Geometric modeling of man-made objects at different level of details

Hao Fang

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

Modélisation géométrique à différent niveau de détails d'objets fabriqués par l'homme

Geometric modeling of man-made objects at different level of details

Hao FANG

INRIA Sophia-Antipolis

Présentée en vue de l'obtention du grade de docteur en Informatique **d'**Université Côte d'Azur **et de** INRIA Sophia-Antipolis
Dirigée par : Florent LAFARGE

Soutenue le : 16/01/2019

Abstract

Geometric modeling of man-made objects from 3D data is one of the biggest challenges in Computer Vision and Computer Graphics. The long term goal is to generate a CAD-style model in an as-automatic-as-possible way. To achieve this goal, difficult issues have to be addressed including (i) the scalability of the modeling process with respect to massive input data, (ii) the robustness of the methodology to various defect-laden input measurements, and (iii) the geometric quality of output models. Existing methods work well to recover the surface of free-form objects. However, in case of man-made objects, it is difficult to produce results that approach the quality of high-structured representations as CAD models.

In this thesis, we present a series of contributions to the field. First, we propose a classification method based on deep learning to distinguish objects from raw 3D point cloud. Second, we propose an algorithm to detect planar primitives in 3D data at different level of abstraction. Finally, we propose a mechanism to assemble planar primitives into compact polygonal meshes. These contributions are complementary and can be used sequentially to reconstruct city models at various level-of-details from airborne 3D data. We illustrate the robustness, scalability and efficiency of our methods on both laser and multi-view stereo data composed of man-made objects.

Keywords: Point cloud, polygonal mesh, semantic segmentation, deep learning, shape detection, geometric primitives, surface reconstruction

Résumé

La modélisation géométrique d'objets fabriqués par l'homme à partir de données 3D est l'un des plus grands défis de la vision par ordinateur et de l'infographie. L'objectif à long terme est de générer des modèles de type CAO de la manière la plus automatique possible. Pour atteindre cet objectif, des problèmes difficiles doivent être résolus, notamment (i) le passage a l'échelle du processus de modélisation sur des données d'entrée massives, (ii) la robustesse de la méthodologie contre des mesures d'entrées erronés, et (iii) la qualité géométrique des modèles de sortie. Les méthodes existantes fonctionnent efficacement pour reconstruire la surface des objets de forme libre. Cependant, dans le cas d'objets fabriqués par l'homme, il est difficile d'obtenir des résultats dont la qualité approche celle des représentations hautement structurées, comme les modèles CAO.

Dans cette thèse, nous présentons une série de contributions dans ce domaine. Tout d'abord, nous proposons une méthode de classification basée sur l'apprentissage en profondeur pour distinguer des objets dans des environnements complexes a partir de nuages de points 3D. Deuxièmement, nous proposons un algorithme pour détecter des primitives planaires dans des données 3D à différents niveaux d'abstraction. Enfin, nous proposons un mécanisme pour assembler des primitives planaires en maillages polygonaux compacts. Ces contributions sont complémentaires et peuvent être utilisées de manière séquentielle pour reconstruire des modèles de ville à différents niveaux de détail à partir de données 3D aéroportées. Nous illustrons la robustesse, le passage a l'échelle et l'efficacité de nos méthodes sur des données laser et multi-vues stéréo sur des scènes composées d'objets fabriqués par l'homme.

Mots cléfs: Nuage de points, maillage polygonal, segmentation sémantique, apprentissage approfondi, détection de forme, primitives géométriques, reconstruction de surface

Contents

Page

Chapter 1
Introduction

1.1 Context

Computer-aided design (CAD) is the computer graphic technique to aid in the industrial design, creation and analysis of productions using geometric modeling techniques [SSBB15]. The core outputs of CAD systems are *CAD models*, which are typically defined as an assemblage of parametric geometric shapes such as curves, surface primitives and volumes.

There are two main types of CAD models. The first one is made by connecting regular shapes as planes, cylinders, cones, spheres, tori and other simple geometric shapes. This type of CAD models is typically well adapted to represent man-made objects as buildings and mechanical pieces. The second type of CAD models uses more complex geometric shapes, mainly *non-uniform rational basis spline models* (NURBS) that can better describe objects composed of free-form surfaces such as organic entities. Commercial software such as *autoCAD* and *Solidworks* allow us to generate complex CAD models. The graphical-user interfaces of these software not only produce CAD models with complete geometric attributes, but also manage the associative relationships and geometric constraints between them.

CAD models are everywhere in our everyday life, going from reverse engineering to telecomunnications through urbanism and entertainment. They are developed to produce mechanical devices in industrial areas such as aerospace and automobile. Practitioners take use of interactive softwares to convert their designing idea to digital CAD models on computers, which makes it possible to perform different kinds of simulation to analyze their creation with physical considerations. With the development of CAD techniques, CAD models are extensively employed in the entertainment. Designers produce imaginary objects and scenes for movies or computer games and apply advanced rendering capabilities for fancy visualization. They are also used in urbanism to digitally recreate cities for visualization-based and simulation-based applications.

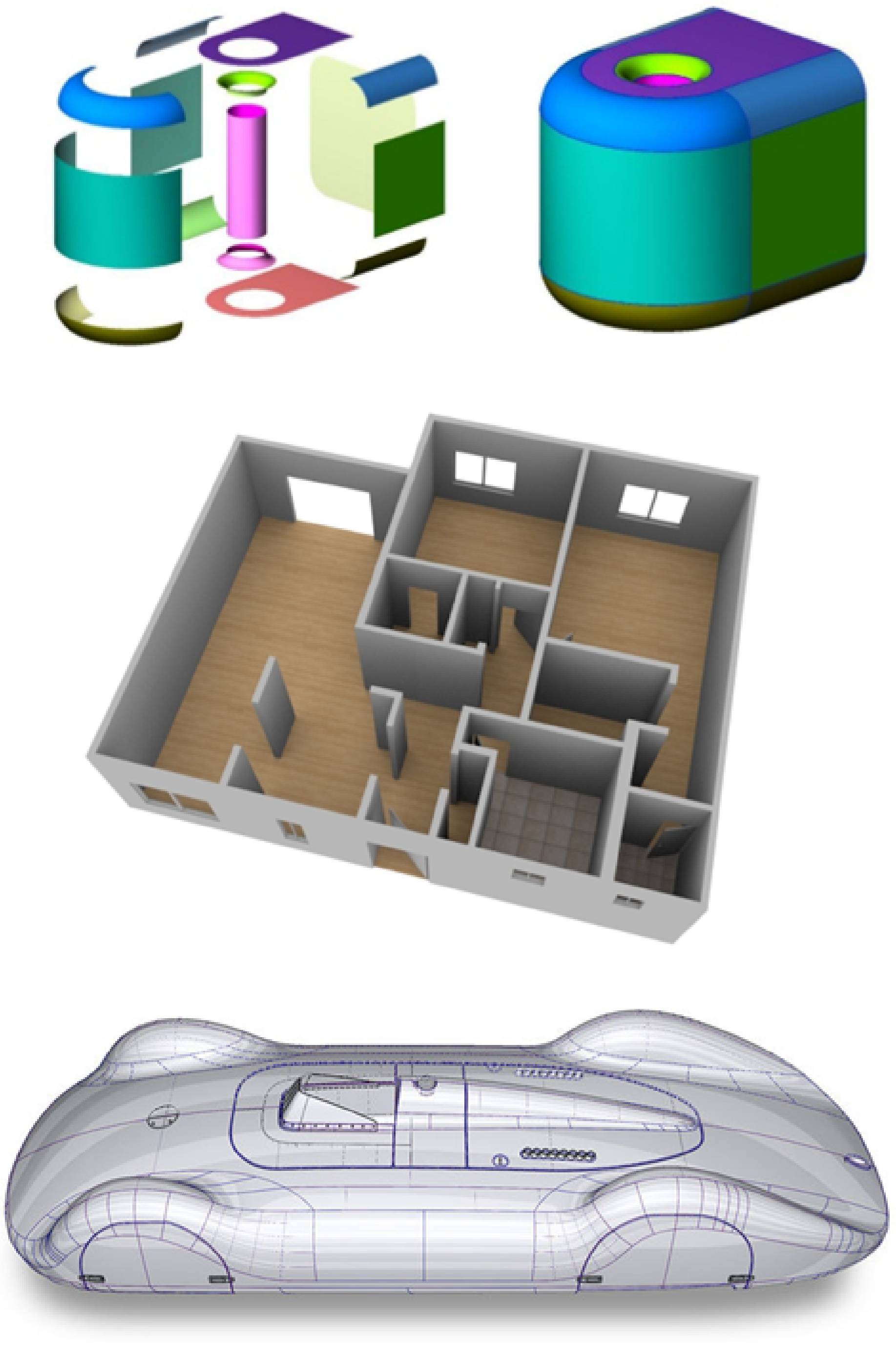


Figure 1.1: CAD models. The first two models on top are represented as a collection of connected surface elements, while the model on bottom is a surface extracted from parametric functions determined by control points. Images from c3dlabs.com, cgi.tutsplus.com and carbodydesign.com

