally expensive and are not scalable since the search space increases exponentially with the number of parameters in a single configuration. To ensure an effective exploration of the search space during random sample generation, i.e., evolution of the Markov chain, it is important to use informative priors. We employ GSPs for buildings detected in Chapter 6 as priors to guide our search. Random configurations of polygonal chains are only generated within reasonable bounds to these GSPs. The resulting guided search explores the search space more effectively compared to a naive random sampling within the search space.

Having presented the likelihood function for assessing polygonal chains constituting configurations that represent building models, we now demonstrate how the various configurations $\boldsymbol{c}_u$ are generated using the detected GSPs as priors. We begin with the shape primitives for the generic gable roof, then move to flat and finally mansard roofs. Finally, we present our approach for modeling the ground.

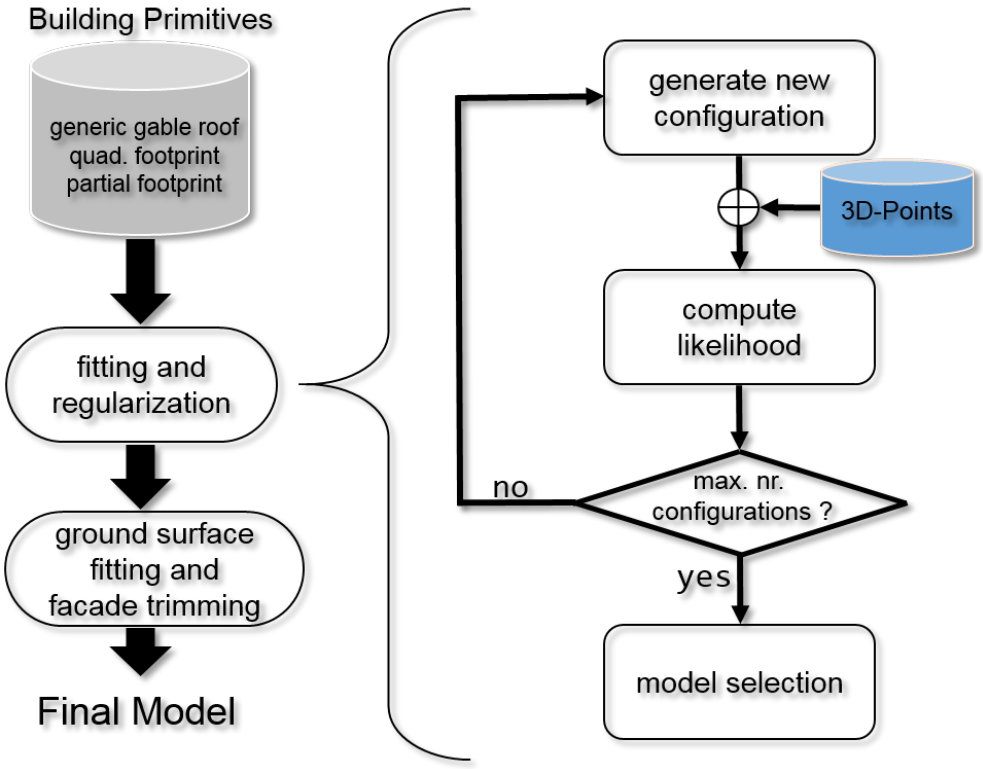

Figure 7.4: Workflow for the generation, assessment and selection of competing configurations (Nguatem and Mayer 2017)

7.1 Polygonal Chain Configurations for Generic Gable Roof

GSPs describing generic gable roofs are highly discriminative as compared to flat or mansard roofs when the input point cloud data is derived by aerial acquisition. This is because the presence of two connected roof segments with similar slope makes a building with a gable roof highly likely. Yet, it is still necessary to search and associate the corresponding facades to the detected generic gable roof segment pairs. This reduces to the search of planar inliers underneath labeled roof segment pairs. Ideally, these are already labeled as facade segments in the output of Chapter 6. Basically, we could search for the labeled facade segments by comparing the location of the centroids of each labeled facade segment with the hull of the corresponding GSP. Unfortunately, a well defined segmentation cannot always be ensured for all facades, e.g., due to missing data or also non-isolated buildings with complex footprints. We, thus, resort to polygon sweeping shown in Figure 7.5 and described in the next section.

7.1.1 Polygon Sweeping

We use a restricted form of the widely employed plane sweeping algorithm (Collins 1996, Gallup et al. 2007). The goal is to detect building facades and to represent them as polygons underneath corresponding labeled roof segment pairs of GSPs. To this end, polygons are swept along and orthogonal to a sweeping line (e.g., yellow line in Figure 7.5). In the illustration depicted in Figure 7.5, the ridge line is used as sweeping line, i.e., polygons are swept along l_a (moved in discrete steps). At every sweeping step, the polygonal chain forming this configuration is assessed using the MSAC likelihood function $\mathcal{L}(\boldsymbol{c}_u)$. The process is repeated in the orthogonal direction to the ridge line as well. To speed up this process, we use a locally cropped version of the input point cloud data set $\mathcal{D}$. The extent of sweeping along the sweeping lines l_b and l_a, is defined based on the convex-hull polygons of the roof segments along and orthogonal to the sweeping line. For example, if h_{max} defines the extent of the convex-hull orthogonal to the ridge line and the vertical (up) direction, then l_b is defined as: $l_b = h_{max} + \mathrm{d}$, with tolerance d. Similarly, if the maximum extent of the hull along the direction of the ridge-line is r_{max}, then $l_a = r_{max} + d$. The value of the tolerance is determined based on how close the buildings are to one another. Throughout our experiments, we empirically set this value to d = 1m (see Table 8.1). To account for missing data, we apply the following heuristics: If one facade is missing, we assume symmetry and replicate it from the detected parallel facade. If both parallel facades for a roof segment pair are missing, we use the extent of the convex-hull of the roof segment to localize the facades. Using the detected facades and the planes derived from the roof segments, we compute the points $p0, p1, p2, p3$ (cf. Figure 7.5) from the intersection of three planes: the two pairs of facade planes derived by polygon sweeping, and one roof segment plane.

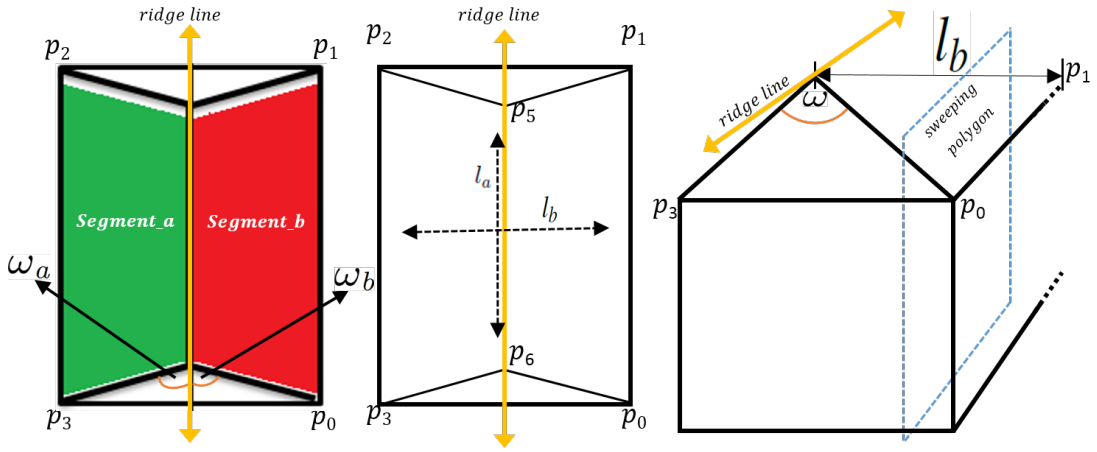

Figure 7.5: Polygon sweeping along and orthogonal to the sweeping line (yellow line) to localize the true facades

7.1.2 Model Selection

Polygon sweeping as described above only leads to basic gable roof models as well as the associated MSAC scores. Additionally, we also capture symmetric and asymmetric (half)-hipped as well as pyramid roof models using the procedure sketched in Figure 7.6, varying the locations of points $\mathbf{p}_5, \mathbf{p}_6$ along the ridge line (Figure 7.6) in discrete steps. At each step, a new configuration, i.e., a polygonal chain consisting of four facades and a roof is derived. Also, these additional configurations are evaluated by an MSAC score using the likelihood function $\mathcal{L}(\boldsymbol{c}_u)$ on the input point cloud data $\mathcal{D}$. The final model is chosen as the polygonal chain that "best" explains the data, i.e., with highest MSAC score.

This approach to modeling buildings inherently assumes aerial data acquisition. It is usually not suitable for 3D point clouds derived from terrestrial sensors, e.g., by SfM/MVS technique on images taken by handheld cameras. In the next section, we present our solution to modeling buildings in 3D point clouds from terrestrial data.

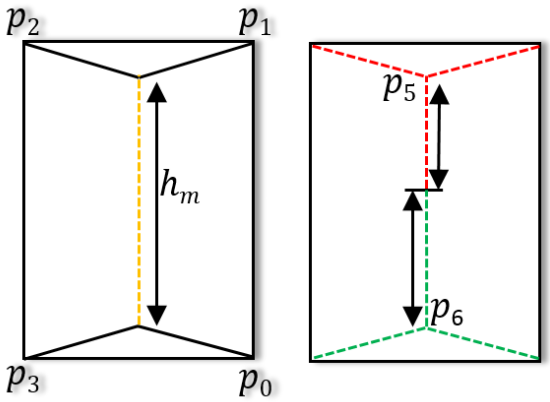

Figure 7.6: Stepwise variation of the locations of points $\mathbf{p}_5, \mathbf{p}_6$ along the ridge line produces new configurations capturing the family of gable, hipped as well as pyramid roof type.

7.2 Polygonal Chain Configurations from Terrestrial Data

3D point clouds acquired from the ground often have more complete data for the facades of buildings than for the roofs. We use a bottom-up approach on these data sets to fit shape primitives. We assume footprints of buildings are provided as GSPs or we detect them using the depth-first search algorithm on the undirected graph of connected facades as described in Section 6.3.2. The footprints are used as priors to guide the stochastic search for the best polygonal chain configuration. Our approach is based on two major assumptions:

- The footprints are quadrilaterals or already decomposed into these.

- The true facades are correctly described by the footprints.

Thus, the major task is to find and fit suitable polygonal chain configurations for the roof above the quadrilateral footprints. In Figure 7.7, we show an illustration of a gable roof using this stochastic roof search and fitting scheme employing four roof configurations (coloured black, green, blue and yellow) and three height levels (l_0, l_2, l_2) above the footprint (NGUATEM et al. 2013b).

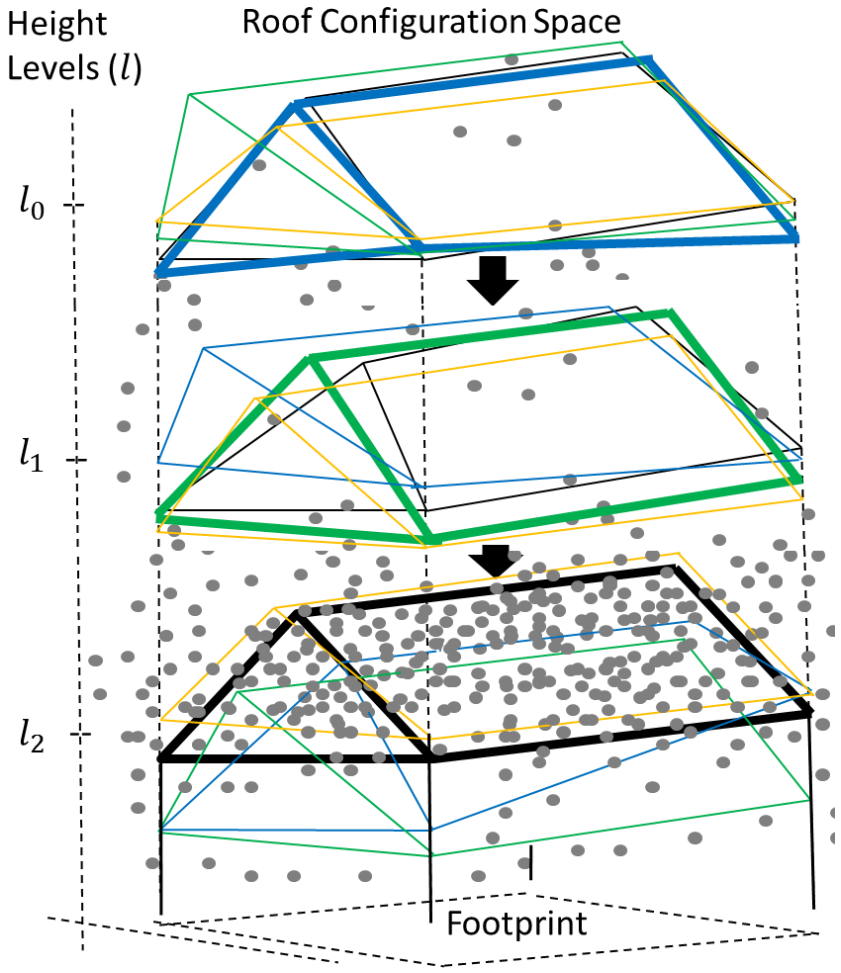

Figure 7.7: Illustration of the roof configuration space for the gable roof: At distinct height levels, the likelihoods of each configuration is estimated and the configuration with highest likelihood is marked in bold. The gray dots represents the data points and the different colors correspond to different configurations (NGUATEM et al. 2013a).

First, we build initial roof configurations (level l_0) and score them using the likelihood function described in Equation 7.1. Then an iterative transition is made along the vertical direction to the next level below and the roof configuration likelihoods are

updated. At each level, the configuration with the highest likelihood (marked bold in Figure 7.7) explains our data better than the other configurations. At height level l_2, the configuration represented in black provides the "best" fit so far, hence, its polygonal chain is chosen as the final model.

We start by generating initial polygonal chain configurations for roofs. This means to choose an appropriate template for the various basic roof types and to randomly vary the parameters according to the corresponding prior distributions. One could extend the generic mansard roof parametrization proposed in (Poullis and You 2009) to include all symmetric and asymmetric variants of these basic roof types. However, this would be too computationally expensive if a scene has very few mansard roofed buildings. We have, thus, developed two unique templates: one for the generic gable roof and the other for the mansard roof which we present in Figure 7.8.

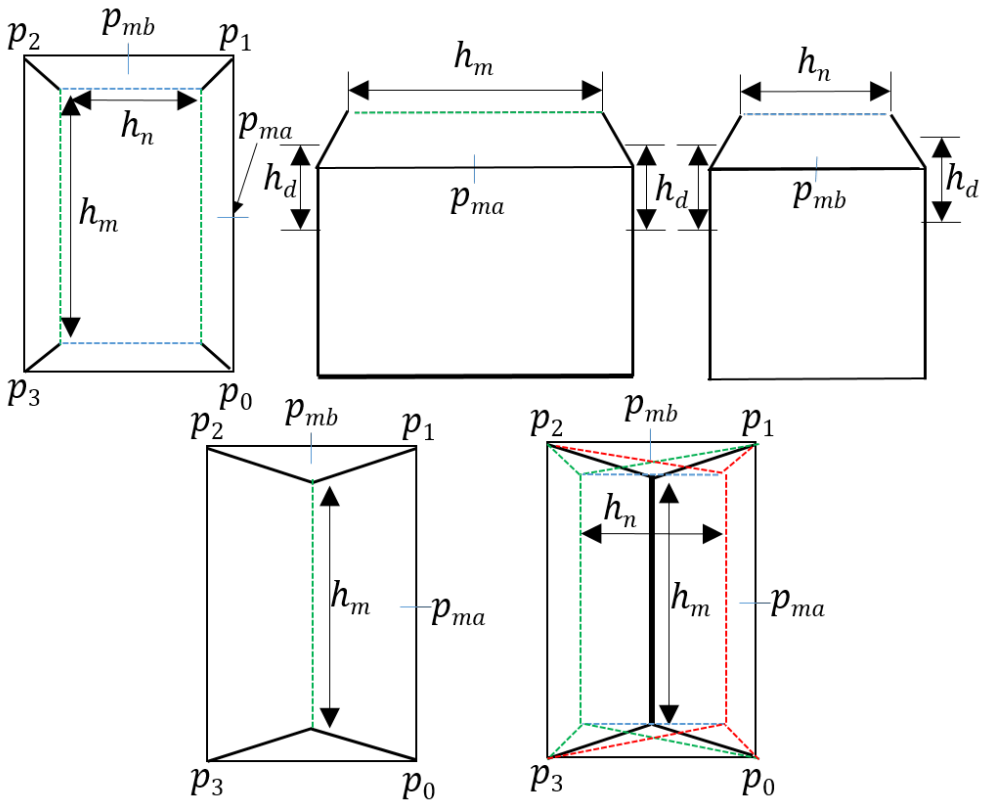

Figure 7.8: Generic templates for both Mansard (top row) and gable roof (bottom row), with a parametrization capturing all possible roof configurations of these types (Nguatem et al. 2013a)

Before elaborating on the roof parameterizations, we first define a prior for the building facades heights, p0, p1, p2, p3. This prior compensates for the uncertainty in the estimates of the heights of the building facades given the footprint. We draw four

samples for the points p0, p1, p2, p3 as follows:

$$p^{(i)} \sim \mathcal{U}(p_k - 0.5 * h_d, p_k + 0.5 * h_d,) \tag{7.3}$$

where $k \in \{0, 1, 2, 3\}$ stands for the respective point p_k, h_d is the height tolerance as depicted in Figure 7.8 and i the index of the i-th configuration. These four draws representing a prior over the height of the facade are fundamental for all roof types. In addition to equation (7.3), we use domain knowledge and derive templates and prior distributions for the ridges, peaks, hips, etc. By enabling the parameters to vary, this gives a complete prior for a certain roof type as we describe below.

Generation of Gable Roof Configurations
Figure 7.8 (bottom row) illustrates the parametrization of the gable roof. A prior over $\mathbf{p}_{mb}$ combined with Equation (7.3), giving the prior over the building height, defines our informative prior model for the gable roof. Samples for the ridge line are derived from realizations of $\mathbf{p}_{mb}$ and are given by

$$p_{mb}^{(i)} \sim \mathcal{N}(p_{mb}, h_t) \tag{7.4}$$

with h_t the tolerance in the location of the ridge line. We vary along the detected ridge line to determine if a pinnacle roof or a hipped roof better explains the data than the simple gable roof as described in Section 7.1.2.

Generation of Mansard Roof Configurations
Likewise, Figure 7.8 (top row) shows the parameterization of the mansard roof.

$$h_n^{(i)} \sim \mathcal{U}(0, h_n), \qquad h_m^{(i)} \sim \mathcal{U}(0, h_m) \tag{7.5}$$

where hn and hm are tolerance values defining the lengths of the ridges and again, i is the index representing the i-th configuration.

Other Roof Types
The most basic roof types are the flat and the shed roofs. Their parametrization is obtained by varying the four points defining the building height, i.e., Equation (7.3).

Model Selection
We illustrated the bottom-up approach to searching and fitting polygons on point clouds using a gable roof model. Yet, the ground truth could be a mansard building. Therefore, for a detected quadrilateral footprint, we search and fit gable, mansard, hipped, pinnacle as well as flat roof configurations. The most likely configuration is the one with highest MSAC score, hence is chosen as the final model. Figures 7.9 and 7.10 show point cloud data (in green) of buildings with pinnacle and mansard roof, respectively. The results of the best fitting pinnacle, gable as well as mansard roof models on these point clouds (yellow points are inliers) are shown. It can be seen that,

while the best fitted model for the data in Figure 7.9 is a pinnacle roof, the mansard roofed model fits the data "best" in Figure 7.10. In accordance with the likelihood, these models have the highest MSAC scores.

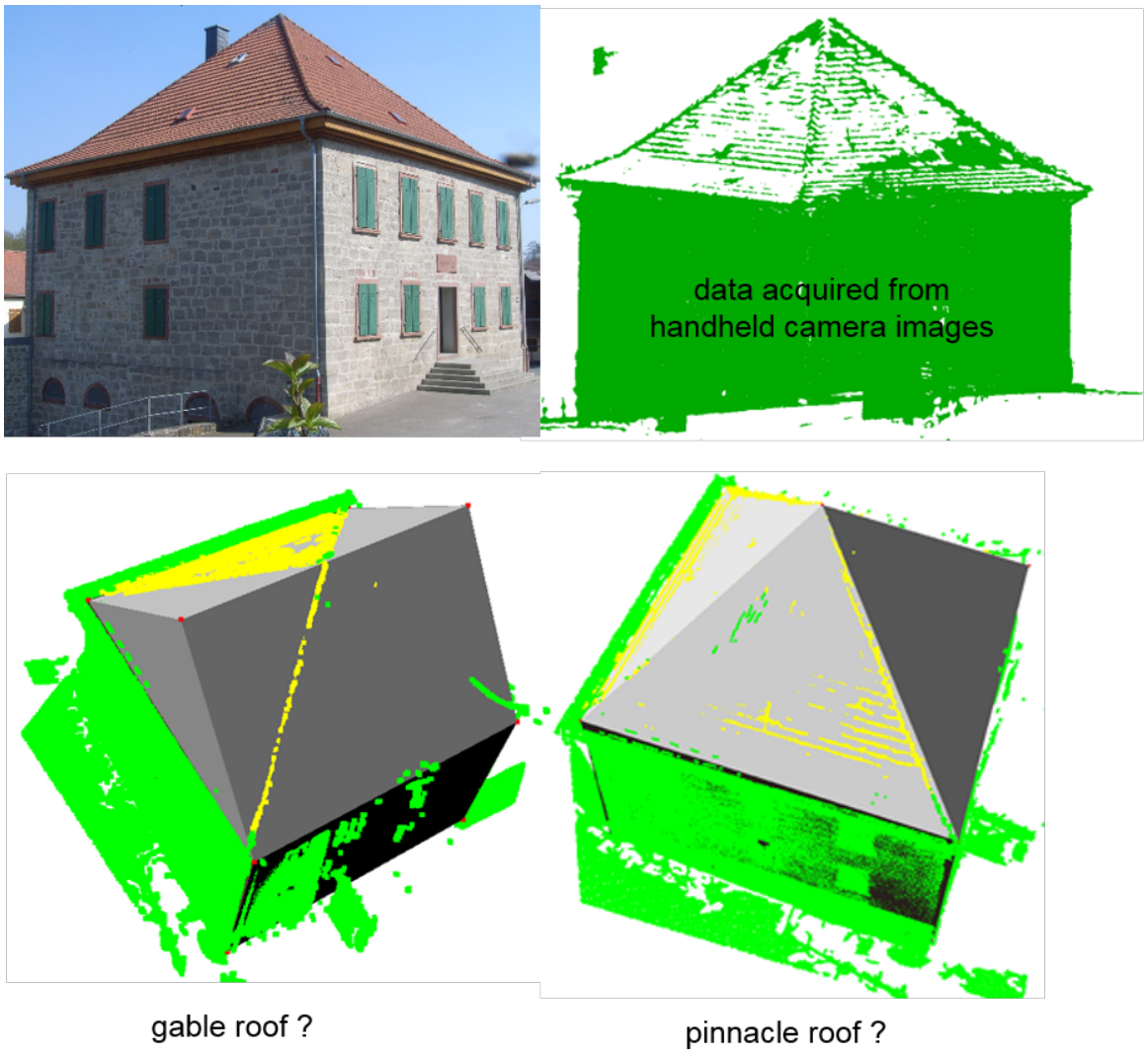

Figure 7.9: Model Selection: Best fit for pinnacle and gable roof type for a building with a pinnacle roof based on a point cloud generated by SfM/MVS from images by hand held cameras. The yellow points are the inliers to the "best" 3D polygonal chain. As expected, the pinnacle roof model explains the data best by providing the best fit.

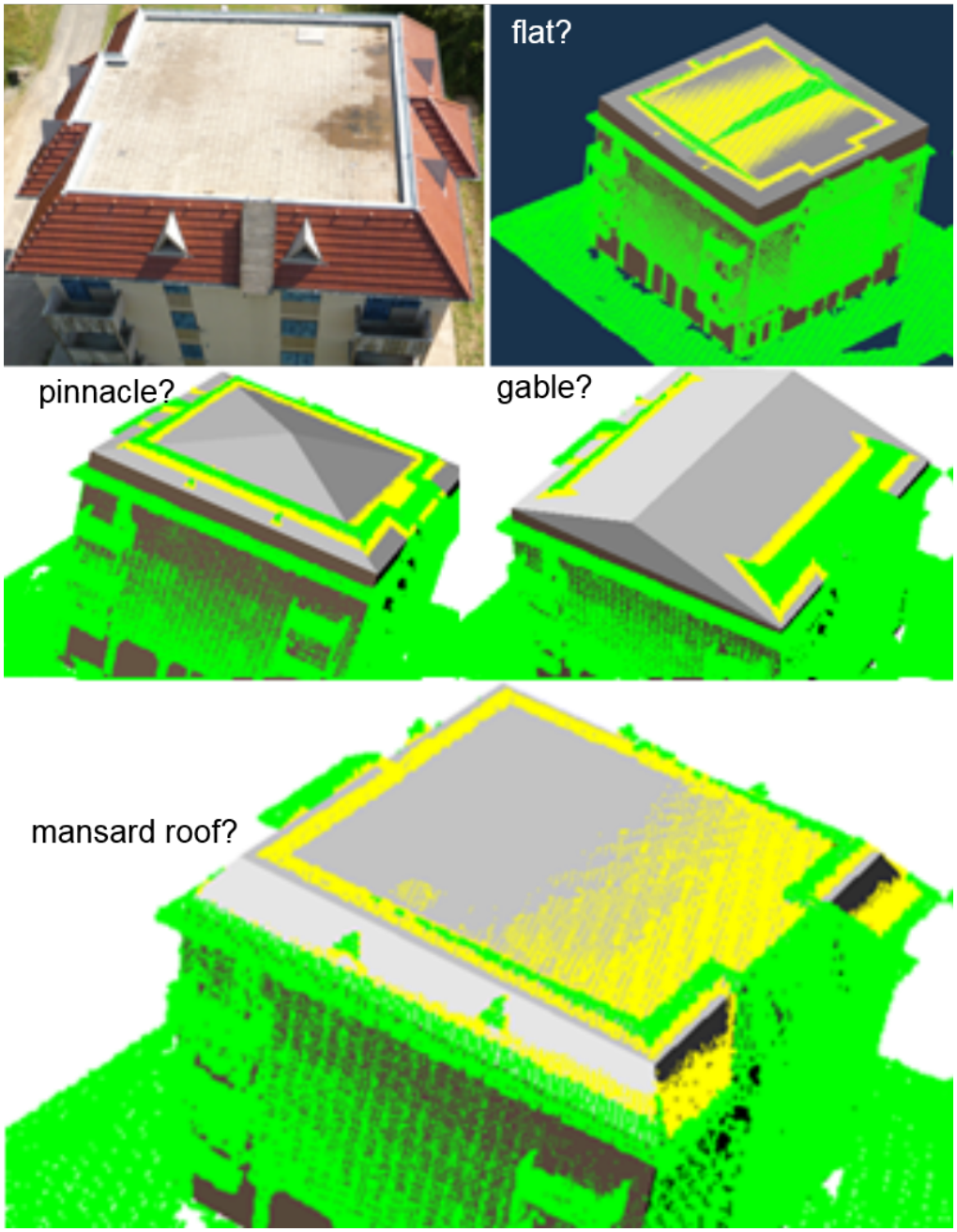

Figure 7.10: Model Selection: Best fit for flat, pinnacle, gable and mansard roof type on a building with mansard roof obtained by SfM/MVS. The yellow points are the inliers to the "best" 3D polygonal chain. As expected, the mansard roof model explains the data best by providing the optimal fit.

7.3 NURBS Surface Fitting and Facade Trimming

An important goal of our work is to model urban scenes with a compact representation whilst preserving the scene semantics. The modeling of a possibly non-planar ground with a plane provides the lowest possible complexity in terms of the number of polygons and, hence, the highest compactness. This approach however, does not fit the data well and often results in the loss of inherent natural smoothness. We resolve this problem using Non-Uniform Rational B-Spline (NURBS) surfaces. Surface modeling with NURBS has two major challenges to be solved: (1) Finding appropriate so-called control points and (2) defining the points that will form a mesh, e.g., for rendering applications. For the former, we apply knowledge from the scene voxelization obtained in the initialization stage during scene segmentation presented in Section 5.2. Particularly, we project centroids of voxels belonging to the ground on their clipped planes and use these as control points. The second problem is called tessellation (Gross and Pfister 2007) and it consists of transforming the mathematical continuous definition of the surface, $S(u, v)$, to a discrete approximation. For complexity and efficiency reasons, we restrict ourselves to a uniform tessellation.

To achieve a more compact representation while preserving the semantics, we limit the extent of the facade to the tessellated ground. For this, we perform Boolean operations on the surfaces by locating collision points of the facade polygons with the tessellated ground mesh and trim the facades beyond these collision points. Figures 7.11 and 7.12 show results of our workflow before and after facade trimming.

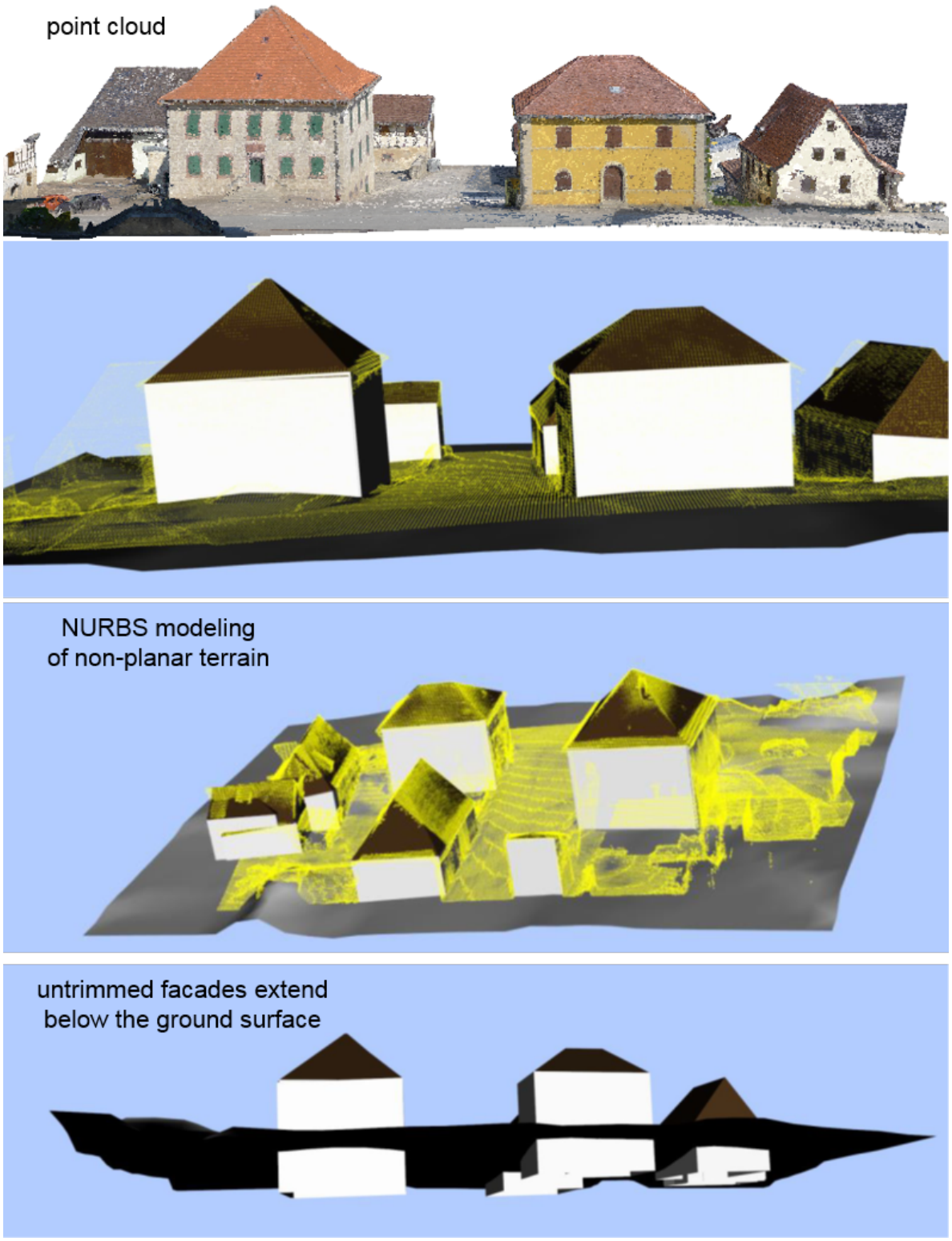

Figure 7.11: The pipeline for modeling buildings from point clouds results in facades protruding below the ground surface which is modeled using NURBS. Downsampled input data (in yellow) are superimposed on the model to show the quality of the fit.

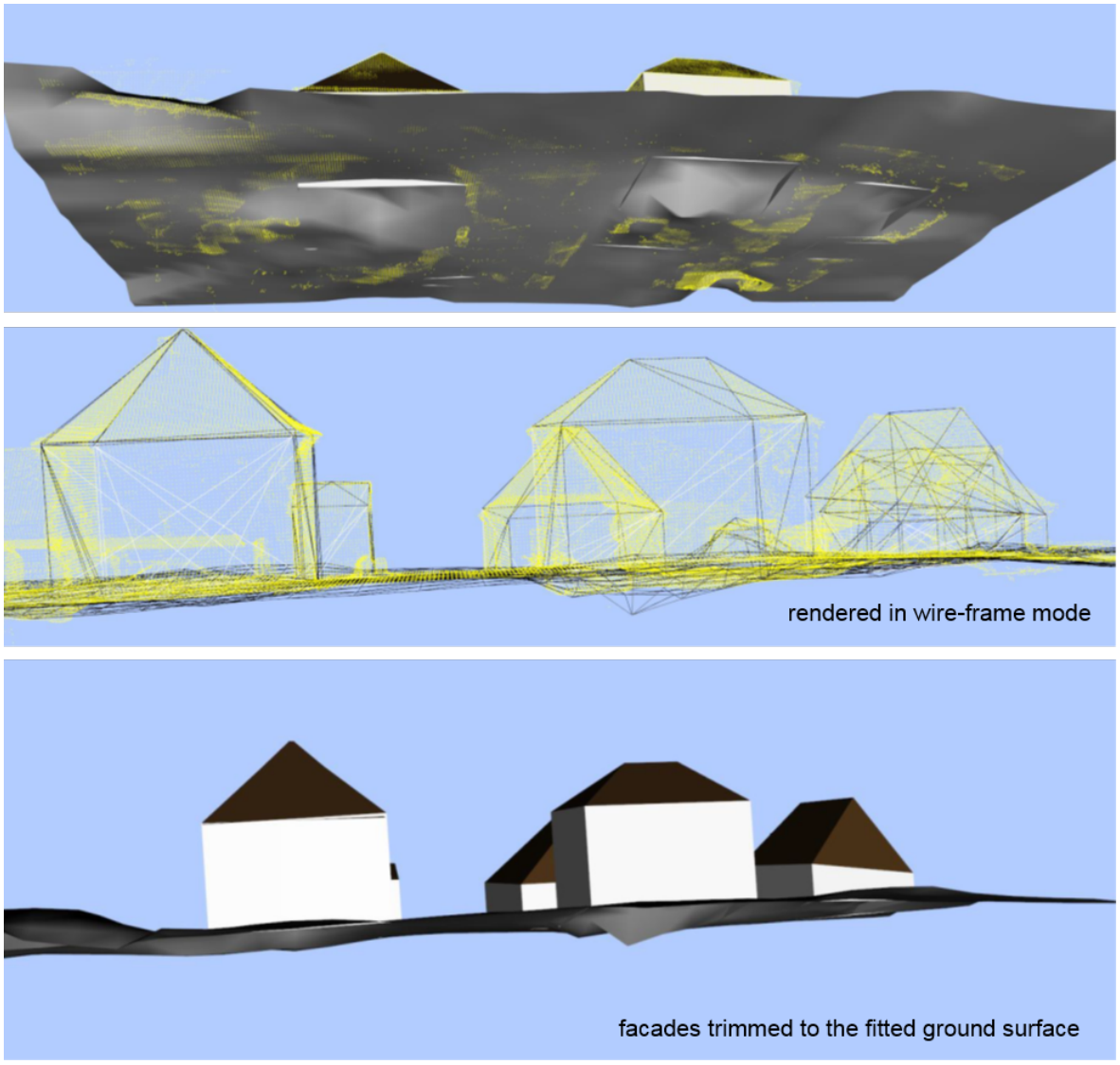

Figure 7.12: A more compact representation is achieved after trimming the facades. Downsampled input data (in yellow) are superimposed on the model to show the quality of the fit.

Chapter 8

Experiments

The previous chapters have presented a pipeline for modeling urban scenes from 3D point clouds derived from both terrestrial as well as aerial data. It operates in three major stages namely, segmentation, scene interpretation and geometric shape fitting, as described in Chapters 5, 6 and 7, respectively. In this chapter, we study the performance of the pipeline using various datasets and the text loosely follows (Nguatem and Mayer 2017). We conduct a variety of experiments with focus on different aspects of our algorithms such as robustness to noise and missing data as well as scalability. Parts of our algorithms, e.g., segmentation, are compared to state-of-the-art algorithms where applicable, substantiating our pipeline. This is also an important basis for finally discussing our results. The datasets have been acquired from scenes with isolated buildings as well as urban residential areas with densely and irregularly structured buildings. The terrain in the scenes ranges from flat to highly non-planar. We use real-world 3D point clouds for all experiments. They have been acquired either by state-of-the-art Structure from Motion followed by Multiview Stereo reconstruction (SfM/MVS) techniques, e.g., by tools such as Agisoft Photoscan, Pix4Dmapper and (Kuhn et al. 2017, Schönberger et al. 2016, Huang et al. 2019), or produced by terrestrial LiDAR (Nüchter 2016).

8.1 Datasets

Synthetic point clouds from sampled 3D CAD models, e.g., the Princeton shape database (Shilane et al. 2004) are noise-free and mainly composed of free-form shapes, including cars and sofas but not buildings. We validate the proposed algorithms on point cloud data from two major acquisition techniques, namely SfM/MVS and LiDAR. These datasets stem from publicly available sources or are self acquired using a standard consumer camera such as the Panasonic LUMIX DMC-LX5. The only manual intervention applied was to ensure that the input point cloud data have a known metric scale and a consistent vertical up direction vector. To this end, we manually scale the input point clouds to a known metric scale and rotate the scene's z-axis so that it is aligned with the scene's vertical up direction, if required.