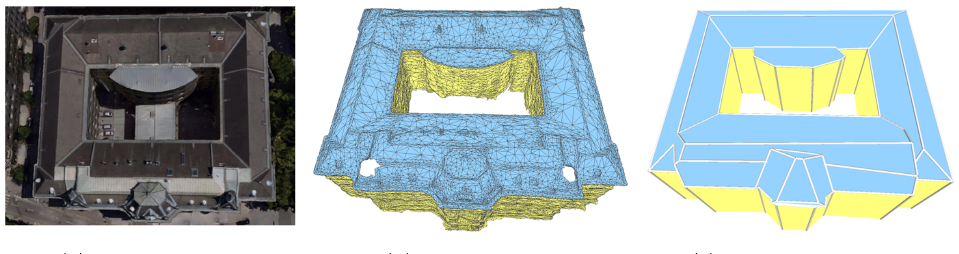


Figure 1.2: Dense meshes vs CAD-style models. Given (a) satellite images, existing Multi-View Stereo approach [HKLP09] produces dense triangular mesh (b). Compared with a CAD-style model (c), the memory storage is 1941Kb and 23Kb for dense triangular mesh and CAD model respectively. Dense mesh of the building consists 12447 *roof* triangle facets, while CAD representation only contains 18 *roof* polygonal facets. Lower number of polygonal facets highly decreases computational complexity for downstream numerical simulation, i.e. heat transfer modeling on roof of buildings.

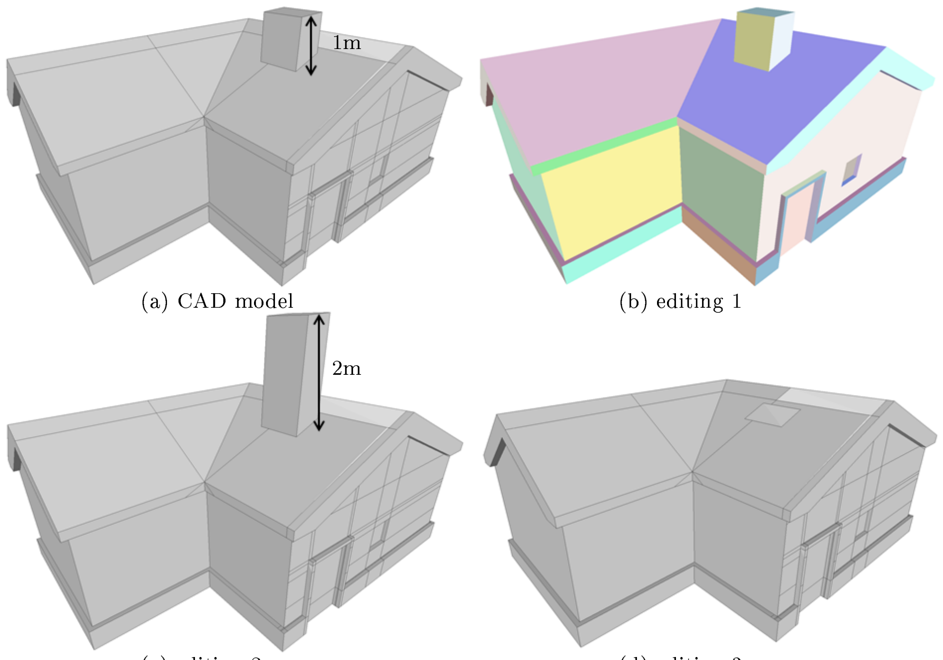


Figure 1.3: Editing capacity of (a) a CAD model assembled by 51 planar shape. All primitives share several common vertices and edges with their neighbors. The structure information and topological restriction enable users to easily (b) painting each part for rendering, (c) modify the height of chimney on the roof and (d) eliminate the chimney from the house.

CAD models exhibit interesting properties for these applications. They are compact, structure-aware and easy to edit.

Compactness. In case of man-made objects, CAD models are usually compact in term of the number of geometric shapes as shown in Figure 1.2. High compactness brings benefit to memory storage and computational efficiency of operations for rendering and simulation.

Structure awareness. CAD models are assembled in a way that each shape typically represents a semantic part of the object, i.e. a planar shape will typically represent a roof section for a building. This property facilitates the use of CAD models and enables users to operate directly on the desired parts, such as thermodynamics modeling on the roof.

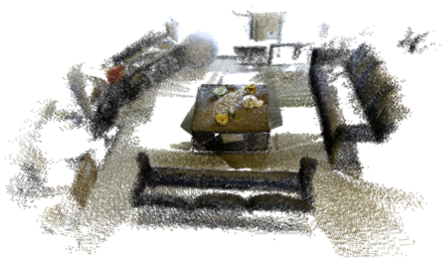
Editing capacity. CAD models usually contain connectivity relationships between surface elements. Those constraints restrict the spatial position of geometric shapes while moving the others. This topological property makes it easy for the users to edit CAD models according their requirements as illustrated in Figure 1.3.

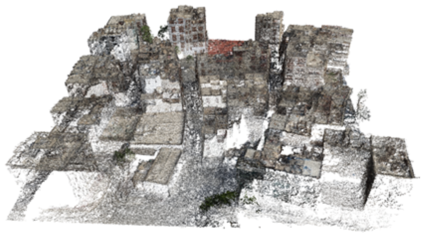
CAD models can be used both for designing imaginary objects and digitalized existing ones. In the later case, users can rely on data measurements, typically Laser scans and multiview-stereo images in 3D, to make the CAD models as close as possible of the physical objects. Such interactive operations are extremely fastidious for the user. For a single CAD model, it is a time-consuming work based on trial and error to reconstruct an geometrically-accurate CAD model, even for an experienced user. Turning this chain of interactions into an automatic process is one of the major challenges in computer graphics and computer vision.

1.2 Challenges

Reconstructing objects from physical measurements in an automatic way has been deeply studied in the literature with mainly methods producing dense meshes. Efficient commercial solutions have been proposed to generate such meshes from Laser and multiview images like ContextCapture from Bentley. Although these meshes have usually a good geometric accuracy, they ignore the semantic and structural dimensions of the objects, contrary to CAD models. Such meshes are typically exploited for immersive experiences into virtual scenes and also 3D printing, but can not be directly used for advanced simulation.

(a) Laser point cloud[LA13]


(b) RGB-D point cloud [HFBM13]


(c) MVS point cloud [LNSW16]

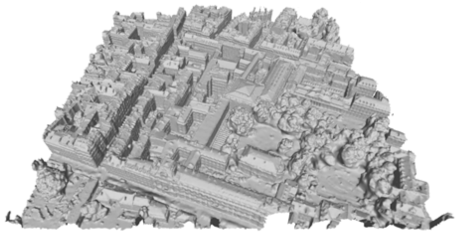

(d) MVS dense mesh [HKLP09]

Figure 1.4: Multi-resources of input 3D data.

Input \ Defect	Noise	Outliers	Missing data	Nonuniform
Laser point cloud	Fair	Good	Fair	Good
RGB-D point cloud	Poor	Poor	Fair	Poor
MVS point cloud	Poor	Fair	Poor	Fair
MVS dense mesh	Good	Good	Fair	Fair

Table 1.1: Subjective evaluation of defects contained in the different types of data measurements.

The main objective of this PhD is to **develop methods to automatically produce CAD-style models from 3D data measurements**. We restrict the study to CAD models composed of planar shapes. In spite of the simplicity, these shapes allow us to represent a large range of man-made objects fairly, such as buildings.