8.1.1 Point Clouds from LiDAR

We conducted experiments with terrestrial mobile laser scans from (Nüchter 2016) as well as self acquired LiDAR scans. The data range from a few hundred 3D points to very large scale data sets containing billions of 3D points (Nüchter 2016). Acquiring 3D

scans from large areas is difficult. Usually, multiple scans are recorded per position and are postprocessed (co-registered) in an offline procedure. The latter fuses the individual scans forming a single consistent point cloud. We use the 3D Simultaneous Localization and Mapping (SLAM) software provided by (NÜCHTER 2016) to merge all scans into a single point cloud $\mathcal{D}$, as required by our modeling pipeline. A portion of this data set is shown on the bottom of Figure 8.1.

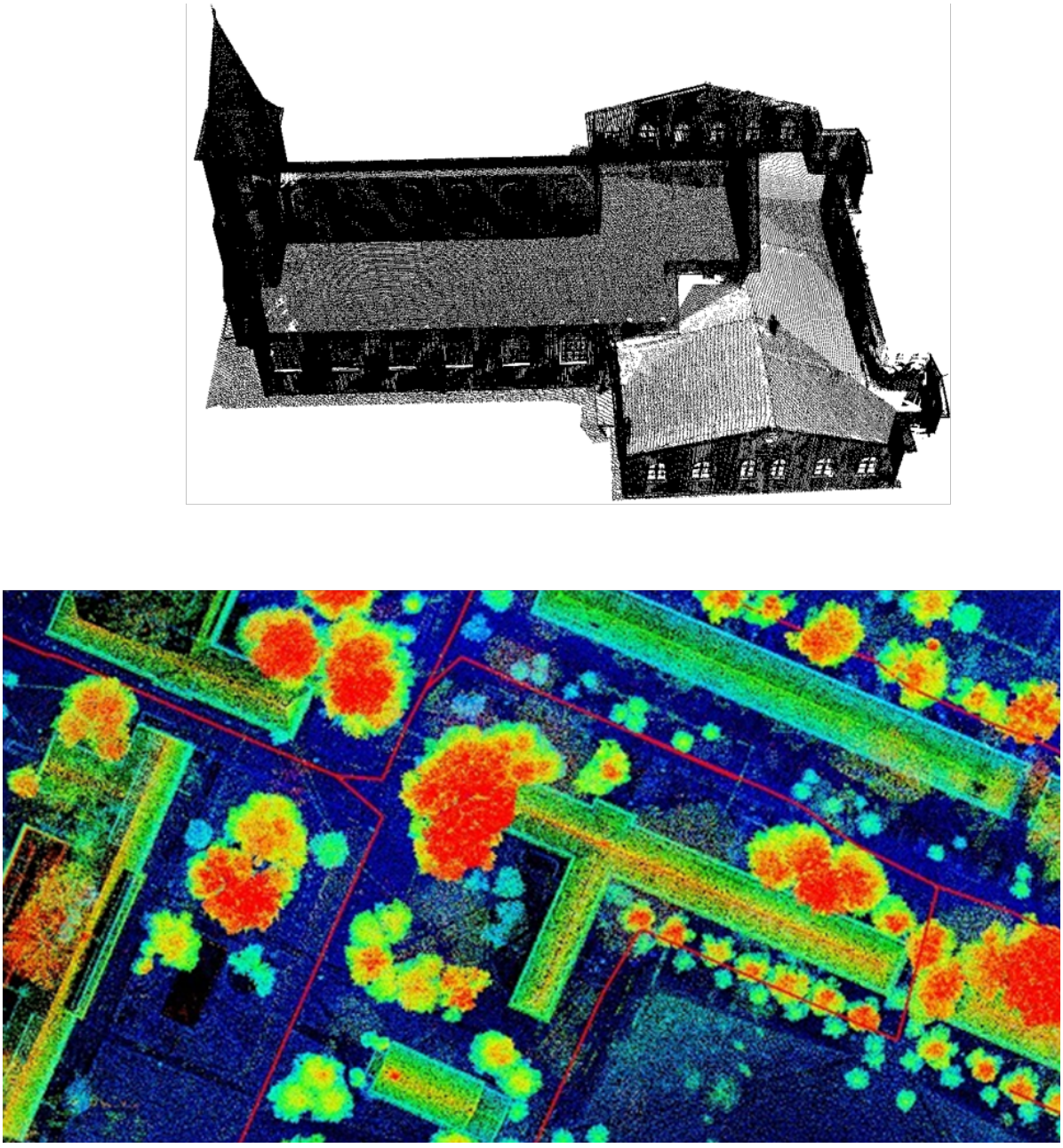

Figure 8.1: Two samples from our test data acquired via terrestrial laser scans. Top: Result from 10 self-acquired scans after semi-automatic co-registration. Bottom: Point cloud data from (NÜCHTER 2016) containing 2.2 billion 3D points

8.1.2 Point Clouds from Images derived by SfM/MVS

Point clouds derived from images by state-of-the-art SfM/MVS techniques, e.g., using tools such as Agisoft Photoscan, Pix4Dmapper and (KUHN et al. 2017, SCHÖNBERGER et

al. 2016, Huang et al. 2019) are becoming very popular due to the tremendous progress made by the development of new algorithms (Furukawa and Ponce 2010, Frahm et al. 2010, Hirschmüller 2008, Huang et al. 2019) and the availability of (cheap) dedicated hardware for the acceleration of computation such as Graphic Processing Units (GPUs). Though 3D point clouds from SfM/MVS can have significant point density variations as well as accuracy issues, the ease and flexibility of data acquisition using standard consumer cameras compared to LiDAR technology is very beneficial for our application. Our 3D point cloud data consist mostly of buildings and vegetation of varying sizes as is often the case in urban scenes. We combine images taken from the ground using handheld cameras with those captured from UAVs when possible. The point clouds often show fine details such as chimneys and balconies, but also challenges such as noise, reconstruction artifacts, holes, point density variations and topological defects. The number of points ranges from a few hundred up to hundreds of millions for scenes with many buildings. Examples are shown in Figure 8.2.

Besides the difference between the two data acquisition modalities mentioned above (SFM/MVS and LiDAR), major characteristics of the input point clouds $\mathcal{D}$ arise from the complexity of the terrain, point density variations, object size diversity as well as building density and structural differences in scale and orientation. Furthermore, substantial vegetation is often present in the point cloud data, since vegetation is common around buildings in urban environments. As our pipeline does not model vegetation, it is considered as noise as well as scene clutter and hence undesirable, giving room for future work as will be discussed in the next chapter.

Figure 8.2: Examples of point cloud data from our test set. This data is derived from images via SfM/MVS. It ranges from 3D point clouds of single buildings to an urban block of 51 buildings captured with images of handheld cameras combined with images from UAVs.

8.2 Implementation

We have implemented our entire modeling framework (segmentation, scene interpretation and geometric shape fitting with model selection) in C++. Special attention was given to scalability by using data structures such as octree and k-d tree (Muja and Lowe 2014) where appropriate to enhance data locality and availability via spatial decomposition, speeding up search. We also use only two software libraries published under the open source licenses Berkeley Software Distribution (BSD) and Mozilla Public License (MPL):

- Point Cloud Library (Rusu and Cousins 2011): Used mainly for data handling, i.e., for reading the input point cloud data as well as for saving all intermediate results to the disc.
- Eigen (Guennebaud et al. 2010): Employed as major backend for all linear algebra computations.

An advantage of this is the ease of a possible integration into larger software systems due to the reduced dependencies and no license restrictions.

Our major goal has been the completeness of the pipeline rather than runtime optimizations. Nevertheless, we also evaluated the performance of the system with respect to runtime efficiency. All tests and timings were performed on a Win7/x64 system with an Intel(R) Xeon(R) CPU E5507 and 64GB RAM memory. Unless otherwise stated, our system chooses random colors for the output of the segmentation, yet assignes three major colors to the polygon chains constituting the resulting CAD/watertight model as follows

- dark brown: roof
- silver brown: facades
- silver: ground

8.3 Design Parameters and Practical Considerations

The proposed pipeline for modeling urban scenes from noisy 3D point clouds is a complete and fully automatic system. It combines automatic segmentation, semantic interpretation and geometric shape fitting with model selection. Surprisingly, the overall system has only a few adjustable parameters. These are summarized alongside their default values in Table 8.1. Throughout all experiments, unless otherwise stated, all parameters are used as presented in this table.

In Chapter 5, we employed the Normal-Inverse-Wishart (NIW) distribution within Gibbs sampling to solve the problem of surface normal clustering in a Bayesian nonparametric setting called the Chinese Restaurant Process (CRP). The NIW distribution served as the base distribution, G_0, in Algorithm 1. It takes four parameters namely,

Table 8.1: Design parameters used and their respective default values

Operating Stage	Parameter	Default Value
Segmentation	RANSAC inlier threshold	0.3 meters
	Voxel leaf size, l_s	1.0 meter
	# MCMC Iterations (level 1)	500
	# MCMC Iterations (level 2)	10
	Max # Clusters (upper bound), NrMax	40
Scene Interpretation	Angular tolerance, $\epsilon 1$	6°
	Angular tolerance, $\epsilon 2$	30°
Shape Fitting	Roof tolerance for Polygon-Sweeping, d	0.5 meters
	MSAC inlier threshold	0.3 meters
	Tolerance in location of the ridge line, h_t	1.0 meter

μ_0, κ_0, Ψ_0, υ_0. Given these parameters and Algorithm 1, the obvious question is, how we can draw samples from NIW? This is a two stage procedure and is explained as follows: A sample from a NIW distribution consists of a mean vector and a covariance matrix which in our case defines a multivariate Gaussian distribution. First, we sample a covariance matrix Σ from an inverse Wishart distribution (Anderson 2003) with parameters Ψ, κ, υ and then a mean vector μ from a multivariate normal distribution with mean μ_o, and variance $1/\kappa\Sigma$. These four parameters can be described as follows:

- a positive scalar $\upsilon > D-1$ defining the degrees of freedom, where $D = 3$ represents the dimensionality of the data:
- Ψ is a positive definite matrix in $\mathbb{R}^3$
- κ is a positive scalar
- μ is a vector in $\mathbb{R}^3$

The vectors and matrices are in $\mathbb{R}^3$, i.e., in accordance with the dimensions of a 3D point in the point cloud data. For all experiments, we set the values for the four parameters for the NIW required as a base distribution to $\mu_0 = (0,0,0)$, κ_0=1, Ψ_0=$\mathbf{I}_3$, υ_0=4, where $\mathbf{I}_3$ is the identity matrix in $\mathbb{R}^3$.

8.4 Evaluation of the Segmentation

The robust segmentation of the point cloud into meaningful patches is the first step in our pipeline for creating 3D models of urban scenes from point cloud data The proposed segmentation algorithm combines the strength of RANSAC, clustering based on infinite mixture models of Gaussians, and region growing. The performance of RANSAC for

detecting parametric models, e.g., planes, has widely been studied, e.g., in (SCHNABEL et al. 2007). Hence, we limit our discussion to the convergence of the Gibbs sampler and the accuracy of the infinite mixture of Gaussian models for clustering of clipped plane normals (CPN). We demonstrate both by studying the effects of changing the value of the concentration parameter (α presented in Algorithm 1) to the segmentation results. Furthermore, we substantiate our choice of the voxel size in this section.

8.4.1 The Concentration Parameter α

In Section 5.3, we explained that the concentration parameter α determines the number of clusters (MUELLER et al. 2015). Larger values of α lead to many more clusters being found, because many isolated CPNs on the unit sphere will be assigned to unique clusters. We used the value $\alpha = 2.0$ for all experiments within this thesis. To study the robustness of our segmentation algorithm with respect to different values of α, we used the dataset depicted in Figure 8.3 containing a single building. The point cloud is acquired by SfM/MVS from images taken from the ground as well as UAV flights. Besides noise and reconstruction artefacts, the scene shows three major orientations, i.e, it consists of just one Manhattan-Frame. Thus, ideally, clustering CPNs should produce three major clusters: Two for all facade points and one for the ground and the points associated with the flat roof.

Figure 8.4 shows the evolution of Gibbs sampling for 500 iterations using three different values of α. At first glance, the results confirms what is written in (MUELLER et al. 2015) by demonstrating that increasing the value of α, many more clusters are found. A deeper look is given by Figures 8.6, 8.7 and 8.8 showing the distributions of the CPNs assigned to the clusters found after 500 iterations confirming the overall increase in the number of clusters by increasing the value of α. Yet, the percentage of CPNs belonging to the three major orientations in the scene as given by the first three bars in Figures 8.6, 8.7 and 8.8 representing the facades as well as the ground and roof, remains basically unaffected by changes in α. The increase in number of clusters is because many CPNs (probably from voxels with substantial noise, reconstruction artefacts and vegetation) tend to be assigned to their own cluster. Nevertheless, our proposed divide and conquer framework inherently adapts to this behavior as connected component analysis resolves issues that may arise if too many clusters are created, e.g., due to noise of vegetation.

We have studied the clustering dynamics, by looking at the number of clusters the, number of CPNs per cluster, and the number of Gibbs iterations required to convergence as shown in Figure 8.4. The complete analysis described above is performed on the first level of our two-level hierarchy presented in Section 5.3, i.e., using the "coreset", hence with initial random assignments. Although initial random assignments are used for cluster initialization, the Gibbs sampling converges on average in less than 200 iterations to the correct k, the number of representative clusters present in the data. Also, the results in Figure 8.5 show that the segmentation is robust against changing the value of α.

Figure 8.3: Point cloud derived from images by SfM/MVS. Besides reconstruction artefacts and noise, this scene has three major orientations: Two orientations for the walls and one for the ground together with the flat roof, making it particularly suitable for assessing the effects of changing parameters for clustering.

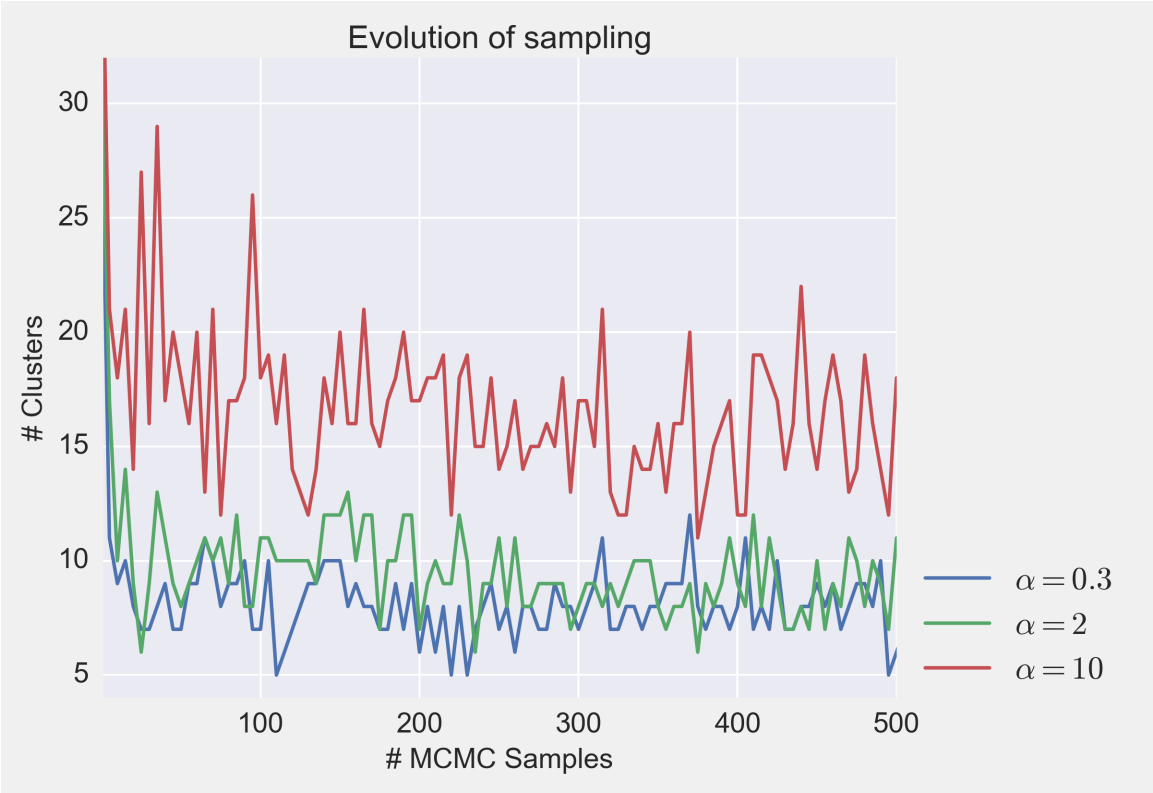

Figure 8.4: Evolution of Gibbs sampling showing convergence after about 200 iterations for three different values of α

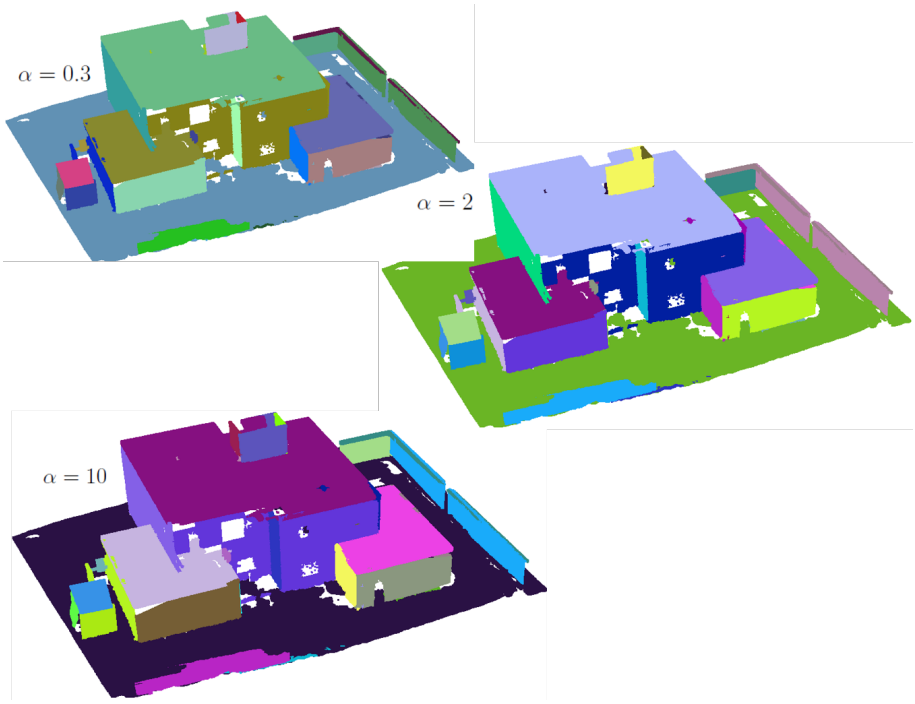

Figure 8.5: The segmentation produces very similar results for different values of α. All patches are randomly colored.

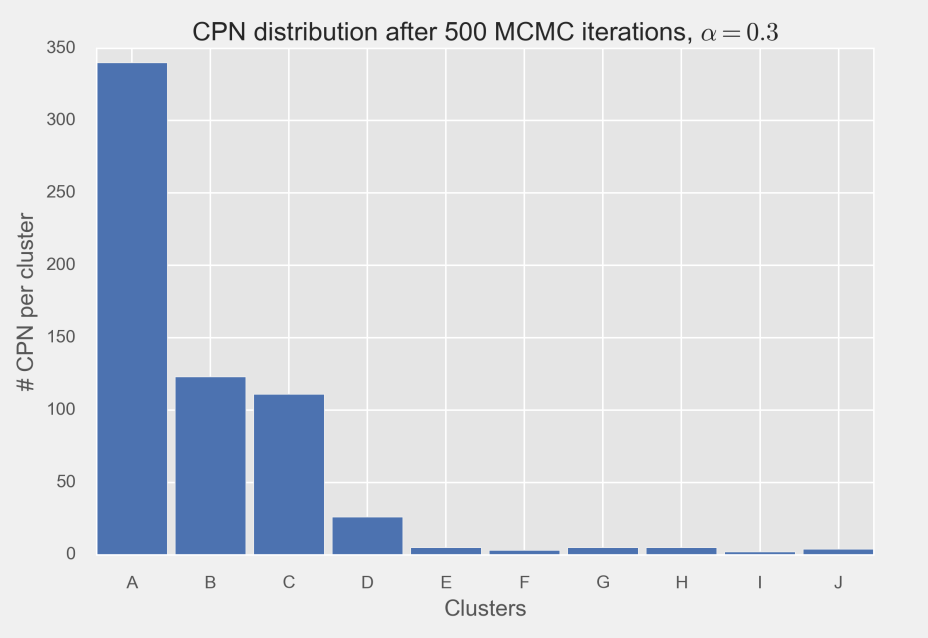

Figure 8.6: Resulting distribution of CPNs to clusters for $\alpha = 0.3$. Three distinct orientations of the scene are represented by the first three bars.