Current solutions mainly focus on automatically processing a part of the scanned data and then let a human-expert interactively complete the reconstruction [ASF+13, CC08]. This interactive framework provides accurate results for individual objects within a limited time. However, human-interaction partly limits the use of such methods while handling more com-

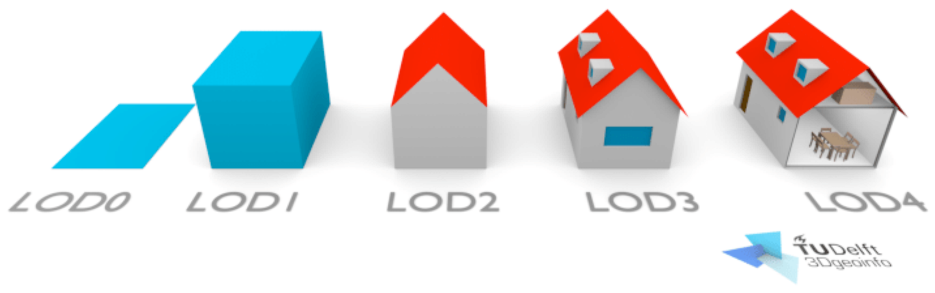


Figure 1.5: Five LODs defined by CityGML 2.0. Each LOD representation can be applied for different use. (Figure adapted from [BLS16])

plex models and scenes. The costs of human resources and manipulation time increase dramatically while dealing with large scale scenes including dozens of objects. The goal of this work is to provide tools to generate CAD-style models from scanned 3D data in an as-automatic-as possible way. This is a scientific challenge with many technical difficulties that we expose below.

Robustness. One obstacle to generating CAD-style model is the robustness of proposed algorithm to input data obtained from multi-resources. Typical 3D data is usually represented as either point cloud obtained from RGB-D sensors and LiDAR scanners, or dense triangular mesh reconstructed from MVS system (Figure 1.4). Each type of data contains different types of defects. Table 1.1 discusses the property of each defect. These artifacts impact severely the 3D reconstruction pipeline in terms of geometric accuracy and computational time. For instance, recovering the objects with only partial input data is a major scientific challenge that typically requires the design of algorithms exploiting geometric and structure information. Also, the existence of large amount of noise points can strongly decrease the robustness of surface reconstruction algorithms.

Scalability. Scalability of algorithms is another concern. Existing methods typically perform well on small scenes or simple objects. However, complex scenes usually contain many complex objects. To process such complex scenes, existing methods typically exploit strong geometric assumptions on the output models, such as Manhattan World assumptions. This reduces the computational complexity of methods, but also leads to produce models with low geometric accuracy.

Structure and semantic-awareness. Structure recovery is another crucial requirement of output model. In shape analysis, *structure* is a generic term ranging from the canonical shape of each part of the object to their adjacency geometric relationship, i.e. coplanar, symmetric, parallelism etc [MWZ+13]. Such knowledge is extremely relevant and useful for further application while dealing with man-made objects and large scale scenes. For instance, generating a specific CityGML formalism LOD2 model requires the main sections of buildings to be represented by flat planar shapes. The main challenge in this part is how to extract accurate structure information from raw input and adapt it into the final reconstruction phase.

On the other hand, obtaining the final CAD-style model is a trade-off between geometric accuracy and structure recovery, while the latter criteria measures the capacity of generating meaningful LODs [BTS+17]. In this case, a better choice is to produce a sequence of reconstruction models, each preserving a target LOD representation [VLA15, BLS16] as shown in Figure 1.5. However, how to exploit the coherence of LODs across the scene without specific defined rules is still not well studied in the literature. Constructing and exploring scale space for various LODs modeling is a difficult open issue.

Semantic-awareness is a complementary property of the output model, which brings great benefit for further use if each structure part contains a semantic attribute. This traditional classification problem has been widely studied ranging from individual object to complex scenes. However, it is still a challenging problem to understand the mutual interaction between different parts in a complex scene.

Geometric and topological correctness. The reconstructed CAD-style models also have to meet several specificities to be used in real-world applications. The most important evaluation criteria is geometric accuracy, which compares the ground truth of the scan and output model via geometric error, i.e the Hausdorff distance. Unfortunately, we have no access to ground truth representation for most of the cases in real application. An alternative is to use input 3D data instead. A high quality output model is required to as close as possible to input 3D data from geometric measurement.

The topological quality of output model is concerned as well, including *2d-manifold* and *self-intersection free* properties. Such 3D model is highly required for various downstream applications, i.e. 3D printing, numerical simulation etc. The main challenging here is to embed those topological properties into the reconstruction framework as soft or hard constraints in the frameworks.

1.3 Contributions

To address those challenges, this thesis proposes three contributions. That forms a reconstruction pipeline from 3D data measurements. First, we decompose a scene into different individual objects. This task can be formalized as a classification problem by predicting the potential semantic label of each 3D data in the scene and group the neighboring datum with same semantic label together as an object. Next, we approximate the surface of each object by a set of simple geometric primitives, i.e. planes at different level of details. This abstraction mechanism highly decreases the complexity of original data under the assumption that the surface of man made objects can be represented by piecewise planar shapes. Finally, the isolated planes are assembled together to form a compact CAD-style model.

3D semantic segmentation. With the development of 3D point cloud acquisition techniques, efficient and robust algorithms to process large scale point cloud is crucial for further applications. Among them, semantic segmentation of 3D data has been one of the most essential steps for several 3D vision tasks, such as autonomous driving, robotics and augmented reality [TCA+17]. Over the last decade, state-of-the-art algorithms would extract low level features from point cloud via geometric prior knowledge for various tasks [NBW12, LM12, WJM13]. Recently, deep learning based feature extraction methods have shown remarkable performance on 2d image semantic segmentation [LSD15, BKC15, CPK+18]. Afterwards, applying deep learning techniques on 3D data has drawn considerable attention in computer vision and photogrammetry community. Because point cloud is unordered and unstructured, it is impossible to apply *convolutional neural network* (CNN) directly on point cloud for end-to-end training. An alternative approach is to first convert point clouds into other intermediate 3D representations and then apply CNN for various task, i.e. multi-view RGB images [SMKLM15, KAMC17, BGLSA18] and voxels [MS15, WSK+15, QSN+16, TCA+17, HSL+17]. However, all these intermediate representations lead to a loss of 3D information among the points or suffer from memory-consuming issue. A milestone work PointNet proposed by [QSMG17] utilizes a composition of basic operators, i.e. *multilayer perceptron* (MLP) and *max pooling* as deep network to extract features directly from point cloud. Surprisingly, this simple architecture learns order-invariant pointwise features and exhibits good performance on multiple tasks. Later on, several approaches were designed to enrich pointwise features by aggregating information in local regions [QYSG17, HWN18, WSL+18], producing more accurate segmentation results on large scale datasets. However, the receptive field is still not clear for points of complex scenes, where objects of different scales are close to each other. In this case, a prior knowledge of global regional context is crucial for more accurate prediction [ZSQ+17]. None of the mentioned methods incorporate reasonable global contextual information to provide a richer pointwise feature.

In Chapter 3, we address the problem of increasing the receptive field of points by inferring regional global contextual information. More specifically, inspired from [ZSQ+17], we design a *3D pyramid scene parsing network* (3d-PSPNet) to enrich local pointwise feature with multi-scale global contextual information. We validate our 3d-PSPNet on three large scale dataset with two baselines. Experimental results prove that the enriched features provide more decent prediction than using the baseline model only. The goal of our approach is not to achieve state-of-the-art performance on all the datasets, but to propose a generic module that can be concatenated with any state-of-the-art 3D neural network to infer richer pointwise features.

Shape detection. Shape detection from raw 3D data is a long-standing problem whose goal consists in turning a large amount of geometric data into a higher level representation based on simple geometric shapes. Instead of reasoning at the scale of 3D atomic elements such as points, triangular facets or voxels, it is often more appealing to directly handle larger geometric shapes in order to both reduce the algorithmic complexity and analyze objects with a higher representation level. Most common geometric shapes include lines, planes and quadrics. In this work, we focus on planar shapes due to their relevance to man-made environments [MZL+09].

Shape detection is typically used as a prior step in a large variety of vision-related tasks ranging from surface reconstruction [BdLGM14, CLP10, SSS09, ZN12, NW17] to object recognition [CSM12, OLA16a] and data registration [FMMCAJ13, ZJM12]. Existing algorithms typically require two user-specified parameters: (i) *a fitting tolerance* ε that specifies the maximal distance of a datum to its associated geometric shape, and (ii) *a minimal shape size* σ that specifies how large a group of samples must be to be considered as a geometric primitive—typically, a number of inliers when dealing with point clouds, or a minimum area for meshes. Finding parameter values that produce desirable results often involves fastidious manual labor: surprisingly, the incidence of these two parameters on shape detection has not been formally studied in the literature.

In Chapter 4, we propose an efficient exploration of this (ε, σ) space of geometric abstractions to find the *structural scales* of an input geometry, i.e., the few simplified representations that are truly meaningful to capture the structure of man-made objects. From a progressive planarity-driven coarsening of the input data, we demonstrate that we can reliably detect structural scales whose characteristics are learned from training sets of different types of objects such as buildings, house furniture, or cars.

Polyhedral surface reconstruction. Primitives are disconnected from each others and constitute an intermediate representation between input 3D data and the output mesh. The third step consists in assembling primitives into a surface mesh. One strategy consists in connecting the primitives using proximity and structural considerations [ASF$^+$13, CC08, LA13, SFF11]. Despite being fast, this solution is not robust to defect-laden data, in particular when primitives are over- or under-detected or when erroneous connections between primitives exist. A more robust strategy consists in slicing a 3D domain by extending the primitives. This leads to the creation of a partition of polyhedral cells or polygonal facets [BdLGM14, CLP10, NW17, VLA15]. The surface is then extracted by labeling the cells as inside or outside the surface, or equivalently, by selecting facets to be part of the surface. Because each primitive exhaustively slices all the others, this solution is more robust to defect-laden data than the first strategy. However, its main shortcoming is the computational burden for slicing the primitives into a partition of atomic surface and volume elements, with typically unreasonable timing and memory issues when more than one hundred primitives are handled.

In Chapter 5, we propose a solution to specifically address the scalability issue of the slicing-based methods. While these methods reason on dense polyhedral cell partitions, we instead build a more flexible and lighter data-structure. The latter is spatially-adaptive in the sense that a primitive slices a restricted number of relevant primitives based on spatial proximity considerations. Moreover, its atomic elements have different structural meanings that will guide the extraction of the output surface. We also propose a surface extraction mechanism that operates from such an irregular data-structure in which cells are not necessarily convex and can have a non-null volume intersection with other cells.

Besides the data-structure, our solution brings several original technical ingredients to the field. Our algorithm has a preliminary step that analyzes the connectivity of primitives in order to search for structurally-valid surface components. This allows us to quickly process a part of the input primitives and solve obvious primitive assembling situations. We also measure data fidelity to primitives directly without relying on input 3D data. Indeed, measuring data fidelity to 3D data makes sense only if primitives could be modified during the assembling step, which would require a dynamic partitioning data-structure. As a result, our outputs do not suffer from artifacts frequently found with existing methods. It also allows our algorithm to run on multiple types of 3D data as dense meshes, and not only point clouds. Our algorithm also offers to the user the possibility to relax some standard geometric properties on the delivered surface as its watertightness or the intersection-free guarantee.

We demonstrate the potential of our algorithm in terms of flexibility, robustness and scalability on different types of objects, going from buildings to mechanical pieces through even free-form shapes. In particular, we show our algorithm is faster and more scalable than state-of-the-art methods by a significant margin.

The structure of this thesis is organized as follows:

- Chapter 2 covers the related works of these problems.
- Chapter 3 designs a deep learning module to aggregate multi-scale contextual clue in a pyramid manner.
- Chapter 4 introduces a scale space exploration mechanism to abstract 3D object at structural scales.
- Chapter 5 proposes an efficient algorithm to reconstruct a polyhedral mesh from large number of planes.
- Chapter 6 gives the conclusion and perspectives of this thesis.

CHAPTER 2

Related work

In this chapter, we review the literature on three aspects of our work: (i) semantic segmentation of 3D data (ii) shape detection from 3D data (iii) surface reconstruction from 3D data.

2.1 Semantic segmentation of 3D data

Given 3D data, one of the most typical problems is understanding its intrinsic properties, which relies on designing robust shape descriptors to characterize various 3D shapes. Such shape descriptors can be directly employed for a set of applications, which includes point correspondence or matching, shape retrieval, object recognition and semantic segmentation etc. We review this traditional and crucial problem from two kinds of methods involved in literature: hand-crafted feature extraction and deep feature learning methods.

2.1.1 Methods exploiting hand-crafted descriptors

Direct descriptors. Typical hand-crafted feature extraction methods rely on designing scale-invariant and rigid-transformation-invariant descriptors directly from input point cloud or mesh. The basic idea is to exploit local clues by analyzing the interaction between each point or triangular facet and their neighbors in a local region.

[JH99] introduced a 3D shape descriptor known as *spin image*, which is a 2D-histogram containing the projection of 3D point along the direction of its normal vectors. This descriptor is pose-invariant and robust to recognize objects even in cluttered scenes. *Shape context* presented by [BMP01] is defined as a point descriptor measuring distribution of neighbors over the local neighborhood according to the relative spatial position to the centering point. Such robust and compact descriptor is then applied to measure similarity between points from two objects for shape matching. Another widely used pointwise feature extractor is *Fast Point Feature Histograms (FPFH)* which is proposed by [RBB09], extracting multi-dimensional local geometry descriptors around each point. These 3D descriptors are usually embedded into a learning procedure to recognize objects from complex scenes. For instance, [GKF09] designed an object recognition system for point cloud of urban environment. They first extracted shape descriptors for each segmented object in the scene using spin image method proposed in [JH99], and then trained a support vector machines (SVM) classifier for recognition. More recently, [HWS16] proposed a fast multi-scale neighborhood feature extraction framework to cope with urban scene point clouds with strong density variation. Different to previous methods, [LM12] took use of prior geometric attributes of various objects in urban scenes as pointwise features, i.e. elevation, scatter, planarity etc. Then the whole urban scene is segmented into four groups: *facade*, *roof*, *vegetable* and *ground* through an unsupervised Markov Random Field (MRF) model.

Compared to a point cloud, a dense triangular mesh preserves the topology information that can be directly used for feature extraction. Typical mesh descriptors include *Gaussian curvature* ([GCO06]), *shape diameter function* and *average geodesic distance* ([HSKK01]). Besides, ([NN07]) proposed an edge and coder detector as scale-dependent geometric features from triangular mesh. More recently, [TM14] involved the *scatter matrix* as point descriptor in a mesh and then used it for interest point detection through a binary classification formulation. Those descriptors could also be applied to supervised learning approach for different tasks. [KHS10] took triangular mesh as input and designed descriptors for both individual facet and pair of adjacent facets. The unary feature is a 374-dimensional vector containing shape context, spin images, curvatures etc, while pairwise features is a 191-dimensional histogram referring dihedral angles, shape diameter differences, contextual label features etc. The final mesh segmentation result is computed through a Conditional Random Field (CRF) with a pre-learned jointBoost classifier.

Indirect descriptors. Large scale scenes usually consist of more than a few millions of points or meshes composed by multiple objects. Each object can be seen as an abstraction of several geometric primitive, i.e. planes, cylinders and spheres. Intuitively, instead of designing point-level or facet-level descriptors, an alternative is to propose descriptors on these intermediate representations. These kinds of indirect descriptors not only increase the robustness to point cloud with noise and outliers, but also avoid the scalability issue.

[OLA16a] introduced an object recognition approach by first extracting planar shapes from raw point cloud and then analyzing relative geometric relationship between extracted planar parts as global features. The classification result is returned by a pre-trained Random Forest classifier. [RLA17] employed a supervised learning MRF approach for textured urban mesh semantic segmentation. The whole mesh is firstly over-segmented into a set of planar shapes. Each primitive is then represented by a descriptor combining both geometric and photometric clues. Finally, a Random Forest classifier is trained on the combined features for semantic labeling. The patch-based feature extraction methods are also widely used for 3D object detection. [ASZ+16] proposed a large-scale semantic parsing approach. The input raw point cloud of an entire building is firstly parsed into different meaningful spaces. Then a 3D sliding window method is exploited for 3D object detection. Each sliding window is described as several geometric features combining both local and global attributes. A pre-trained multi-class SVM detector is further used to evaluate each candidate sliding window.

Besides supervised learning approaches, 3D patch descriptors can also be used for unsupervised frameworks. [MPM+14] proposed a patch-based representation method to characterize geometric descriptors on each fitting rectangle instead of on the original point cloud. Certain characteristic features, i.e. area, ratio of width to length, non-coplanarity are computed for each corresponding fitting rectangle. Such representations are then exploited to measure the similarity between each parts. Final object detection and classification are performed through clustering approach on the embedded feature space. Similarly, [HFL12] co-segmented different parts of objects within same categorization via clustering the over-segmented patches in multiple feature spaces. More recently, some part-level descriptors are designed by considering interaction between different shapes for more advanced tasks, i.e. *Interaction Context* (ICON) [HZvK+15] for 3D shape functionality analysis. For large scale urban mesh semantic segmentation, [VLA15] introduced geometric attributes of each superfacets according to prior knowledge of urban scenes. Recently, [ZLHW17] proposed a similar approach for point cloud urban scene understanding. Besides unary geometric features of each supervoxesl, they also formulated higher-order semantic relationships between patches into the MRF model.