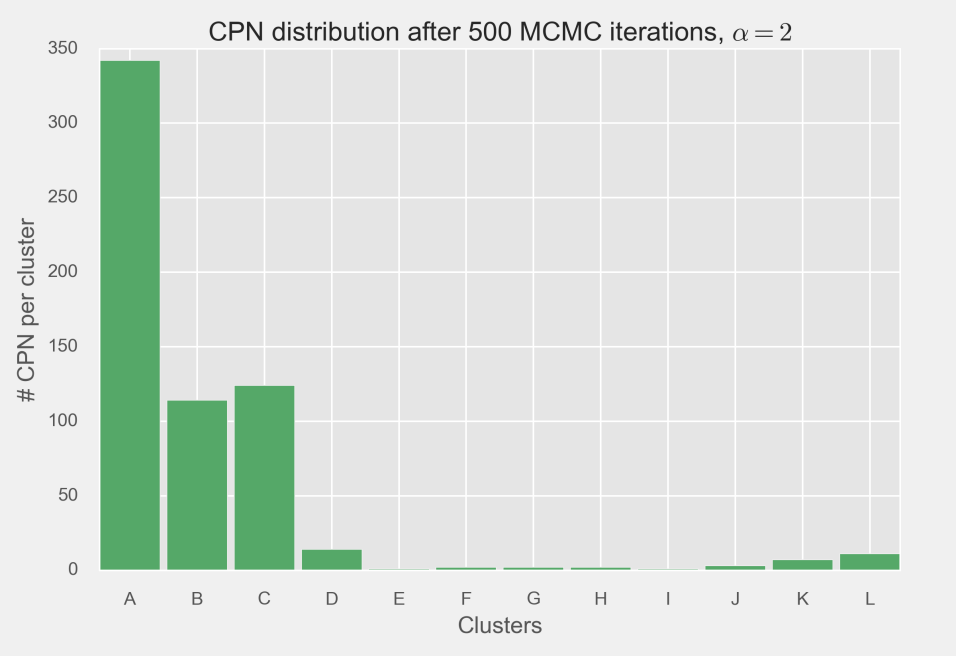

Figure 8.7: Distribution of CPNs to clusters for $\alpha = 2$. Three distinct orientations of the scene are represented by the first three bars.

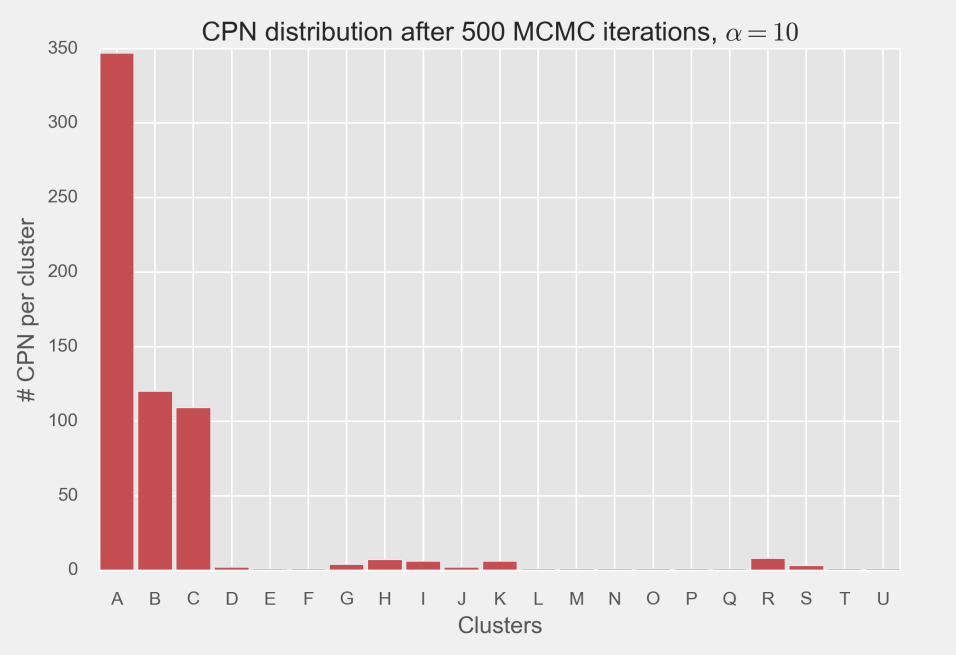

Figure 8.8: Distribution of CPNs to clusters for $\alpha = 10$. Three distinct orientations of the scene are represented by the first three bars.

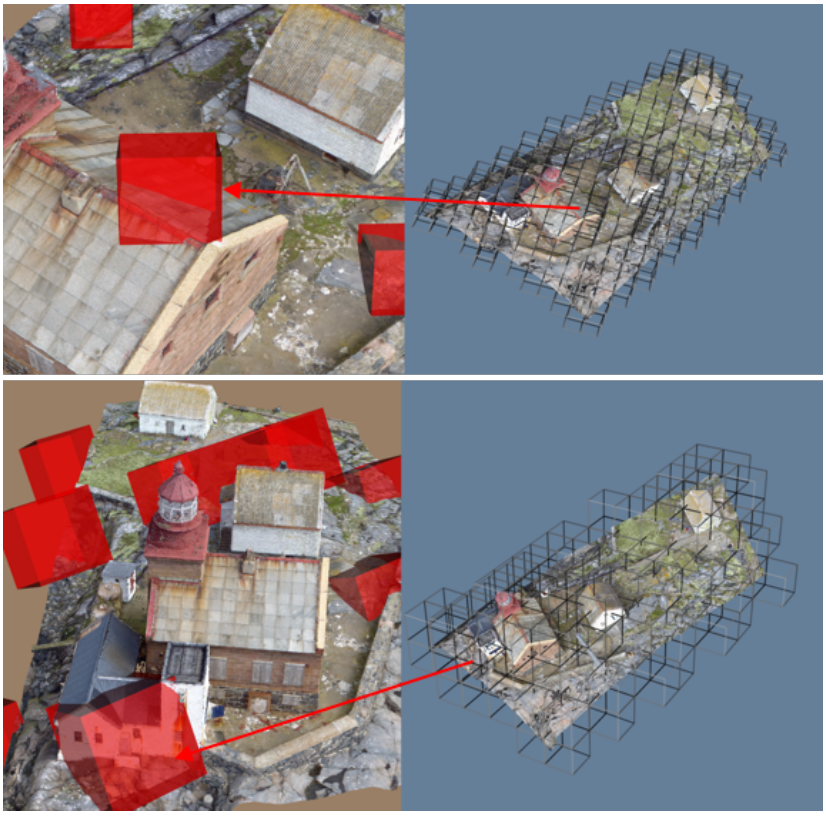

Figure 8.9: Choosing the right voxel size is crucial for the success of our modeling pipeline. If the voxel size is too small, many fine details, e.g., windows or doors, are modeled. In the top row, an appropriate voxel size of 2m is chosen as compared to the voxel size of 10m in the bottom row.

8.4.2 Choice of Voxel Size and RANSAC Threshold

An important step in the first stage of our segmentation algorithm is the division of the scene into non-overlapping fixed-size voxels. The outcome of the segmentation is sensitive to the choice of the size of the voxels. To choose an adequate voxel size, it is important to know a-priori the scale of the objects present in the scene. While it may suffice to adopt a voxel size of a few centimeters for segmenting indoor scenes using the proposed segmentation algorithm, larger voxels are appropriate for buildings in urban scenes. Yet, when the voxel size is set too large, a complete wall or building can be within a voxel (see Figure 8.9). Our ultimate goal is to model the essential geometry in the scene creating 3D LOD 2 models of buildings in addition to a possibly non-planar ground. Therefore, inherent fine details, e.g., doors or windows which are distinct at finer scales are not considered for this stage. Empirically, we have found that a voxel size of 1 meter and a RANSAC inlier threshold of 0.3 meters work well for all our datasets. We use these values for all examples presented in this thesis.

8.5 Evaluation of Model Fitting

The main focus of this thesis is to generate accurate 3D models of buildings from raw point clouds derived from LiDAR scans and from images by SfM/MVS techniques. For an objective comparison to existing algorithms as well as to be compared to future work, results for publicly available benchmark datasets consisting of accurate 3D data, e.g., point cloud data, and point-wise annotations or calibrated images and respective pixel-wise annotations are essential. In computer vision and remote sensing, many publicly available benchmark data sets exist. These are, however, focused on tasks different from ours, such as point cloud classification (Rottensteiner et al. 2014, Munoz et al. 2009), 3D object detection and 3D tracking (Geiger et al. 2012), 3D evaluation of MVS reconstruction algorithms (Strecha et al. 2008) and 3D scene understanding (Dai et al. 2017). Also, because of the needs of research in automated driving, annotated 3D data for SLAM and segmentation of urban scenes is becoming very popular (Nüchter 2016, Geiger et al. 2012, Cordts et al. 2016).

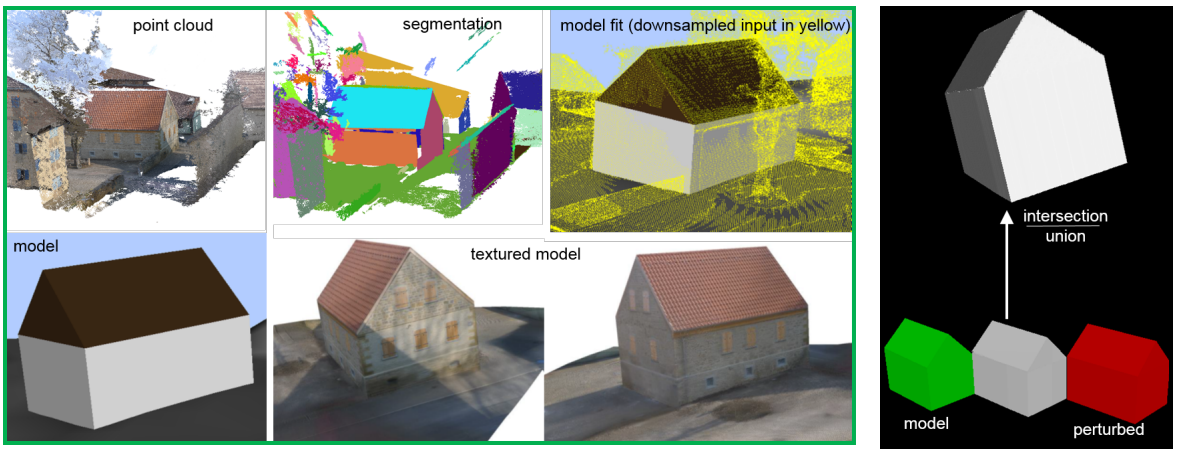

Figure 8.10: In the absence of "true" ground truth, we use this input data from SfM/MVS as "pseudo ground-truth" to analyze the modeling performance under additive noise (Nguatem and Mayer 2017).

However, in the area of research pursued within this thesis, there are currently no benchmarks with a unifying and widely accepted metric, to quantify the quality of 3D models derived from point clouds. This is mainly because the creation of high-quality 3D models, e.g., for the development of immersive games, movies, and virtual worlds, and suitable as ground-truth, is a notoriously difficult task, often requiring hundreds of hours of skilled labor with expensive and sophisticated software tools (e.g., 3ds Max). This problem is exacerbated for 3D content that exhibits fine detail at multiple scales, such as commonly encountered in urban scenes. Thus, for quantitative evaluation of our modeling pipeline, we use the evaluation setup developed in (Nguatem and Mayer 2017). The focus lies on the performance of the pipeline in the presence of missing data as well as noise. This is depicted in Figure 8.10.

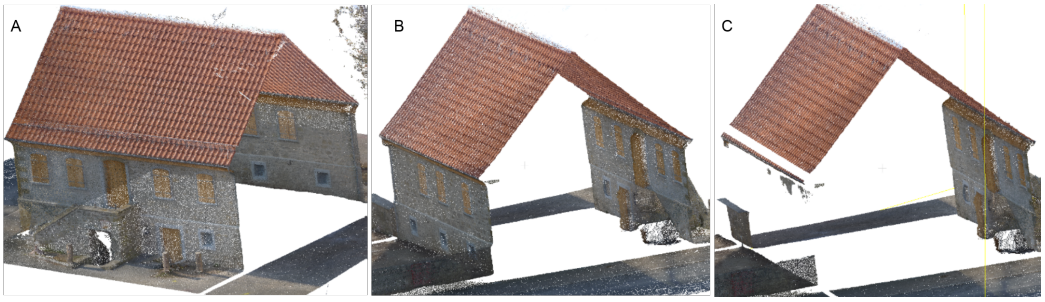

Figure 8.11: Missing data is very common during data acquisition for facades, e.g., due to occlusion. Manually deleting facades from the test data, creates modified data reflecting this reality. The comparison of the modeling results on this artificial data with that on the original data allows to assess the performance of our pipeline in the case of missing data.

The basic idea is to manually perturb point clouds of real scenes by successively adding noise or removing portions of the point cloud. These modified data sets then create separate test sets, are modeled by means of our pipeline and the results are compared against the results for the original data without perturbations. Our comparison uses the intersection-over-union (IoU) of volumes (Jaccard Coefficient) defined by,

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{8.1}$$

where A and B are the volumes of the model with and without perturbations (Figure 8.10 red and green model, respectively). A perfect match gives a Jaccard coefficient of 1 and the case of no match results in a value of 0. I.e., the higher the Jaccard coefficent, the better the match, signifying the robustness of the modeling pipeline to incomplete data. Our test data set stems from images derived by SfM/MVS and contains 20 million 3D points. The data shows a scene representing a gable roof building and its immediate surroundings. First, we model the scene with our pipeline and compute the textured model from the resulting polygons. The visual inspection of the textured model shows appealing results, therefore, in the absence of "real" ground-truth, we consider this as the "pseudo ground-truth".

8.5.1 Effects of Incomplete Data

To evaluate our template based model fitting in the very likely case of incomplete data, e.g., due to occlusion, we manually segment three different combinations of the facades from the test data set. The modified test data is shown in Figure 8.11. First, we model the scene without perturbation, then we model all modified versions of the test data and finally we compute the Jaccard coefficient defined by Equation (8.1) where A and B are the volumes of the models with and without perturbation, respectively. We achieve an average Jaccard coefficient of 0.93 for this gable roof building, showing a good robustness against incomplete data, hence, substantiating our pipeline.

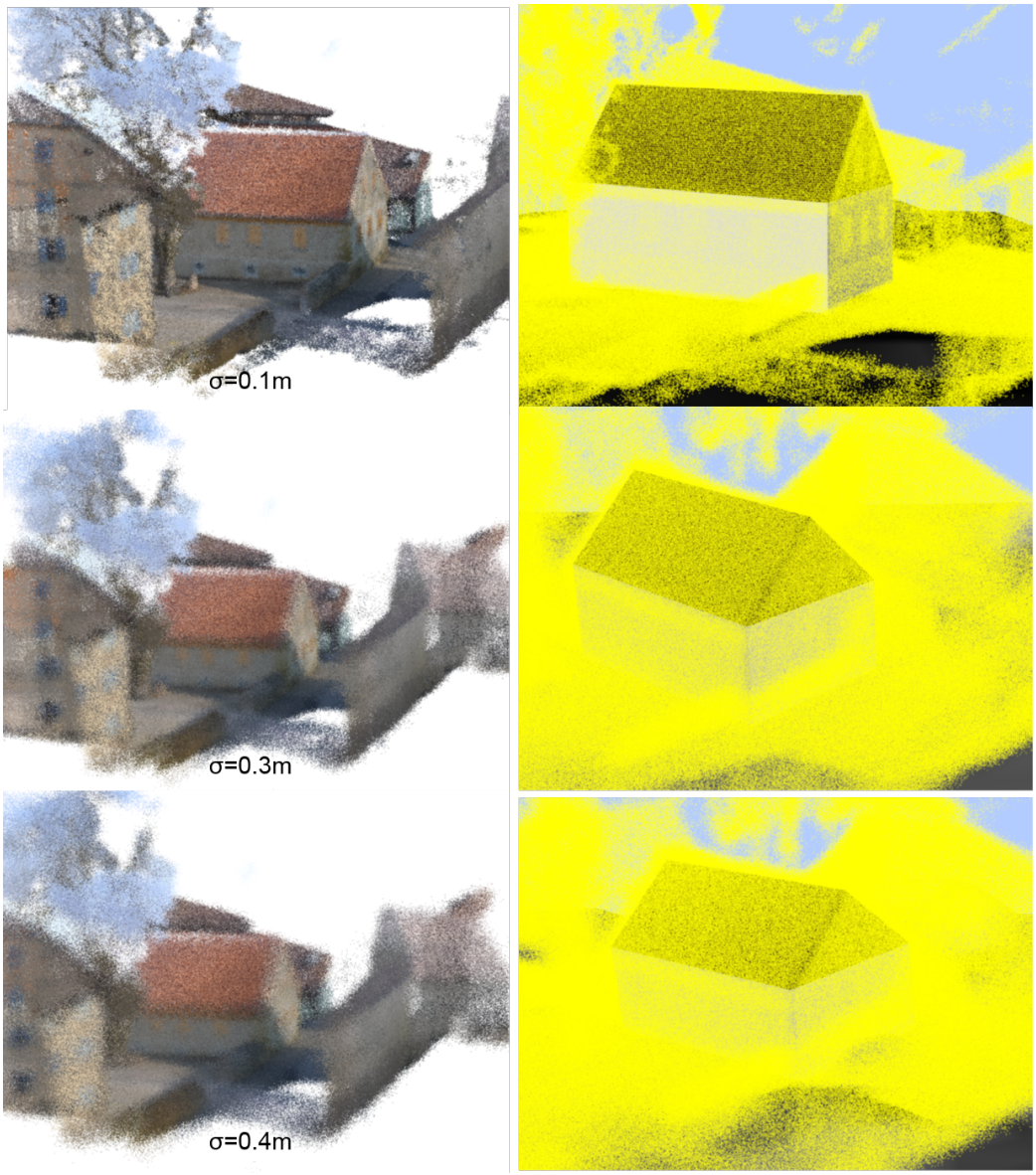

Figure 8.12: Data modified by additive Gaussian noise (left column) with increasing standard deviation σ pose a challenge for our modeling pipeline. Nevertheless, our pipeline could successfully model the simple gable roof building (right column) showing robust behavior for substantial noise.