2.1.2 Deep learning methods

With the success of deep learning techniques, especially Convolutional Neural Network (CNN) applied on 2D images analysis, the major concern in the 3D Vision community is to find an alternative way to apply deep learning methods on 3D data analysis. Traditional hand-crafted feature extraction methods aims at designing robust features that are explainable. However, deep learning techniques prefer to construct a complex model composed of large amount of basic operations to represent features. We review recent deep learning approaches for different 3D data representations, including *voxels*, *multi-view images*, *mesh* and *point clouds*.

Multi-view images. With the success of deep learning techniques being applied on 2d image classification [KSH12] and segmentation [LSD15, BKC15, CPK+18], a direct question is how to employ this powerful feature learning tool, i.e. CNN directly to 3D data. A straightforward idea is to represent 3D data via multi-view rendering images and design a multi-view CNN network for further application. Based on this idea, [SMKLM15] first proposed such an architecture for 3D data recognition. Because the output is just a label of the rendered image, this method is not designed to incorporate the correlation among all images. More recently, [KAMC17] employs image-based *Fully Convolutional Network (FCN)* for part-based mesh segmentation. Similarly, [BGLSA18] performs a FCN based network for 2d image semantic segmentation and back project the labels to original large scale urban scene point cloud. These multi-view based methods indeed benefit from CNN for feature learning. However, there is a potential loss of 3D information during the rendering procedure from 3D space to 2D images, which brings obstacles for CNN to recover the lost geometric information.

Voxels. In the early stage of applying deep learning techniques to 3D data, forerunners usually converted 3D data into voxelized occupancy grids as intermediate representation [MS15]. In this case, it is simple to employ 3D-CNN for voxel feature learning and train the netwok in an end-to-end mode. [WSK+15] promoted a 3D-CNN based framework for object category recognition and shape completion. In [QSN+16], the authors exploited two distinct network architectures of volumetric CNNs to improve the performance of both voxel based and multi-view based approaches. More recently, [HSL+17] designed a multi-scale 3D-CNN network for large scale urban scene segmentation. Similarly, [TCA+17] involved a 3D-FCN model for voxel-level prediction and post-process the predictions with methods proposed in [ZJRP+15]. Very recently, [DN18] designed a joint 2D-3D network to first analyze multi-view RGB images and then map the features back to volumetric grid of input 3D scene. This joint 2D-3D method incorporate both RGB features and geometric features and yield more accurate prediction result for each voxel. All of these frameworks produced promising segmentation results on large scale dataset. In addition, [ZSN+17, DBI18] extracted local 3D volumetric patches and learned local geometric descriptor for characterizing correspondences between 3D point cloud. However, all these voxel-based

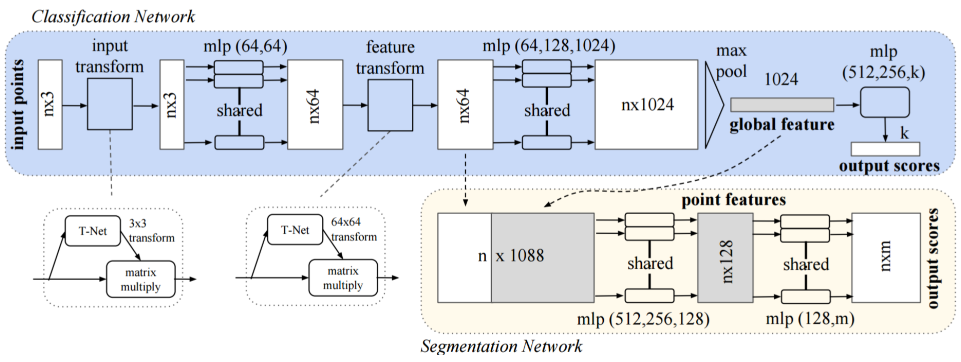


Figure 2.1: PointNet diagram. [QSMG17] designed the first deep neural network for raw point cloud feature learning. The intuition behind is to stack a sequence of basic operations to construct an order-invariant model. The proposed architecture processes each point independently using MLP with shared parameters and aggregates them into global features through max pooling. The learned features are then fed into two branches for different task: classification and segmentation. Image courtesy of [QSMG17].

methods suffer from the computational memory issues. The resolution of grids at each dimension is limited to less one hundred while losing tremendous 3D information. In this case, many works focused on reducing the computational burden caused by sparsity of grid occupancy by employing more intelligent data structure [RUG17, KL17].

Non-Euclidean data. Another attempt of applying deep learning techniques to 3D data is called *geometric deep learning*. Unlike traditional deep learning methods that are employed on grid-structured data, i.e. image, voxel, this kind of methods aim at exploiting geometric information directly from non-Euclidean domains, such as graphs and manifold meshes [SK17]. [BZSL13] introduced a generalized convolutions methods applied to Graph Laplacian known as *spectral networks*. Then, [HBL15] extends this method by incorporating a Graph Estimation process that decreases the learning complexity. Another idea proposed by [MBBV15] attempted to project manifold data to local geodesic system that are analogous to "patches" in image. Feature descriptors are then learned by feeding each patch to a series of filtering operators, which achieves good performance in shape description, retrieval and correspondence. More recently, [MGA$^+$17] used a global parametrization approach to map sphere-type shapes to flat-trous, which defines a translation-invariant convolution in local part.

Point clouds. Different with structured and ordered image or voxel

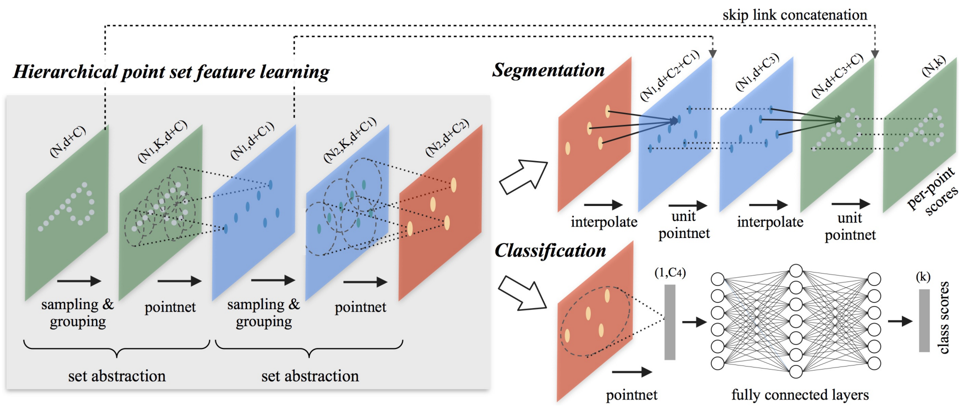


Figure 2.2: PointNet++ diagram. [QYSG17] extended their previous work using a hierarchical architecture by aggregating pointwise features within each local region. The grouped features are propagated to original points through a upsampling operation. Image courtesy of [QYSG17].

representation, 3D point cloud is in general unordered and unstructured. This obstacle blocks the path to employ CNN directly on raw point cloud analysis. The main concern behind is how to design an order-invariant and differentiable feature extraction operator that can be trained end-to-end. Recently, [QSMG17] proposed a simple but powerful neural network composed of a stack of basic operators that can handle unordered raw point cloud directly. The main idea is to process each point independently with a sequence of *multi layer perceptrons* (MLP) that shared weights for all points. The learned pointwise features are either aggregated into a global feature for classification task or used for point-level semantic segmentation task (see Figure 2.1). The baseline PointNet model ignores the intersection relationship between points. Thus, many works focused on learning richer pointwise features by incorporating local dependencies of each point in its local neighborhood [QYSG17, HWN18, WSL+18, SJS+18, LS18] (Figure 2.2 presents the architecture of an extended work called PointNet++). All of these methods achieves better performance on various 3D classification and semantic segmentation datasets than PointNet. Meanwhile, PointNet also serves as a general pointwise feature extraction tool for other tasks, including 3D object detection [QLW+17], point cloud upsampling [YLF+18], instance segmentation [WYHN18] and 3D reconstruction [GFK+18]. However, the receptive field of each point in the 3D scene has not been widely studied in the literatures. None of these methods really deal with extracting suitable global contextual information to enrich pointwise feature.