8.5.2 Effects of Increasing Noise

Point cloud acquisition algorithms from images via SfM/MVS produce very appealing results. Nevertheless, achieving results with low noise still remains a huge challenge. We, thus, study the effects of noise on our modeling from 3D point clouds pipeline similarly to the case of incomplete data described above. To this end, test data sets are created

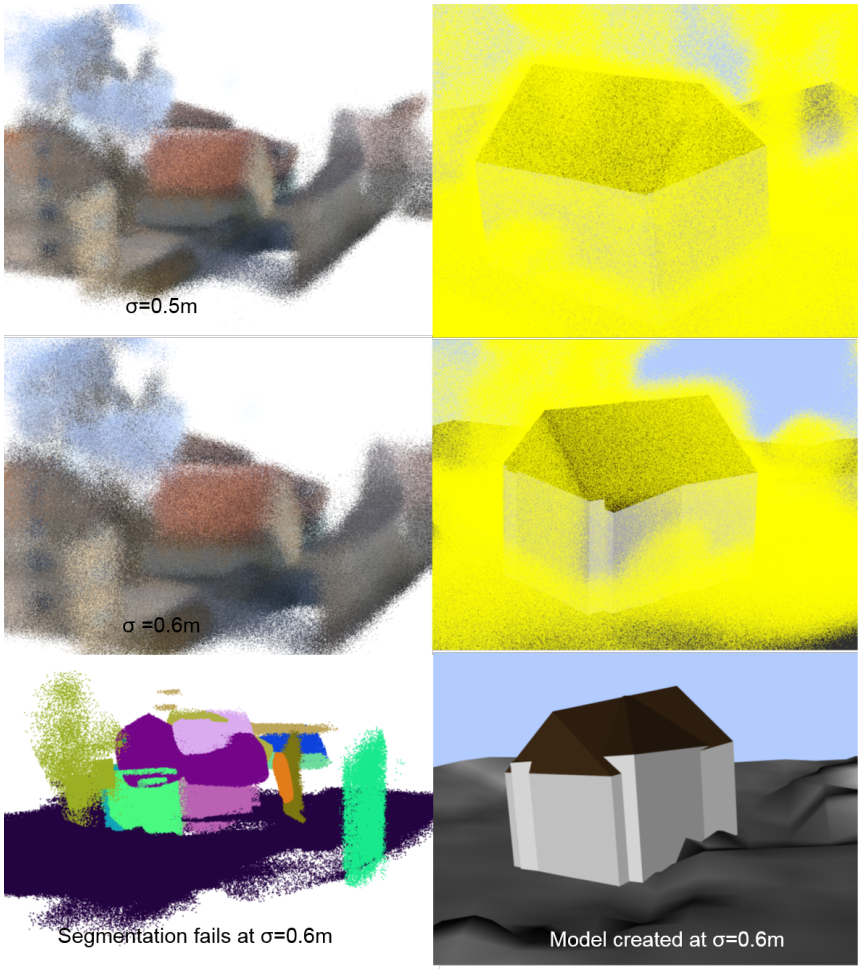

Figure 8.13: Starting at about $\sigma = 0.6$, the segmentation completely fails and produces many more patches than present in the scene (all segmented patches are randomly colored). This leads to the fitting of inconsistent polygons.

by modifying the point cloud depicted in Figure 8.10 by systematically changing the 3D points. In Figure 8.12 (left column), the effects of adding zero mean Gaussian noise to the points in the test data, i.e., $\hat{\boldsymbol{x}}_i \sim \mathcal{N}(\boldsymbol{x}_i, \sigma)$, and the results (right column) produced by our pipeline on the modified data is shown. Different noise levels are chosen by increasing the standard deviation σ. Beyond $\sigma = 0.5m$, our modeling pipeline breaks, because the segmentation fails (Figure 8.13).

8.5.3 Effects of Vegetation

In Chapter 6, we assigned four distinct labels to the output of our segmentation algorithm. Amongst the four class labels, we presumed that segments associated with the class label "rest" are predominantly of the class vegetation and will not be considered

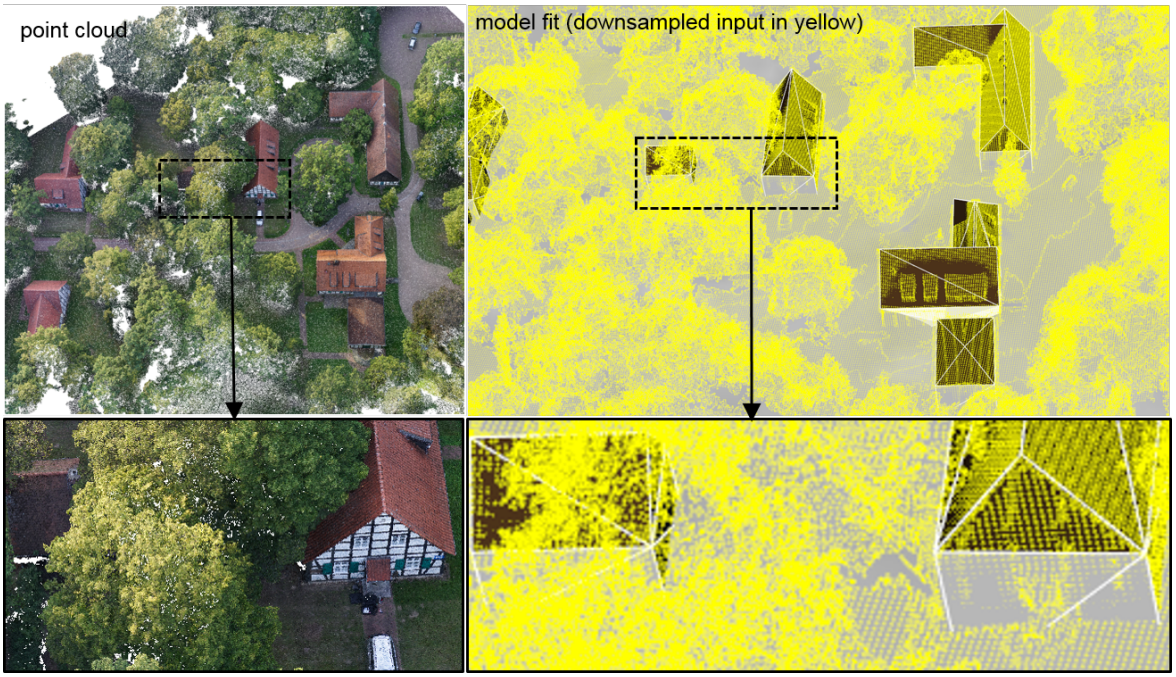

Figure 8.14: Substantial vegetation is often a challenge for data processing for urban scenes. Yet, even in this example dataset with plenty of vegetation, our pipeline is able to fit the data robustly and correctly (right). This is largely due to the robustness of RANSAC combined with probabilistic clustering.

for further processing within this thesis. Therefore, vegetation data is considered non-systematic noise and is not fitted to planar surfaces. This is mainly because neighboring CPNs of voxels with substantial vegetation will show random orientations, and hence, cannot be within the same cluster (cf. Chapter 5.3). Examples showing the robustness of our modeling pipeline to substantial amount of vegetation is presented in Figures 8.14 and 8.15. In both scenes, the point clouds where acquired from images and derived by SfM/MVS.

8.6 Scalability

Our segmentation algorithm uses RANSAC which itself is fast and robust. However, large-scale patch segmentation in point clouds is still a computationally expensive task. Scalability is, thus, an important aspect in our algorithmic layout also because the recent advances in 3D acquisition algorithms, e.g., from SfM/MVS, are leading to data of increasing quality as well as quantity. Our algorithm is inherently scalable since, we divide the point cloud into non-overlapping voxels and search for planes within every voxel independently. Additionally, scene voxelization is a form of data compression, because it groups and represents many 3D points (within a voxel) by a single CPN.

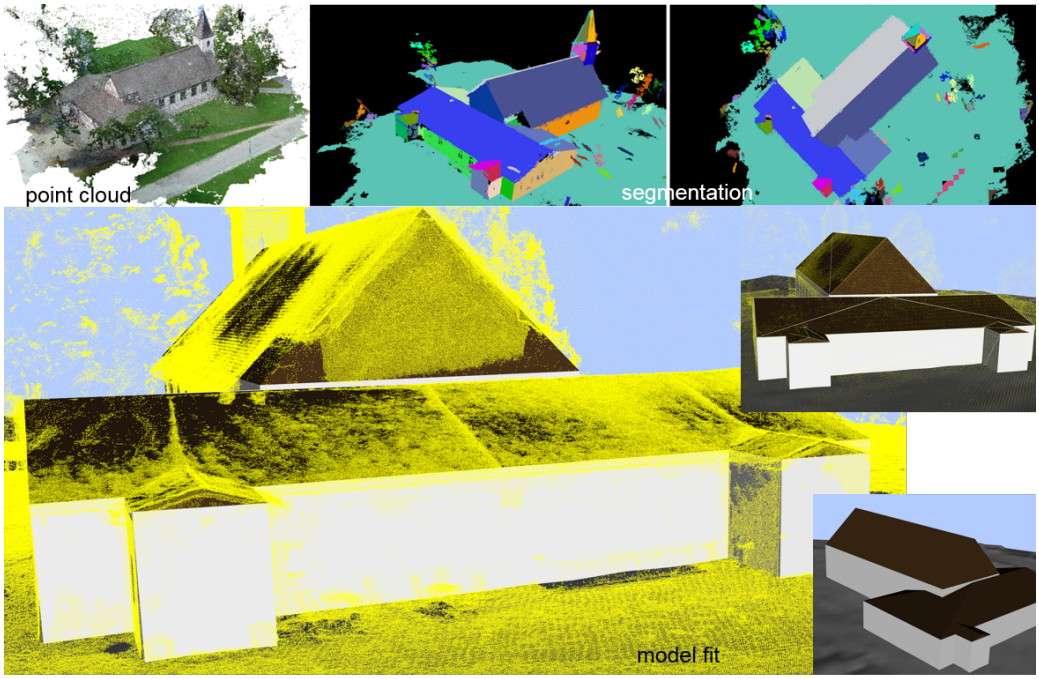

Figure 8.15: Robustness with respect to vegetation is a desirable characteristic for algorithms working on real world data sets of urban scenes. In this example, downsampled input data in yellow are superimposed on the model to show the quality of the fit of the derived polygons for buildings and the ground. Segmented patches are randomly colored.

The application of RANSAC within such a local support is quite efficient because of the strong local planarity within the voxels as urban scenes largely consist of planar surfaces. Therefore, the best plane is often found in a few iterations, because only points within the voxel are considered for the search. Furthermore, the number of CPNs is many magnitudes lower than the number of 3D points. These constraints make it possible to process huge 3D point cloud data such as of the scene depicted in Figure 8.16 with 2.2 billion 3D points acquired via terrestrial LiDAR scans in a limited amount of time. The overall run-time of our algorithm using the hardware described above on this data set is 3.5 hours. A major speedup could be achieved by parallel processing. This is currently outside of the scope of this thesis and could be considered as a possible extension.

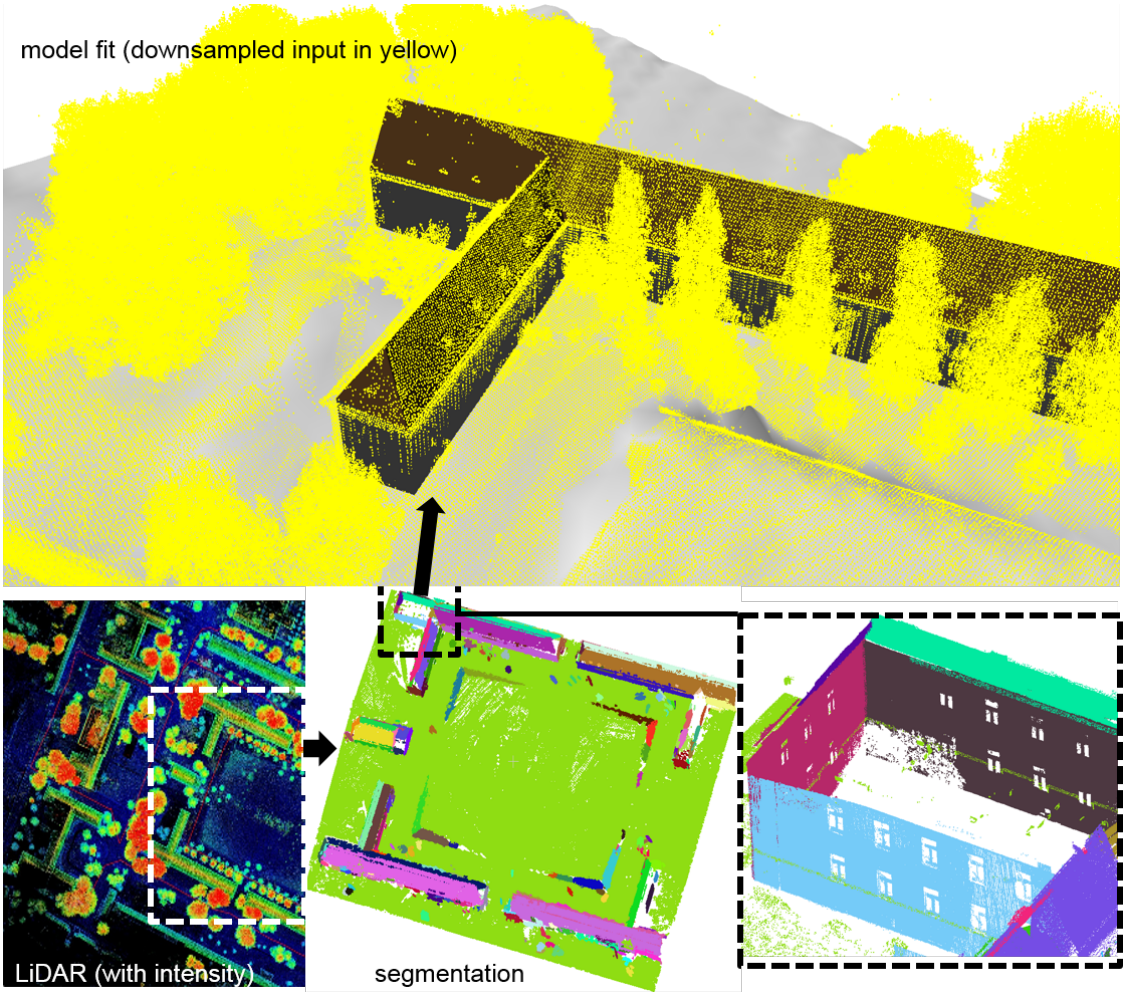

Figure 8.16: Scalability is an important aspect of our algorithmic design. In this example, we demonstrate the performance of our algorithm on a very large data set containing 2.2 billion 3D points acquired by terrestrial LiDAR scans. A major challenge has been to enable interactive visualization at this scale with our described hardware.

8.7 Qualitative Results

Besides for the already presented examples, we also tested our modeling pipeline on many self-acquired data sets. In all experiments, we used the default parameters as described above, without any fine tuning. Visual inspection shows that the results are very appealing and substantiate the appropriateness of our approach. This can be seen in Figures 8.17 to 8.22.

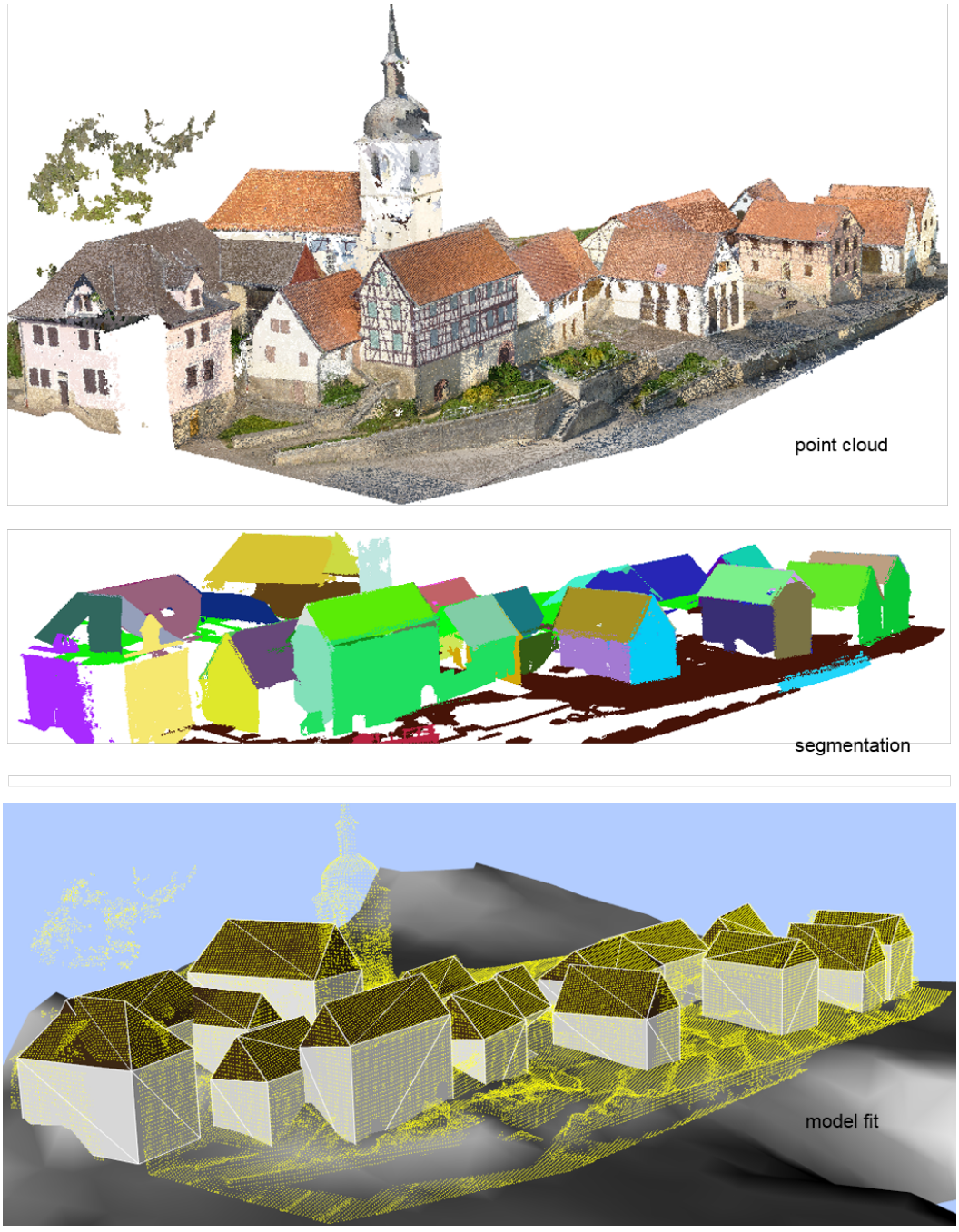

Figure 8.17: The input data contains 52 million noisy 3D points from SfM/MVS of a residential area with nonplanar terrain. Our pipeline segments the scene into patches and models all buildings besides the church steeple and the nonplanar ground correctly.

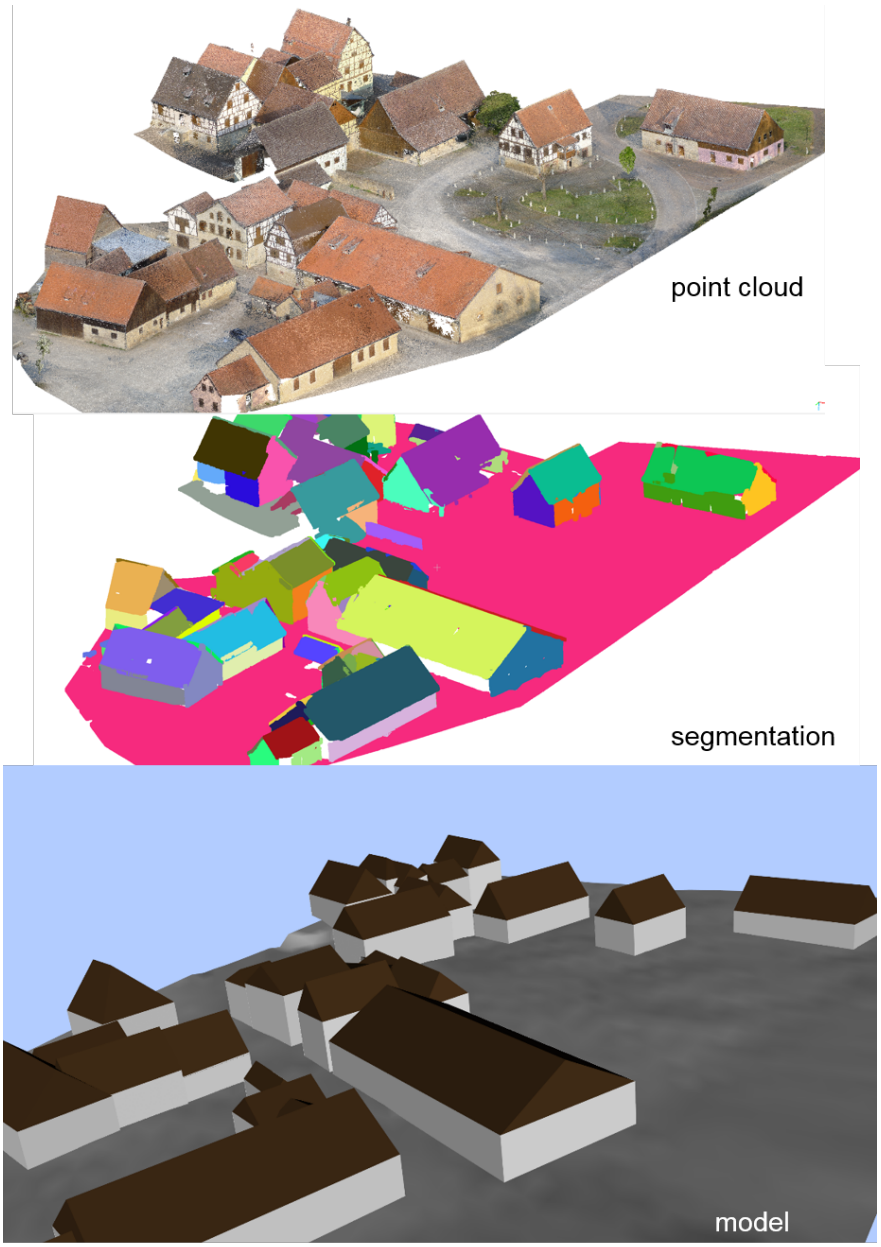

Figure 8.18: The input data consist of 67 million noisy 3D points from SfM/MVS of a residential area with nearly planar terrain. Our pipeline segments the scene into patches. The ground is the patch with the largest convex hull area (red).

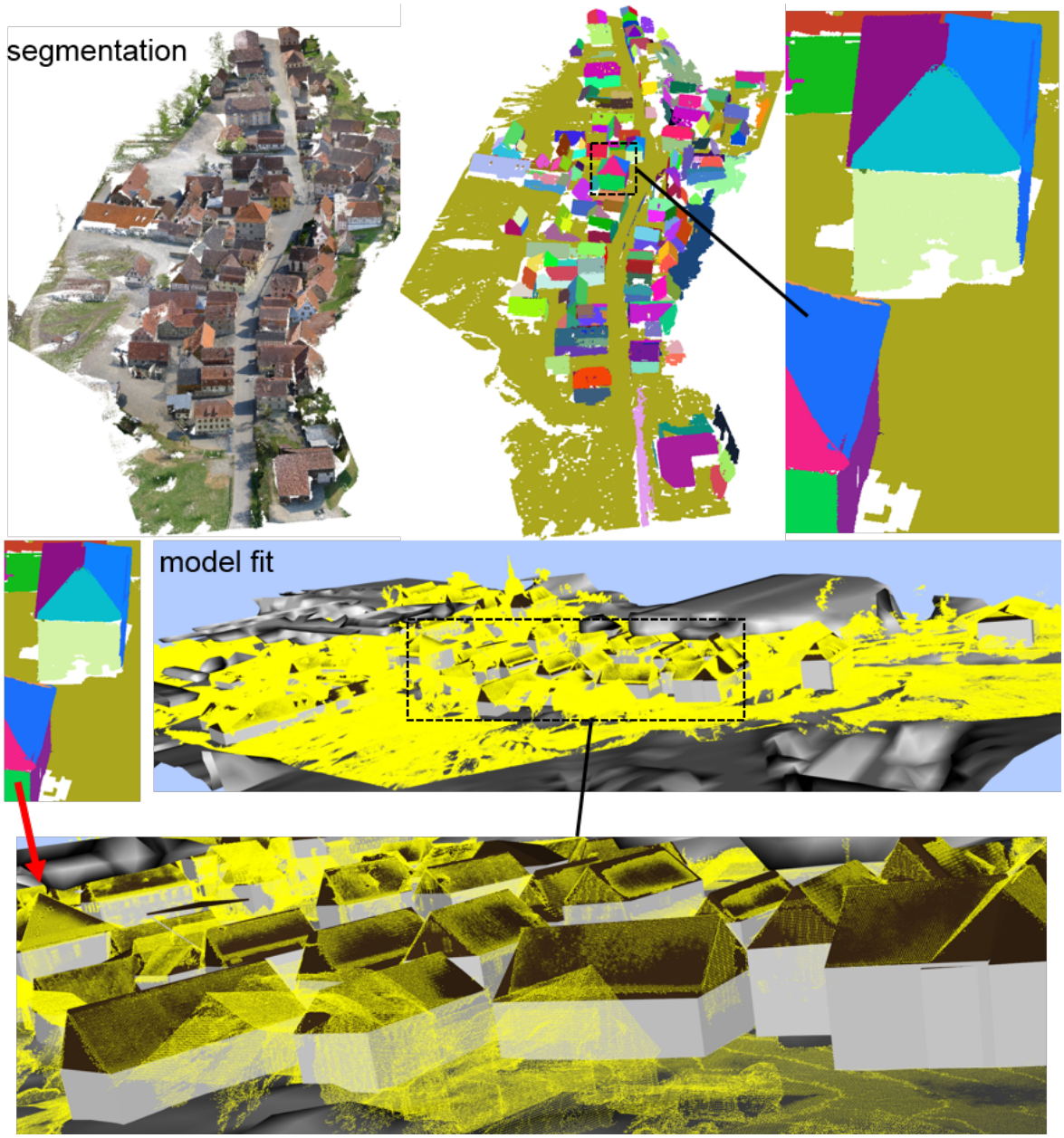

Figure 8.19: Dataset "Bonnland". The input are 107 million noisy 3D points derived from images by SfM/MVS. For the residential area, 51 buildings with quadrilateral footprints have been fitted. The runtime is 2.5 hours with the current code (not optimized). All colors for the patches after segmentation are randomly chosen.

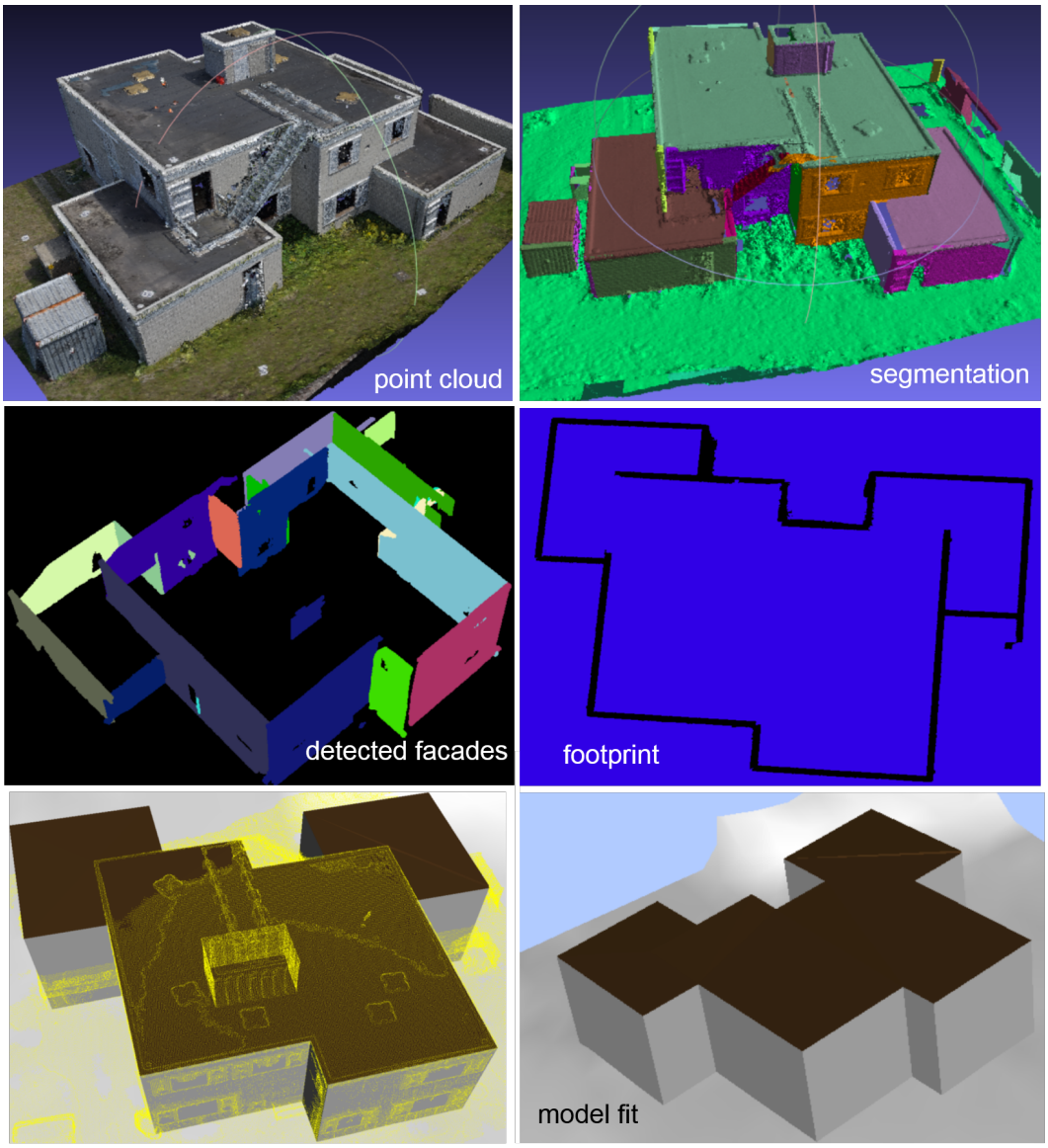

Figure 8.20: Buildings with complex footprint are usually challenging. In this example, the data contains 47 million noisy 3D points from SfM/MVS. A major challenge is to detect the footprint for which we applied the algorithm presented in Section 6.3.2. Together with the detected flat roofs this leads to the final LOD 2 model for which equal facade height is assumed. This assumption has to be removed in future work (cf. Section 10.2).

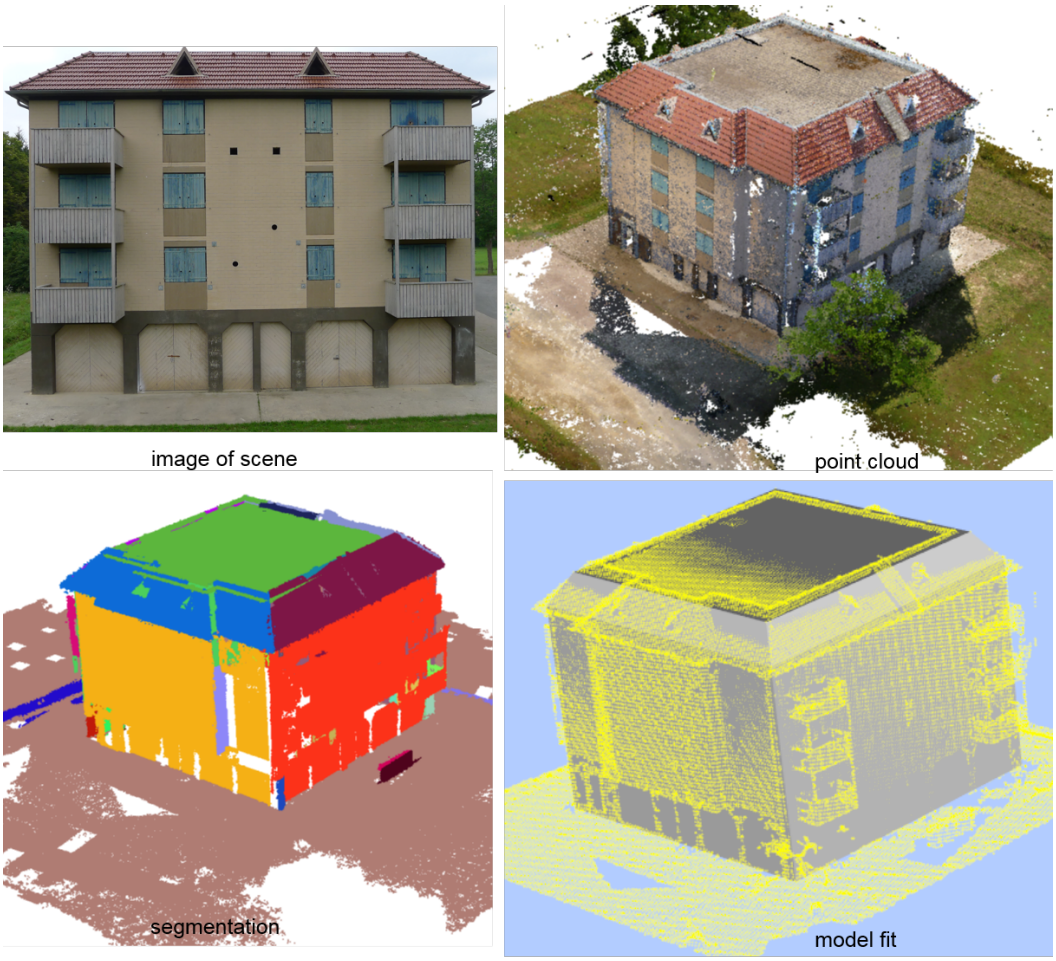

Figure 8.21: Dataset "haus51": The input are 42 million noisy 3D points derived from images by SfM/MVS. The images are captured at multiple UAV flights and with a hand-held camera. The runtime is 20 minutes with the current code (not optimized). All colors for the patches after segmentation are randomly chosen. A mansard roof model "best" explains the data. This can be seen via the model fit combined with a downsampled version of the input points in yellow. Our LOD 2 models does not capture roof and facade superstructures. These are possible areas for future work (cf. Section 10.2).

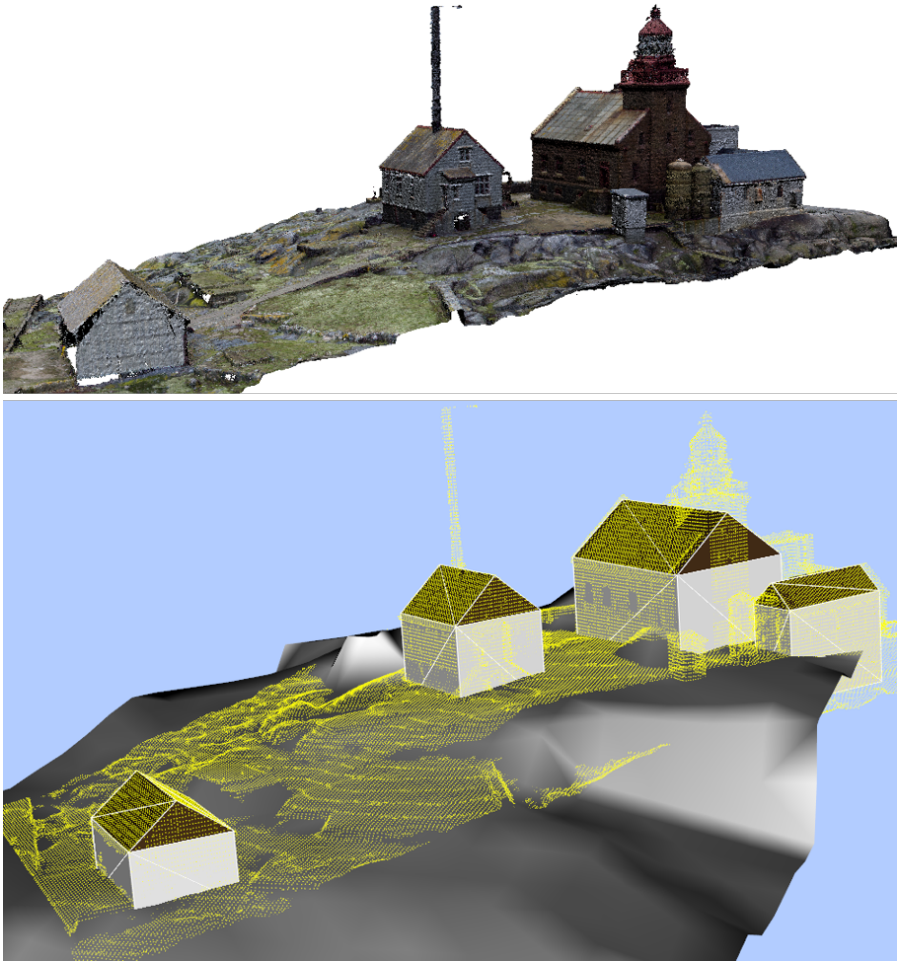

Figure 8.22: Modeling buildings on hilly terrain is usually challenging. In this example, the dataset contains 17 million noisy 3D points from SfM/MVS representing four buildings. Besides noise and reconstruction artifacts, this dataset has three major challenges for modeling from point clouds: Highly nonplanar terrain, high point density variations (the stony ground is richer in texture than the building facades and the roofs, and therefore, more 3D points are on the ground than on the buildings) and the roof superstructures. While the first two challenges where correctly tackled by our pipeline as can be seen from the results above, the modeling of roof-superstructure is possible future work (cf. Section 10.2).

Chapter 9

Strengths and Weaknesses

In the preceding chapters, we have presented a fully automatic pipeline to process point clouds of urban scenes and generate watertight 3D CAD models of buildings as well as smooth surfaces for the ground. The proposed algorithms are suitable for point cloud data derived from images by SfM/MVS as well as from LiDAR. Amongst others, modeling urban scenes is motivated by applications from robotics, computer vision, remote sensing as well as computer graphics and the video game industry. In robotics for example, an autonomous agent such as a self-driving car, performing everyday tasks like moving around a geofenced environment and transporting clients, must know the objects in its environment: the buildings, the ground, the streets, etc. Detailed knowledge about these objects includes information about their geometry as well as semantics and structural layout, such as "this facade is connected to these other facades, this street is drivable, this is vegetation". For example, in video games, precise knowledge about the environment enables to spawn new agents on flat or mansard roof buildings rather than gable roof buildings. With the proposed approach, the creation of 3D models of urban scenes can be automated. To our best knowledge, this has so far not been possible for objects in point cloud data of real world scenes with substantial noise, or has been done manually by a professional 3D content creation artist. Towards this goal, several extensions of existing state-of-the-art methods have been presented which are summarized below.