Multi-sensors fusion. With the development of data acquisition techniques, certain complex applications like self-driving have ability to collecting various data from multi sensors, i.e. RGB cameras, radar and LiDAR. This system requires a framework to fuse the information gathered from 2D and 3D data and has been widely studied for 3D object detection. Some approaches [CMW$^+$17, KML$^+$17] first projects 3D LiDAR point clouds into Front View (FV) or Bird's-Eye View (BEV) images. Then they apply 2D convolutional operations on those 2D representations as well as camera images. The network then merge the region-wise features at an intermediate layer through element-wise concatenation and jointly predicts object class and 3D oriented box regression. Those methods produces promising results on real world data while still suffering from 3D information loss. More recently, [LWYU18] exploit a continuous fusion layer to learn how to project 2D image features onto BEV feature maps and fuse them more accurately.

2.2 Shape detection from 3D data

The automated detection of geometric shapes from 3D measurement data is an instance of the general problem of fitting mathematical models to data. Typical geometric primitives include planes, cylinders, spheres etc. The survey proposed in [KAZB18] presented a detailed review of existing 3D shape detection algorithms and we follow their taxonomy pattern in this section to provide an overview of related works.

2.2.1 RANSAC

Random sample consensus, known as RANSAC, firstly proposed by [FB81], has been widely used in various tasks of computer vision and computer graphics. The basic idea is randomly selecting samples to fit mathematical models, i.e. lines in 2D image or planes in 3D point cloud. Then we choose the one with best fitting ability to the data. In presence of outliers, RANSAC-based algorithms typically perform the best among all kinds of methods. For shape detection, the goal is not to find the best shape, but to output a set of shapes each satisfying certain checking criteria. The typical fitting ability of each shape is measured by counting the number of allocated inliers meeting some geometric hypothesis, i.e. normal deviation and Euclidean distance. [SWK07] introduced an effective iterative RANSAC-based approach to detect several kinds of 3D primitives from unorganized point cloud. They also make the software available for the whole community. Detected shapes construct a compact abstraction of original data and provide an access to extracting higher-level features. [SWWK08] extended their previous work by constructing a topology graph capturing the proximity relationship between each pair of shapes. Such information plays an important role for downstream applications, i.e. surface reconstruction [LA13] and object recognition [OLA16a].

2.2.2 Accumulation space

The basic idea of accumulation space methods is that inliers of expected geometric primitives in Euclidean space are supposed to be close to each other in parameter space. Thus, the basic pipeline firstly embeds original data onto parameter space, and then clusters the embedded points into various groups considered as detected shapes. The Hough Transform [Hou62] is the most popular accumulation space method and has been applied to detect simple shapes in 2D images as lines and curves [DH72]. After that, [HSSM14] and [RVDH05] proposed efficient approaches to detect planes and cylinders respectively from point clouds. However, the main obstacle of those methods is a lack of boundary in parameter space, which brings burden to computational resources. Two extended Hough Transforms are designed by [WPM$^+$14] to solve the computational memory issue by exploiting the sparsity of the parameter space and detecting 3D shapes in a more accurate way. Another accumulation space method is known as Gaussian sphere mapping. [CC08] detected planes by grouping projections of oriented points on the gaussian sphere. Similarly, [QZN14] proposed a framework to detect cylinders from complex industrial areas. Normal vectors of each point are projected onto a unit sphere and thus points of cylinder in Euclidean space preserve a ring shape on the surface of gaussian sphere. This observation helps segmenting cylinder points into different groups according to the corresponding principal axis direction. Then points within each cluster are projected onto the orthogonal plane where each circular pattern can be detected through displacement to plane center.

2.2.3 Region growing

Region growing is considered as another popular kind of methods to detect shapes from 2D or 3D data. Different with RANSAC and accumulation space methods, region growing is better at extracting geometric components that are connected. This method iteratively fits a primitive to a seed point and emit certain geometric hypothesis to points inside its local neighborhoods. Parameters of fitting primitive are updated while propagating the inliers and the final primitive is detected until fitting conditions are no longer valid. Afterwards, a validity checking criterion is performed to decide whether to keep this detected primitive or not. At the beginning, region growing was applied for image segmentation by grouping the pixel with similar color intensity together [TB97]. Afterwards, [RvDHV06] introduced an approach for 3D shape detection from point cloud, which is very efficient when input data is relatively clean. Such method can also be used to detect complex primitives for large scale urban modeling [LM12]. Another advantage of region growing is that some higher-level topology information like adjacency graph can be automatically constructed while propagating the hypothesis to local neighbors.

2.2.4 Shape regularization

Intuitively, primitives detected from man-made objects exhibit some meaningful geometric relationships between each other, such as parallelism, coplanarity, symmetry and orthogonality. Recent research makes an effort to regularizing detected primitives according to such relationships. [LWC+11] proposed a method known as *Globfit* that iteratively fit data to primitives and regularize them through a constrained optimization approach. Differently, [OLA16b] detected those regularities through a hierarchical approach. The whole primitive configuration is then reinforced by performing those regularizations. Shape regularization can also be formalized as a labeling problem by selecting shapes from a finite set of candidates. [MMBM15] first detected initial primitives and generated multiple candidate primitives centered at each initial one. Then they encoded different geometric relationships between candidate primitives into a constrained integer programming problem, where the optimum corresponded to a primitive configuration with best shape regularity.

2.3 Surface reconstruction from 3D data

Surface reconstruction from defect-laden data is still one of the most challenging problems and has been widely discussed in past decades. Our review of corresponding literature covers two major kinds of methods: smooth and piecewise-planar surface reconstruction.

2.3.1 Smooth surface reconstruction

Smooth surface reconstruction aims at recovering a quasi-continuous surface via either implicit or explicit methods.

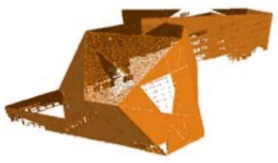
Implicit methods. Poisson Surface Reconstruction method [KBH06] and its extension version [KH13] are considered as the most popular implicit tools to create watertight mesh from unordered points. These approaches define an implicit function at each point, i.e. signed distance function, and then the final surface is extracted as its zero iso-surface. KinectFusion designed by [IKH$^+$11] is a real-time 3D reconstruction framework with RGB-D frames as input. This method first integrates the data into a volumetric representation based on [CL96] and then extracts the surface as zero-crossing where the values of truncated signed distance functions change sign. Another traditional but still powerful method is to extract a polygonal mesh from voxel grids called Marching cube [LC87], which was first applied to surface reconstruction from medical images. Implicit methods are effective but some of them require more input attributes such as normal vectors. Yet, most of these methods are widely used in real world applications.

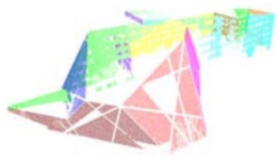
Explicit methods. Instead of constructing an implicit function, explicit methods formulate the surface reconstruction as a binary labeling problem. More precise, the 3D space is first divided into a set of volumetric cells and we assign each cell as *inside* or *outside* of the object. The final surface is extracted as the incident facets of two adjacent cells with different labels. [LPK09b, LPK07] split the 3D space by generating the Delaunay Triangulation of points and compute the surface visibility via lines of sight as data term of each tetrahedra. The final surface solution favors high quality mesh by imposing certain "soft" constraints as a regularization term. Another popular volumetric cell is a voxelized grid, which is widely used in MVS surface reconstruction systems [VTC05, FCSS09]. Such methods generate promising results while sampling points are dense and in presence of noise.

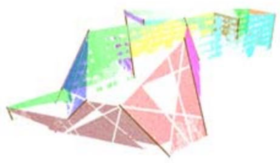
2.3.2 Primitive-based surface reconstruction

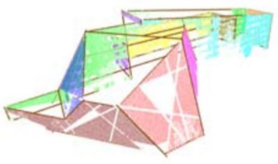
Unlike objects that can be approximated by curve surface, many urban environments or man-made objects preserve higher-level geometric regularities. A better way is to represent such scenes by polyhedral meshes such that each facet corresponds to a large polygon. Two steps are involved in this solution: (i) detect planar primitives from original 3D data (ii) assemble them as the final mesh. We mainly discuss the literature incorporating primitive assembling problem.

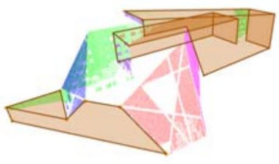
Connectivity-based methods. These methods assemble detected primitives by reasoning on proximity and structural considerations. Analyzing a connectivity graph to detect and link points intersecting plane triples


(a) Scanned data.


(b) Points are clustered and representative planes are fitted. (Section 3.1 and 3.2)


(c) Some of the plane intersections are computed. (Section 3.3)


(d) All the planes and intersections are recovered. (Section 4)


(e) Some of the faces of the target polyhedron are extracted. (Section 5)

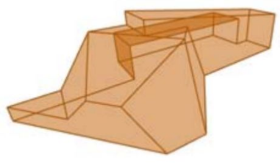

(f) User intervention incorporated, the final model is reconstructed.