9.1 Contributions

In particular, our algorithms provide a robust and scalable pipeline for modeling urban scenes from point clouds, can handle point cloud data with substantial noise, as well as point density variation, and operate on point clouds acquired both from images by SfM/MSV and by LiDAR techniques. Their strength is based on our large-scale segmentation (Nguatem and Mayer 2016) algorithm, that decomposes the scene into meaningful units, e.g., roof elements, facades and the ground. The segmentation algorithm itself relies on the robustness of RANSAC and probabilistic clustering providing consistent patch segmentation. This is followed by semantic scene interpretation using simple architectural rules and finally template-based shape fitting and model selection which provide robustness against incomplete data and occlusions. A salient feature of the pipeline is that it has very few tuning parameters and operates fully automatically. We summarize the major contributions of our algorithms as follows:

1. A robust, scalable probabilistic point cloud segmentation algorithm which uses a nonparametric Bayesian framework for clustering and automatically infers, K, the number of segments present in the scene.

2. A consistent set of basic architectural rules that enables semantic decomposition of urban scenes into the four meaningful categories, the ground, roof element, facade, and the rest, without making an assumption about a Manhattan-World or structural symmetry in the building layout.

3. Polygon-sweeping detects facades in noisy point clouds as a substitute for the more general plane-sweeping algorithm usually employed.

4. A likelihood function based on MSAC is introduced that scores polygonal chains representing a complete model of a building and, thus, ensures model completeness even in the case of missing data. Model selection is employed to choose consistently between competing models.

5. NURBS surfaces are used as a natural representation for the modeling of the ground, as apposed to the commonly used planes. This retains the inherent natural smoothness.

6. Boolean algebra is employed to clip redundant geometry of facades underneath the ground, hence enabling a more appealing representation.

9.2 Limitations and Possible Extensions

While we have described an automatic pipeline for modeling buildings from 3D point clouds of urban scenes giving very appealing results, there are two major limitations of our approach in solving the given task as discussed below.

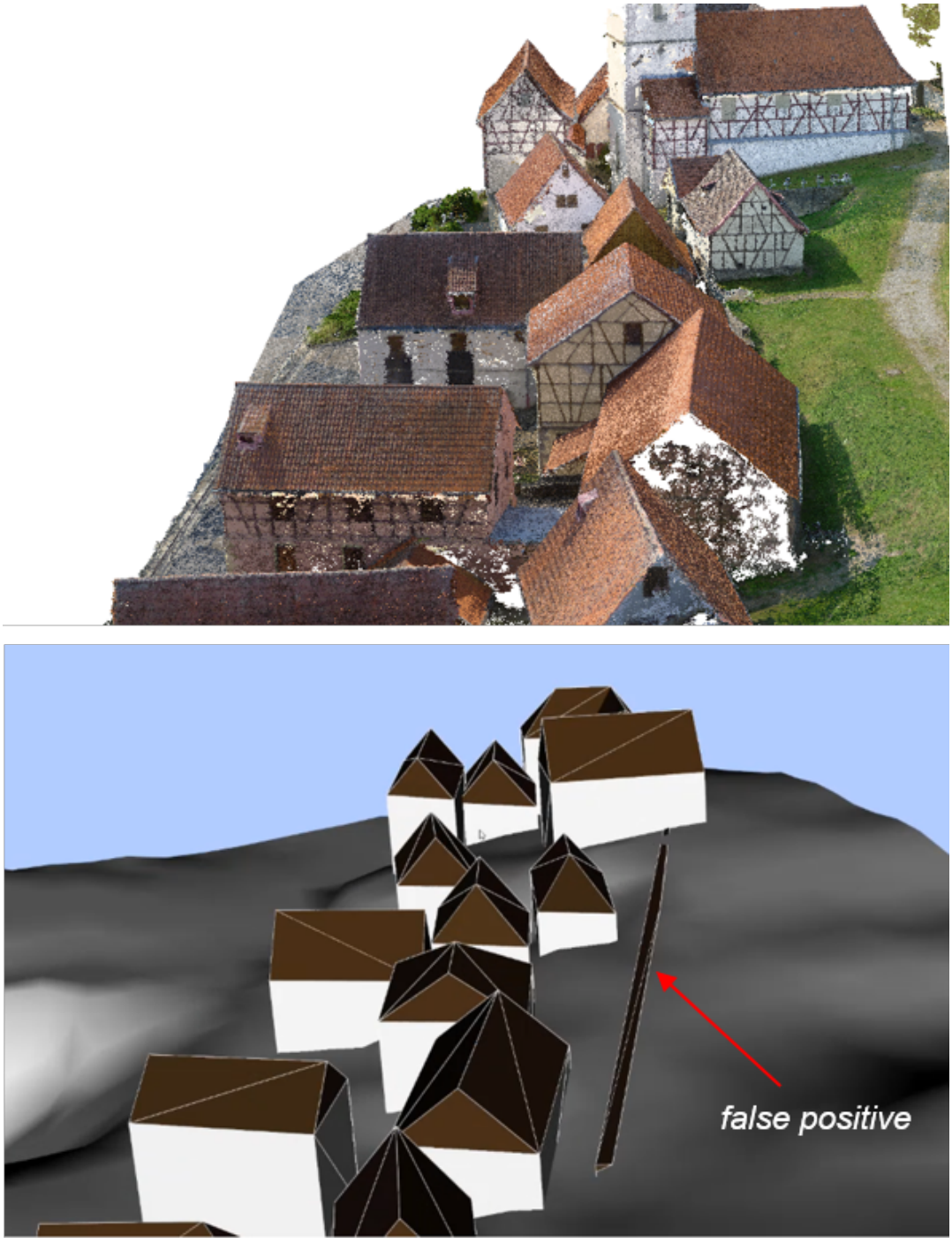

Figure 9.1: In this example, our pipeline correctly models all buildings and the highly non-planar ground. However, an adverse configuration of the ground was falsely detected as a Geometric Shape Primitive (GSP) for buildings and a template was fitted as well. This example shows the need for postprocessing the results of the described pipeline.

9.2.1 Necessity of Postprocessing

In Chapter 6, labels are assigned to patches derived from the scene based on simple geometric and architectural rules. However, certain configurations of the acquired data may conform to these basic rules and match the predefined templates for buildings, even though they are no buildings. For example, in the presence of plenty of vegetation or very hilly terrains, certain configurations of patches may lead to false detections of roofs or facades. These can often be corrected during geometric shape fitting with model selection. However, shape fitting itself just fits the data without any consistency check, hence can sometimes fit a false detection (cf. Figure 9.1). To remedy this situation, postprocessing is inevitable. This could employ a consistency check based on reference values, e.g., minimum distance from the ground to the roof, minimum ratio of the enclosed volume of a buildings to the surface area of the roof, etc. Furthermore, the acquired data is hardly complete and missing data for ground modeling often leads to holes as can be seen in Figure 9.2. They also have to be filled by postprocessing.

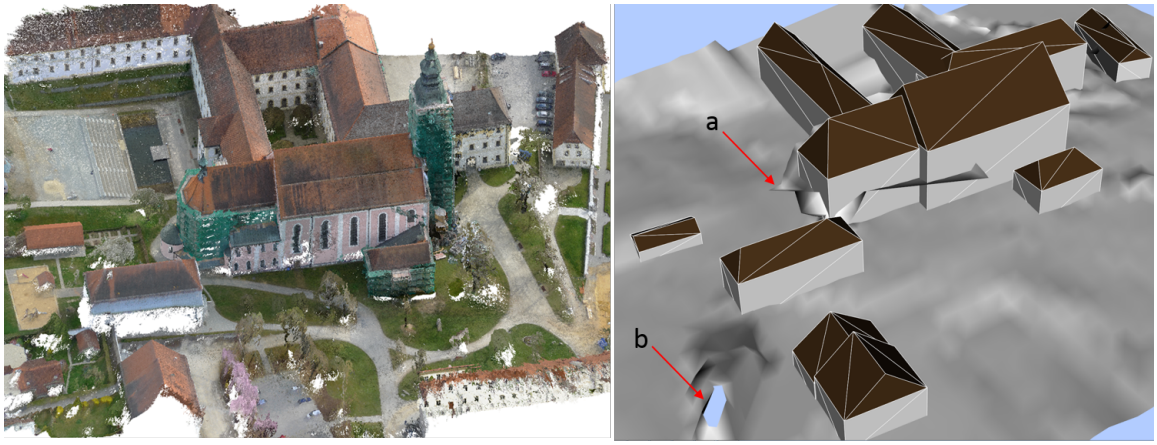

Figure 9.2: Missing data result in holes (b) and reconstruction artefacts with unnecessary or wrong geometry during ground modeling (a). Amongst others, these two can be resolved during postprocessing.

9.2.2 Strong Local Planarity

While providing scalable and robust segmentation of locally planar patches, our pipeline is not suitable for modeling buildings with arbitrary shape. Furthermore, the use of predefined templates assumes that the shapes of buildings strongly conform with these templates, which is often not the case for non-residential buildings such as churches (cf. Figure 9.3). A possible extension to our pipeline could be to first try to explain large parts of the point cloud with simple templates by fitting the presented GSPs for buildings and then extending this set to include more complex 3D geometric shapes such as spheres, cylinders and cones. Finally, the remaining parts of the point cloud data would be modeled using triangular meshes. The final result would then consist of the

integration of partial models into a hierarchically structured hybrid model which can approximate arbitrary shapes.

Figure 9.3: Due to the local planarity assumption, it is difficult to correctly model buildings with surfaces exhibiting a strong local curvature. A possible way to remedy this problem is to include free-form surfaces not only for the ground but for the buildings as well. This however, will increase model complexity.

Chapter 10

Conclusion

10.1 Summary

We have proposed a fully automatic pipeline for modeling buildings in an urban environment from 3D point clouds. Our workflow relies on probability theory and scales to datasets with hundreds of millions of noisy 3D points derived from images or LiDAR scans. It is based on a robust and accurate large-scale segmentation using a combination of RANSAC and nonparametric Bayesian clustering, a set of basic rules to enable scene semantic interpretation, polygon-sweeping as well as geometric shape fitting with model selection, and finally NURBS surface fitting for modeling both man-made (buildings) and natural (the ground) surfaces. Besides robustness against substantial noise, efficiency and scalability, our approach offers several outstanding properties in comparison to existing approaches, e.g., standard meshing algorithms. First, it abstracts the scene into a compact but accurate representation while maintaining semantics which allows for an easy integration into different applications. Second, it retains the natural smoothness inherent in large parts of the ground by modeling with NURBS. And finally, it operates fully automatically and has very few parameters to tune enabling a simple usage.

10.2 Future Work

In the previous chapter, we have presented the limitations and possible extensions of our proposed pipeline for modeling buildings from 3D point clouds. In addition to these extensions, further possibly fruitful lines of future work along the direction of this research are described below.

10.2.1 Variable Voxel Size

We have proposed an approach for the segmentation of 3D point clouds based on a divide-and-conquer algorithm that divides the scene into voxels of a fixed size. This makes a final smoothing routine necessary which removes effects from the fixed-size scene voxelization. Fixed-size scene voxelization is also not suitable for hybrid model representations for buildings, particularly 3D models as described in this thesis combined with a triangle mesh representation. In order to overcome this deficiency, a more flexible voxelization should be applied, i.e., with variable voxel size. One possible approach for automatically determining a suitable voxel size could be, to start with a fixed voxel size and merge or divide neighboring voxels based on their curvature. A suitable metric for

quantifying curvature at a particular point is the ratio of the sum of the eigenvalues to the smallest eigenvalue using all points within the voxel under consideration as the influencing neighborhood.

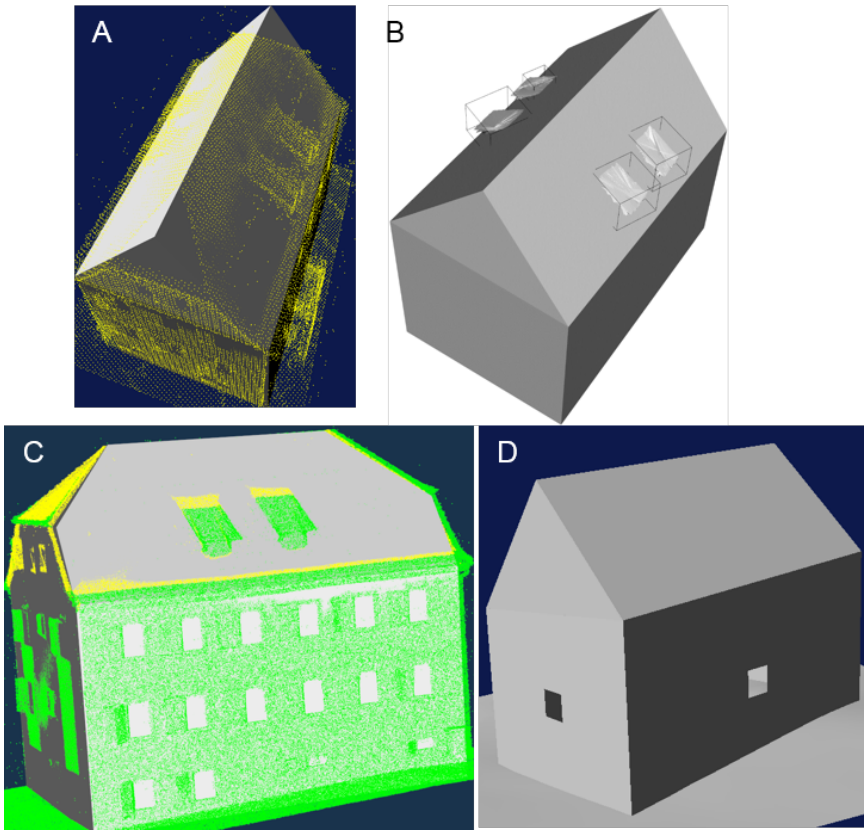

Figure 10.1: Currently, our modeling pipeline processes 3D point clouds and produces LOD 2 type models (A). To improve our 3D model representations to LOD 3 or higher in the future, it would be desirable to detect roof superstructures (B), refine roof models (C) as well as add balconies if present. Furthermore, windows and doors (D) should be detected and integrated in the models.

10.2.2 Increase of the Level of Detail – LOD

Our proposed approach is meant to produce LOD 2 models for buildings with local planar surfaces. In addition, we have modeled the non-planar ground with a NURBS surface to capture its inherent natural smoothness. Yet, many applications require an increased accuracy and models with more detail. For example, in simulations, e.g., for a personal robotic assistant to perform complex manipulation tasks in the environment such as entering buildings via opening the doors based on their door handle position, a higher level of detail (LOD 4) is needed. A very first step in this direction is to detect the windows and doors present on the facades (see Figure 10.1). They could be localized on

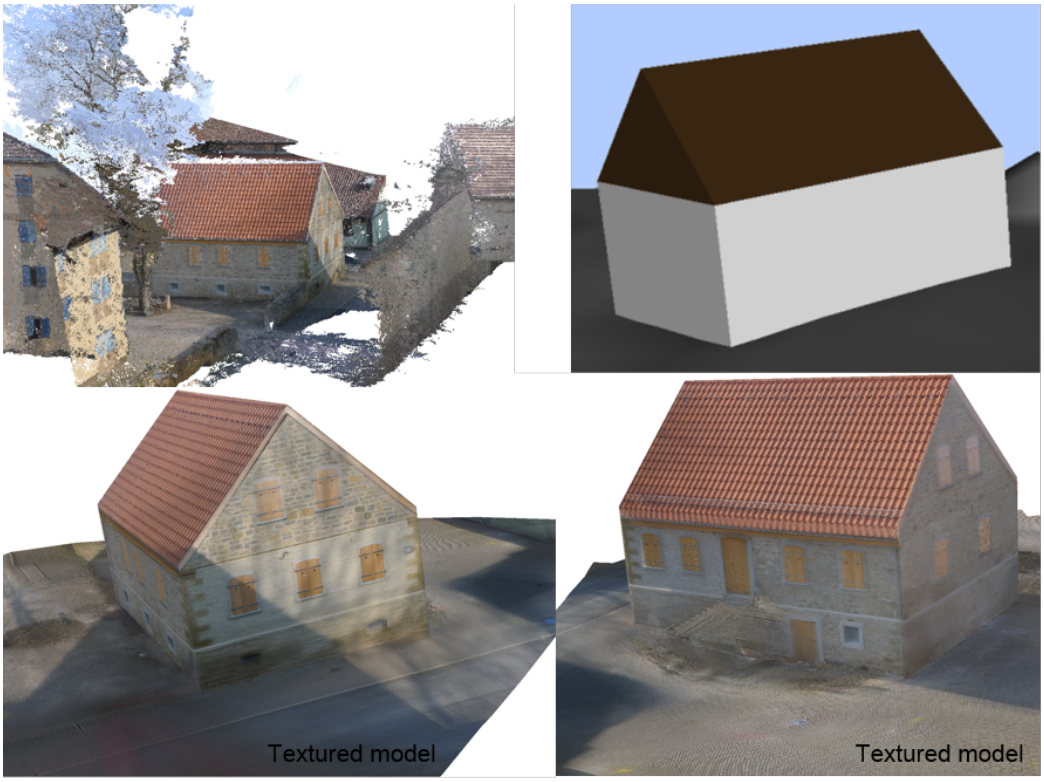

Figure 10.2: Mapping texture to 3D models improves the visual appearance.

the LOD 2 planar facades, e.g., using the algorithms described in (Nguatem et al. 2014). Integrating detected windows and doors to the results of our proposed pipeline leads to LOD 3 models. So far, vegetation was considered as noise or scene clutter within our pipeline. Including it will increase the model fidelity. With the current trend in machine learning for object classification, the modeling can be further refined by classifying the images, segmenting the areas for vegetation and finally modeling it in 3D.

10.2.3 Models with Texture Mapping

While we have demonstrated how 3D geometric models of buildings in urban environments including the semantics can be derived from 3D point clouds, this thesis has not addressed the fusion of geometry and texture information. Texture is often important and sometimes it is even necessary for particular applications for which geometry alone is not enough. For example, the shapes of two facades could be identical, yet one is blue and has three windows while the other is red and has four glass windows without frames. In the previous chapters, we have chosen three major colors and rendered the polygon chains constituting the resulting CAD/watertight model in accordance with the labeled segments. Yet, for point clouds derived by SfM/MVS from images, a more appealing result with a higher fidelity can be achieved by adding textures to the models.

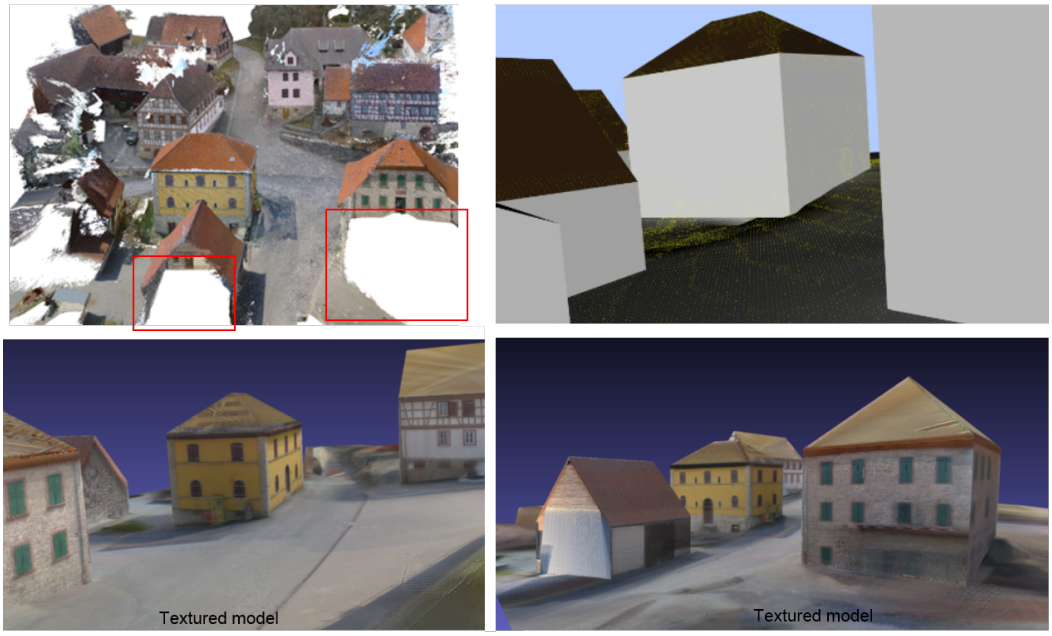

Figure 10.3: Texturing models leads to an appealing visual result by combining corresponding pixel values from different images (blending). In this examples, modeling errors may arise due to missing data from the two facades. Texture blending however, can hide these errors.

Figure 10.2 depicts a simple attempt to map textures on a 3D model. However, mapping textures on 3D models of outdoor scenes at this scale is itself a non-trivial task. This is due to occlusions, radiometric differences within the image set from which the point cloud originates, etc. For mapping textures on the scene shown in Figure 7.1, images are sub-sampled to 1/4 resolution and radiometric differences are not corrected to speed up the computation. Though not optimal, the results are appealing and are clearly better than the models shown in Figure 7.1. An additional advantage of texture mapping is that errors generated by our pipeline can be hidden by blending texture (combining corresponding pixel values from different images) in a post-processing stage (cf. Figure 10.3).

Abbreviations for frequently cited conferences and journals

3DV : International Conference on 3D Vision (formerly: 3DIMPVT)
AISTATS: International Conference on Artificial Intelligence and Statistics
BMVC : British Machine Vision Conference
CVIU : Computer Vision and Image Understanding
CVPR : IEEE Conference on Computer Vision and Pattern Recognition
ECCV : European Conference on Computer Vision
EUROGRAPHICS : Conference of the European Association for Computer Graphics
GCPR : German Conference on Pattern Recognition (formerly: DAGM)
ICCV : IEEE International Conference on Computer Vision
ICRA : International Conference on Robotics and Automation
IJCV : International Journal of Computer Vision
ISPRS : International Society of Photogrammetry and Remote Sensing
JISPRS : ISPRS Journal of Photogrammetry and Remote Sensing
LNCS : Lecture Notes in Computer Science
PAMI : IEEE Transactions on Pattern Analysis and Machine Intelligence
PFG : Photogrammetrie - Fernerkundung - Geoinformation
SIGGRAPH : International Conference and Exhibition on Computer Graphics and Interactive Techniques

Bibliography

Airoldi, E. M., Blei, D., Erosheva, E. A. and Fienberg, S. E. (2014): *Handbook of Mixed Membership Models and Their Applications*, Chapman & Hall/CRC.

Aldous, D. J. (1985): *École d'Été de Probabilités de Saint-Flour XIII — 1983*, Springer, Berlin, Heidelberg, chapter Exchangeability and related topics, 1–198.

Anderson, T. (2003): *An Introduction to Multivariate Statistical Analysis*, Wiley Series in Probability and Statistics, Wiley.

Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M. and Savarese, S. (2016): 3D semantic parsing of large-scale indoor spaces, *CVPR*, 1534–1543.

Arvo, J. (1995): Stratified sampling of spherical triangles, *SIGGRAPH*, 437–438.

Basu, S., Davidson, I. and Wagstaff, K. (2008): *Constrained clustering: Advances in algorithms, theory, and applications*, Chapman & Hall/CRC.

Boulch, A., Saux, B. L. and Audebert, N. (2017): Unstructured point cloud semantic labeling using deep segmentation networks, *EUROGRAPHICS Workshop on 3D Object Retrieval*, The Eurographics Association, 17–24.

Brodu, N. and Lague, D. (2012): 3D terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology, *JISPRS* **68**: 121 – 134.

Charles, R. Q., Su, H., Kaichun, M. and Guibas, L. J. (2017): Pointnet: Deep learning on point sets for 3D classification and segmentation, *CVPR*, 77–85.

Chauve, A. L., Labatut, P. and Pons, J. P. (2010): Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data, *CVPR*, 1261–1268.

Chum, O. and Matas, J. (2005): Matching with PROSAC – Progressive sample consensus, *CVPR*, Volume 1, 220–226.

Collins, R. T. (1996): A space-sweep approach to true multi-image matching, *CVPR*, 358–363.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. and Schiele, B. (2016): The cityscapes dataset for semantic urban scene understanding, *CVPR*, 3213–3223.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T. and Niessner, M. (2017): Scannet: Richly-annotated 3D reconstructions of indoor scenes, *CVPR*, 2432–2443.

Demir, I., Aliaga, D. G. and Benes, B. (2015): Procedural editing of 3D building point clouds, *ICCV*, 2147–2155.

Dick, A., Torr, P. and Cipolla, R. (2004): Modelling and interpretation of architecture from several images, *IJCV* **60**(2): 111–134.

Dimitrov, A., Gu, R. and Golparvar-Fard, M. (2016): Non-uniform B-spline surface fitting from unordered 3D point clouds for as-built modeling, *Computer-Aided Civil and Infrastructure Engineering* **31**(7): 483–498.

Ericson, C. (2004): *Real-Time Collision Detection*, Interactive 3D Technology, Morgan Kaufmann Series in Interactive 3D Technology.

Esteban, C. H. and Schmitt, F. (2004): Silhouette and stereo fusion for 3D object modeling, *CVIU* **96**(3): 367–392.

Evans, M., Hastings, N. and Peacock, B. (2000): von Mises distribution, *Ch. 41 in Statistical Distributions* 189–191.

Faugeras, O. (1993): *Three-Dimensional Computer Vision: A Geometric Viewpoint*, Artificial Intelligence, MIT Press.

Ferguson, T. S. (1973): A Bayesian analysis of some nonparametric problems, *Ann. Statist.* **1**(2): 209–230.

Fischler, M. A. and Bolles, R. C. (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.

Fisher, R. (1953): Dispersion on a sphere, *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* **217**(1130): 295–305.

Fossati, A., Gall, J., Grabner, H., Ren, X. and Konolige, K. (2012): *Consumer depth cameras for computer vision - Research topics and applications*, Advances in Computer Vision and Pattern Recognition, Springer-Verlag.

Frahm, J.-M., Fite-Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.-H., Dunn, E., Clipp, B., Lazebnik, S. and Pollefeys, M. (2010): Building Rome on a Cloudless Day, *ECCV*, 368–381.

Furukawa, Y. and Ponce, J. (2010): Accurate, dense, and robust multi-view stereopsis, *PAMI* **32**(8): 1362–1376.

Gallup, D., Frahm, J.-M., Mordohai, P., Yang, Q. and Pollefeys, M. (2007): Real-time plane-sweeping stereo with multiple sweeping directions, *CVPR*, 1–8.

Geiger, A., Lenz, P. and Urtasun, R. (2012): Are we ready for autonomous driving? The KITTI vision benchmark suite, *CVPR*, 3354–3361.

Goesele, M., Curless, B. and Seitz, S. (2006): Multi-view stereo revisited, *CVPR*, Volume 2, 2402–2409.

Görür, D. and Rasmussen, C. E. (2010): Dirichlet process Gaussian mixture models: Choice of the base distribution, *Journal of Computer Science and Technology* **25**(4): 653–664.

Gross, M. and Pfister, H. (2007): *Point-Based Graphics*, The Morgan Kaufmann Series in Computer Graphics, Morgan Kaufmann.

Guennebaud, G., Jacob, B. and others (2010): Eigen v3, http://eigen.tuxfamily.org.

Hastie, T., Tibshirani, R. and Friedman, J. (2009): *The elements of statistical learning: data mining, inference and prediction*, Springer.

Hennig, C., Meila, M., Murtagh, F. and Rocci, R. (2015): *Handbook of cluster analysis*, Chapman & Hall/CRC.

Hirschmüller, H. (2008): Stereo processing by semiglobal matching and mutual information, *PAMI* **30**(2): 328–341.

Hoppe, F. M. (1984): Pólya-like urns and the Ewens' sampling formula, *Journal of Mathematical Biology* **20**(1): 91–94.

Hou, F., Qin, H. and Qi, Y. (2016): Procedure-based component and architecture modeling from a single image, *Visual Computing* **32**(2): 151–166.

Huang, H., Brenner, C. and Sester, M. (2011): 3D Building Roof Reconstruction from Point Clouds via Generative Models, *19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 16–24.

Huang, H., Kuhn, A., Michelini, M., Schmitz, M. and Mayer, H. (2019): *3D urban scene reconstruction and interpretation from multisensor imagery*, Academic Press, 307–340.

Isack, H. and Boykov, Y. (2011): Energy-based geometric multi-model fitting, *IJCV* **97**(2): 123–147.

Ishwaran, H. and James, L. F. (2001): Gibbs sampling methods for stick-breaking priors, *Journal of the American Statistical Association* **96**: 161–173.

Kass, M., Witkin, A. and Terzopoulos, D. (1988): Snakes: Active contour models, *IJCV* **1**(4): 321–331.

Kazhdan, M., Bolitho, M. and Hoppe, H. (2006): Poisson surface reconstruction, *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, Eurographics Association, 61–70.

Kent, T. J. (1982): The Fisher-Bingham Distribution on the Sphere, *Journal of the Royal Statistical Society. Series B (Methodological)* **44**(1): 71–80.

Kolbe, T. H. and Bacharach, S. (2006): Citygml: An open standard for 3D city models, *Directions Magazine*.

Kolmogorov, V. and Zabih, R. (2001): Computing visual correspondence with occlusions using graph cuts, *ICCV*, Volume 2, 508–515.

Kuhn, A., Hirschmüller, H., Scharstein, D. and Mayer, H. (2017): A TV prior for high-quality scalable multi-view stereo reconstruction, *IJCV* **124**(1): 2–17.

Lafarge, F. and Alliez, P. (n.d.): Surface reconstruction through point set structuring, *Proc. of Eurographics*, Volume 32, 225–234.

Lafarge, F. and Mallet, C. (2012): Creating Large-Scale City Models from 3D-Point Clouds: A Robust Approach with Hybrid Representation, *IJCV* **99**(1): 69–85.

Landrieu, L. and Simonovsky, M. (2018): Large-scale point cloud semantic segmentation with superpoint graphs, *CVPR*, 4558–4567.

Li, M., Nan, L., Smith, N. and Wonka, P. (2016a): Reconstructing building mass models from UAV images, *Computers and Graphics* **54**: 84 – 93: Special Issue on CAD/Graphics.

Li, M., Wonka, P. and Nan, L. (2016b): *Manhattan-World Urban Reconstruction from Point Clouds*, 54–69.

Lim, E. H. and 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. and Yang, R. (2013): Semantic decomposition and reconstruction of residential scenes from lidar data, *SIGGRAPH* **32**(4): 1–10.

Lo, A. Y. (1984): On a class of Bayesian nonparametric estimates: I. density estimates, *Ann. Statist.* **12**(1): 351–357.

Marton, Z. C., Rusu, R. B. and Beetz, M. (2009): On Fast Surface Reconstruction Methods for Large and Noisy Datasets, *ICRA*, 3218–3223.

Mayer, H., Bartelsen, J., Hirschmüller, H. and Kuhn, A. (2012): Dense 3D Reconstruction from Wide Baseline Image Sets, *Outdoor and Large-Scale Real-World Scene Analysis*, Springer Lecture Notes in Computer Science 7474, 285–304.

Mitra, R. and Müller, P. (2015): *Nonparametric Bayesian Inference in Biostatistics*, Springer Series in Statistics.

Monszpart, A., Mellado, N., Brostow, G. and Mitra, N. (2015): RAPter: Rebuilding man-made scenes with regular arrangements of planes, *SIGGRAPH* **34**(4): 103:1–103:12.

Mueller, P., Andrés Quintana, F., Jara, A. and Hanson, T. (2015): Bayesian nonparametric data analysis, *Springer Series in Statistics*.

Muja, M. and Lowe, D. G. (2014): Scalable nearest neighbor algorithms for high dimensional data, *PAMI* **36**(11): 2227–2240.

Müller, P., Wonka, P., Haegler, S., Ulmer, A. and Van Gool, L. (2006): Procedural modeling of buildings, *ACM Transactions on Graphics* **25**(3): 614–623.

Munoz, D., Bagnell, J. A. D., Vandapel, N. and Hebert, M. (2009): Contextual classification with functional max-margin Markov networks, *CVPR*, 975–982.

Mura, C., Mattausch, O. and Pajarola, R. (2016): Piecewise-Planar Reconstruction of Multi-Room Interiors with Arbitrary Wall Arrangements, *Computer Graphics Forum* **35**(7): 179–188.

Musialski, P., Wonka, P., Aliaga, D. G., Wimmer, M., Van Gool, L. and Purgathofer, W. (2013): A Survey of Urban Reconstruction, *Computer Graphics Forum* **32**(6): 146–177.

Nan, L. and Wonka, P. (2017): Polyfit: Polygonal surface reconstruction from point clouds, *ICCV*, 2372–2380.

Neal, R. M. (2000): Markov chain sampling methods for Dirichlet process mixture models, *Journal of Computational and Graphical Statistics* **9**(2): 249–265.

Nguatem, W. and Mayer, H. (2016): *Contiguous Patch Segmentation in Pointclouds*, GCPR, 131–142.

Nguatem, W. and Mayer, H. (2017): Modeling urban scenes from pointclouds, *ICCV*, 3857–3866.

Nguatem, W., Drauschke, M. and Mayer, H. (2013a): Roof Reconstruction from Point Clouds Using Importance Sampling, *City Models, Roads and Traffic (CMRT). Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **II-3/W3**: 73–78.

Nguatem, W., Drauschke, M. and Mayer, H. (2013b): Roof reconstruction from point clouds using importance sampling, *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* **II-3/W3**: 73–78.

Nguatem, W., Drauschke, M. and Mayer, H. (2014): Localization of Windows and Doors in 3Ð Point Clouds of Facades, *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* 87–94.

Nishida, G., Garcia-Dorado, I., Aliaga, D. G., Benes, B. and Bousseau, A. (2016): Interactive sketching of urban procedural models, *SIGGRAPH* **35**(4): 1–11.

Nüchter, A. (2016): Robotic 3D scan repository, `http://kos.informatik.uni-osnabrueck.de/3Dscans/`.

Oesau, S., Lafarge, F. and Alliez, P. (2015): Planar Shape Detection and Regularization in Tandem, *Computer Graphics Forum* **35**(1): 203–215.

Papon, J., Abramov, A., Schoeler, M. and Wörgötter, F. (2013): Voxel cloud connectivity segmentation - supervoxels for point clouds, *CVPR*, 2027–2034.

Phadia, E. G. (2016): *Prior Processes and Their Applications (Nonparametric Bayesian Estimation)*.

Pham, T., Chin, T., Yu, J. and Suter, D. (2014): The random cluster model for robust geometric fitting, *PAMI* **36**(2): 1658–1671.

Poullis, C. (2013): A framework for automatic modeling from point cloud data, *PAMI* **35**(11): 2563–2575.

Poullis, C. and You, S. (2009): Photorealistic large-scale urban city model reconstruction, *IEEE Transactions on Visualization and Computer Graphics* **15**(4): 654–669.

Pu, S. and Vosselman, G. (2009): Knowledge based reconstruction of building models from terrestrial laser scanning data, *JISPRS* **64**(6): 575 – 584.

Qian, N. and Qian, D. N. (1997): *Binocular disparity and the perception of depth*, Elsevier.

Radke, R. J. (2012): *Computer Vision for Visual Effects*, Cambridge University Press, New York, NY, USA.

Raguram, R., Chum, O., Pollefeys, M., Matas, J. and Frahm, J.-M. (2013): USAC: A universal framework for random sample consensus, *PAMI* **35**(8): 2022–2038.

Ripperda, N. and Brenner, C. (2006): Reconstruction of Facade Structures Using a Formal Grammar and RjMCMC, 750–759.

Rottensteiner, F., Sohn, G., Gerke, M., Wegner, J., Breitkopf, U. and Jung, J. (2014): Results of the ISPRS benchmark on urban object detection and 3D building reconstruction, *JISPRS* **93**: 256–271.

Rusu, R. B. and Cousins, S. (2011): 3D is here: Point Cloud Library (PCL), *ICRA*, 1–4.

Saatkamp, J. and Schmittwilken, J. (2007): Generative models and Markov Chain Monte Carlo techniques for detection and reconstruction of stairs from point clouds, *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXVI (4/W54)*, 111–119.

Schnabel, R., Wahl, R. and Klein, R. (2007): Efficient RANSAC for Point-Cloud Shape Detection, *Computer Graphics Forum* **26**(2): 214–226.

Schönberger, J. L., Zheng, E., Pollefeys, M. and Frahm, J.-M. (2016): Pixelwise view selection for unstructured multi-view stereo, *ECCV*, 501–518.

Sedgewick, R. (2002): *Algorithmen (2. Aufl.)*, Pearson Studium.

Shilane, P., Min, P., Kazhdan, M. and Funkhouser, T. (2004): The Princeton shape benchmark, 167–178.

Siciliano, B. and Khatib, O. (2007): *Springer Handbook of Robotics*, Springer.

Snavely, N., Seitz, S. M. and Szeliski, R. (2006): Photo tourism: Exploring photo collections in 3D, *SIGGRAPH* **25**: 835–846.

Straub, J., Chang, J., Freifeld, O. and Fisher III, J. W. (2015): A Dirichlet process mixture model for spherical data, *AISTATS* **38**: 930–938.

Straub, J., Rosman, G., Freifeld, O., Leonard, J. J. and Fisher III, J. W. (2014): A mixture of Manhattan frames: Beyond the Manhattan world, *CVPR*, 3770–3777.

Strecha, C., von Hansen, W., Gool, L. J. V., Fua, P. and Thoennessen, U. (2008): On benchmarking camera calibration and multi-view stereo for high resolution imagery, *CVPR*, 1–8.

Talton, J. O., Lou, Y., Lesser, S., Duke, J., Měch, R. and Koltun, V. (2011): Metropolis procedural modeling, *SIGGRAPH* **30**(2): 11:1–11:14.

Torr, P. H. S. and Zisserman, A. (2000): MLESAC: A new robust estimator with application to estimating image geometry, *CVIU* **78**: 138–156.

Verdie, Y., Lafarge, F. and Alliez, P. (2015): LOD generation for urban scenes, *SIGGRAPH* **34**(3): 30:1–30:14.

Vosselman, G. (2013): Point cloud segmentation for urban scene classification, *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **XL-7/W2**(2): 257–262.

Weinmann, M. (2016): *Reconstruction and Analysis of 3D Scenes: From Irregularly Distributed 3D Points to Object Classes*, Springer.

Wu, I.-C., Lu, S.-R. and Hsiung, B.-C. (2015): A BIM-based monitoring system for urban deep excavation projects, *Visualization in Engineering* **3**(1): 2.

Yershova, A. and LaValle, S. M. (2004): Deterministic sampling methods for spheres and SO(3), *ICRA*, Volume 4, 3974–3980.

Zeng, H., Wu, J. and Furukawa, Y. (2018): Neural procedural reconstruction for residential buildings, 759–775.

Zhou, Q.-Y. and Neumann, U. (2009): A streaming framework for seamless building reconstruction from large-scale aerial LiDAR data, *CVPR*, 2759–2766.

Zhou, Q. Y. and Neumann, U. (2012): 2.5D building modeling by discovering global regularities, *CVPR*, 326–333.

Zhou, Q.-Y. and Neumann, U. (2013): Complete residential urban area reconstruction from dense aerial LiDAR point clouds, *Graphical Models* **75**(3): 118–125.

Zhu, L., Shen, S., Gao, X. and Hu, Z. (2018): Large scale urban scene modeling from MVS meshes, *ECCV*, 640–655.