Figure 2.3: Pipeline of connectivity-based method [CC08]. This framework relies on clustering points into planes (b) and then computing their intersection relationships (c,d). Those information permits users to recover certain planes' boundary polygon (e). User intersection is finally involved to complete the boundary detection of the left primitives. This method is very efficient for clean data but less robust in presence of noise. Moreover, user intersection increases the complexity to use. Image courtesy of [CC08].

[CSAD04, CC08, SFF11, vKvLV11] usually works well when the correct connectivity between primitives can be recovered (see Figure 2.3). To be robust to challenging data, one interactive solution is to automatically snap primitives when the connectivity is obvious, and let the user complete the output surface for the conflicting situations [ASF$^+$13]. Another solution consists in mixing polyhedral surface components with flexible free-form patches [LA13, LPK09a]. Such a representation however does not offer the level of compactness and simplicity of pure polyhedral surfaces. Despite being fast, connectivity-based methods suffer from a lack of robustness to defect-laden data, in particular to over- and under-detection of primitives and erroneous connections between primitives. Our approach exploits some principles of these methods as a preliminary step to quickly solves obvious plane assembling situations: it allows us to lighten the time-consuming slicing operations.

Slicing-based methods. The core of these methods consists in partitioning a 3D domain by extending primitives. The partitioning datastructure is typically a 3D tesselation made of polyhedral cells, which are themselves composed of polygonal facets. The output surface is then ex-

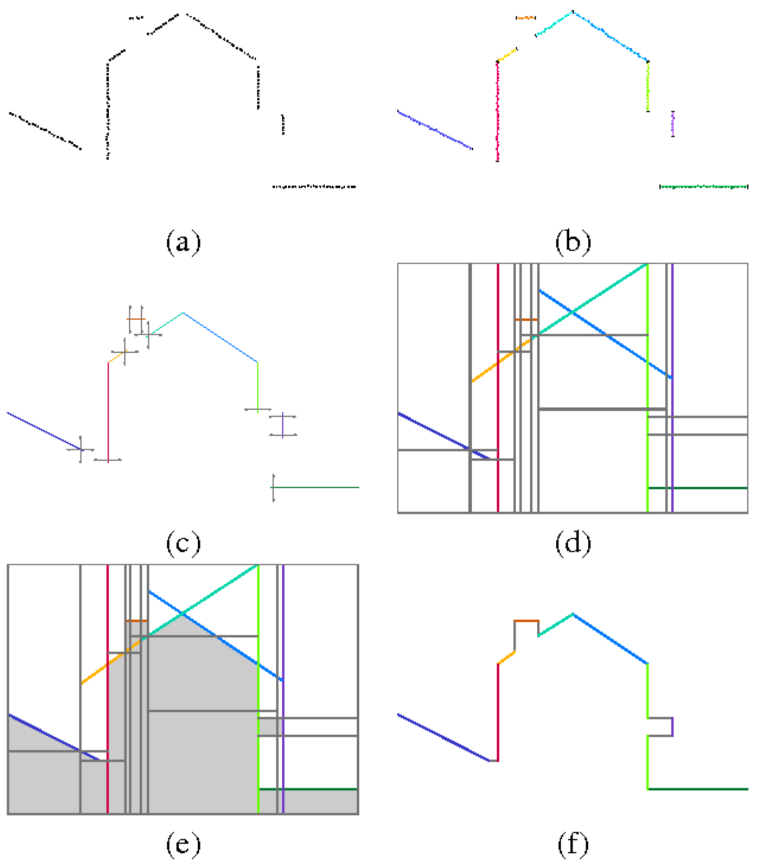


Figure 2.4: Pipeline of slicing-based method [CLP10]. Planes and ghost primitives (b,c) are first detected from point clouds (a). Then 3D domain is split into several volumetric cells by slicing those planes (d). The final surface mesh (f) is extracted by labeling each cell as *inside* or *outside* of the object. Such method is more robust in presence of artifacts but less efficient while slicing large number of planes. Image courtesy of [CLP10].

tracted by selecting a subset of facets from the tesselation. Because each primitive naively slices all the others, such a data-structure is particularly dense and time-consuming to compute. Some methods decompose the slicing operations into spatial blocks [CLP10, BdLGM14] (see Figure 2.4). Such piecewise partitions increase scalability by a reasonable margin, but blocks do necessarily align well with data. These methods also add artificial primitives along vertical and horizontal axes in the partition to be more robust to missing primitives, assuming the observed object aligns with these arbitrary directions. A discrete partitioning [SDK09, VLA15] that avoids computing the exact geometry of the whole partition is a less costly option, but typically engenders geometric artifacts when the discretization is not fine enough. An-

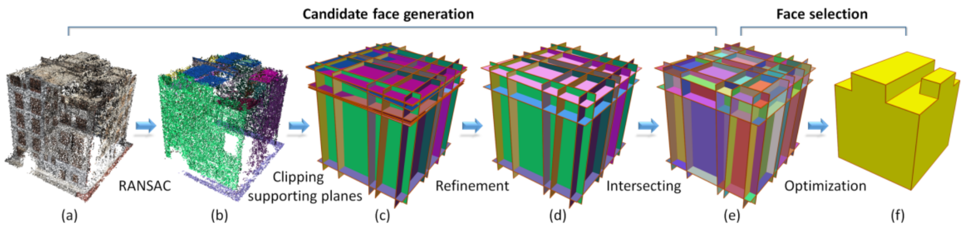


Figure 2.5: Pipeline of slicing-based method [NW17]. Planes (b) are first detected from point clouds (a) by RANSAC and then refined (d) by supporting planes (c). Next, they compute the intersections between refined planes and generate a set of candidate faces (e). Finally, surface extracted as a subset of candidate facets formulated as an integer programming (f). This framework provides convincing and high quality polygonal mesh but still suffering the computational constraint while detecting intersections. Image courtesy of [NW17].

other possible solution consists in filtering and simplifying the input set of primitives to remove redundant planes and reduce the computational burden of the slicing operations [NW17] (see Figure 2.5). Primitives can also be parsed with domain-specific knowledge [TMT10]. All these methods offer a good robustness to imperfect configurations of primitives, but exhibit a limited scalability due to the lack of flexibility of their partitioning data-structures. Our approach proposes two key ingredients to solve the scalability issue: a new light and spatially-adaptive partitioning data-structure and a preliminary connectivity analysis that reduces the number of primitives to be processed during slicing operations.

Methods with geometric assumptions. Some works also exploit strong geometric assumptions. The Manhattan-World assumption [CY00] enforces planes to follow only three orthogonal directions. This assumption reduces both the geometry of output 3D models and the solution space to explore. Such an assumption is interesting for modeling some buildings [LWN16] and approximating shapes very coarsely [HJS$^+$14]. Geometric regularities as parallelism and symmetry are also popular for reconstructing man-made objects. Such considerations bring robustness to the connectivity analysis of primitives [HK12, ZN12]. Another frequent geometric assumption is to restrict the output surface to have a disk-topology with a 2.5D view-dependent representation. This is well adapted to reconstruct buildings from airborne data [VKH06, ZBKB08, PY09, LM11], facades from street-side data [BSVG15], indoor scenes from images [CF14] and piecewise planar depth maps from multi-view stereo images [SSS09, GFP10]. Note also that some works assume observed objects are likely to be found in large CAD databases and indirectly reconstruct them by solving a recognition problem [ISS17]. Although these assumptions efficiently reduce the solution space in general, methods exploiting them are restricted to specific applications. To the contrary, our approach does not require such application-specific assumptions.

CHAPTER 3

Semantic segmentation of 3D data

3.1 Introduction

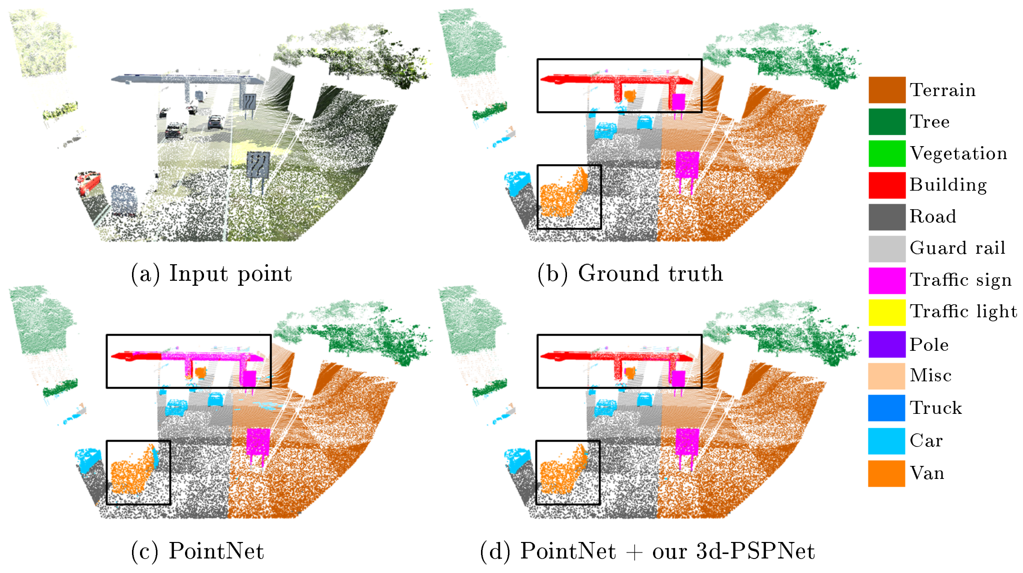


Figure 3.1: Semantic segmentation results on vKITTI dataset with and without our 3d-PSPNet. Given input point cloud of a complex scene (a), PointNet [QSMG17] baseline provides accurate prediction label for most of the points (c). However, compared with ground truth (c), it fails to predict correct labels for points of large size objects (see points in black rectangles). PointNet baseline equipped with our 3d-PSPNet improves prediction results by enriching global contextual information (d).

Analyzing and extracting geometric features from 3D data is a fundamental step for complex 3D scene understanding. Recent methods show the efficiency and possibility of deep learning techniques employed directly on raw point cloud, without transferring it into intermediate 3D representations. However, the use of contextual information in complex 3D scene has not been widely studied in the literature. In this chapter, we propose a 3D pyramid module to enrich pointwise features with multi-scale contextual information, which is inspired by global feature aggregation methods exploited for 2D-image scene understanding [ZSQ+17]. We evaluate our method on three large scale datasets with two baseline models. Experimental results show that the enriched features bring significant improvements for both indoor and outdoor scene semantic segmentation tasks (see Figure 5.1 as an example).

The goal of our chapter is to enlarge the receptive field of points by incorporating multi-scale contextual information in sub-regions. To do so, we build a generic 3d-PSPNet module that can be concatenated after any state-of-the-art pointwise feature learning approach. Our chapter is organized as follows: Section 3.2 presents the architecture of 3d-PSPNet. We analyze the pointwise features obtained with and without our 3d-PSPNet in Section 3.3. To demonstrate the performance of our method, we apply 3d-PSPNet to three large scale datasets with two baseline models, and the experimental results are shown in Section 3.4.

We start from a point cloud $P = \{p_1, p_2, \dots, p_n\}$, where each point $p_i \in \mathbb{R}^c$, c is the number of input features for each point, i.e. position, color, normal etc. By utilizing recent deep neural point cloud feature learning networks [QSMG17, QYSG17], we can obtain pointwise features $F = \{f(p_1), f(p_2), \dots, f(p_n)\}$, where $f(p_i) \in \mathbb{R}^{f_1}$ is an f_1-dimensional feature vector. Our objective is to capture reasonable global contextual clues for each point and return enriched pointwise features $\hat{F} = \{\hat{f}(p_1), \hat{f}(p_2), \dots, \hat{f}(p_n)\}$. Inspired by [ZSQ+17], an alternative is to exploit global contextual features in several scale sub-regions. Figure 3.2 illustrates the diagram of the network.

3.2 Methodology

Our 3d-PSPNet exhibits a pyramid structure. At each pyramid scale l, we capture contextual clue by 3 basic operations.

- **Grid pooling.** Given input points $P = \{p_1, p_2, \dots, p_n\}$ and pointwise features $F = \{f(p_1), f(p_2), \dots, f(p_n)\}$ returned by PointNet or PointNet++, this step projects each point to a local sub-region. More specifically, we first split the whole scene into $2^{l-1} \times 2^{l-1} \times 2^{l-1}$ voxelized cells, each preserves the same size. After that, all points are grouped into the corresponding grid according to their spatial position in the scene.

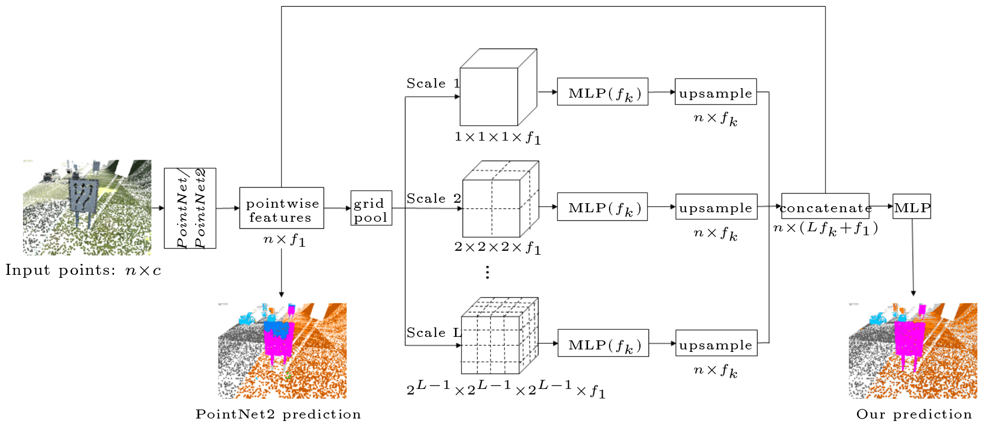


Figure 3.2: Diagram of our 3d-PSPNet. Given the point cloud of a complex scene, our framework uses PointNet [QSMG17] or PointNet2 [QYSG17] (for clarity issue, we replace PointNet++ by PointNet2 in the following sections) as baseline model to first capture pointwise local features. After that, a pyramid structure model is employed to exploit multi-scale global contextual features at each sub-regions. Finally, the input local pointwise features and learned multi-scale contextual features are concatenated together. The enriched features are capable to produce better semantic segmentation results than using baseline model only (see the segmentation results on urban scene. PointNet2 predicted accurate labels for most of the points, but fails at the top part of *traffic-sign*. Using our 3d-PSPNet, the misslabeled points are corrected).

This basic operation enables each point to exploit contextual information in its sub-region independently. Note that we keep record of cell index where each point is projected to as $G = \{g(p_1), g(p_2), \ldots, g(p_n)\}$ for further processing. Finally, to capture the contextual clue in every grid, a basic max pooling layer is employed on all the points in grid (i, j, k) and we obtain a f_1-dimensional global feature vector f^l_{ijk}. Our grid pooling layer outputs a $2^{l-1}\times 2^{l-1}\times 2^{l-1}\times f_1$ tensor at scale l.

Another choice of grouping the points into different 3D grids relies on the use of *sampling layer* proposed by [QYSG17]. This fine-to-coarse approach selects a fixed number of most distant points using iterative *farthest point sampling* (FPS). Then, the other points are clustered into the group of selected points according to query ball or kNN method. Compared to the voxelized grid grouping method, this choice requires more computational time to select distant points and preserves unstructured grid shape. However, it solves the sparsity of point cloud, ensuring that each sub-regional grid have some projected points inside. We refer this architecture as *adaptive grid method.* We discuss the comparison between these two architectures in Section 3.4.5.

- **Sub-regional feature aggregation.** Note that our grid pooling operator projects all points and corresponding features onto different sub-regions. Then, we enhance each sub-regional global feature f^l_{ijk} via a sequence of MLP with output channels $(f_2, \dots, f_d)$. Enriched global feature at grid (i, j, k) is now a f_k-dimensional feature vector $\hat{f}^l_{ijk}$. Note that all the cells at each pyramid scale l share the same MLP weights, which preserves the order-invariant property of our network. To sum up, this step outputs a $2^{l-1} \times 2^{l-1} \times 2^{l-1} \times f_d$ tensor at scale l.

- **Grid upsampling.** The previous step provides an enhanced global contextual vector $\hat{f}^l_{ijk}$ in each grid, which serves as a representable clue for all the points inside each sub-region. Our goal is to output an enriched feature for all the points. To do so, we use the point-to-cell assignments, which is noted as G, to upsample all the points in each grid and assign them a global contextual feature of its corresponding grid s.t. $\hat{f}^l(p_i) = \hat{f}^l_{g(p_i)}$. Finally, this step outputs the enhanced sub-regional contextual pointwise feature $\hat{F}^l = \{\hat{f}^l(p_1), \dots, \hat{f}^l(p_n)\}$.

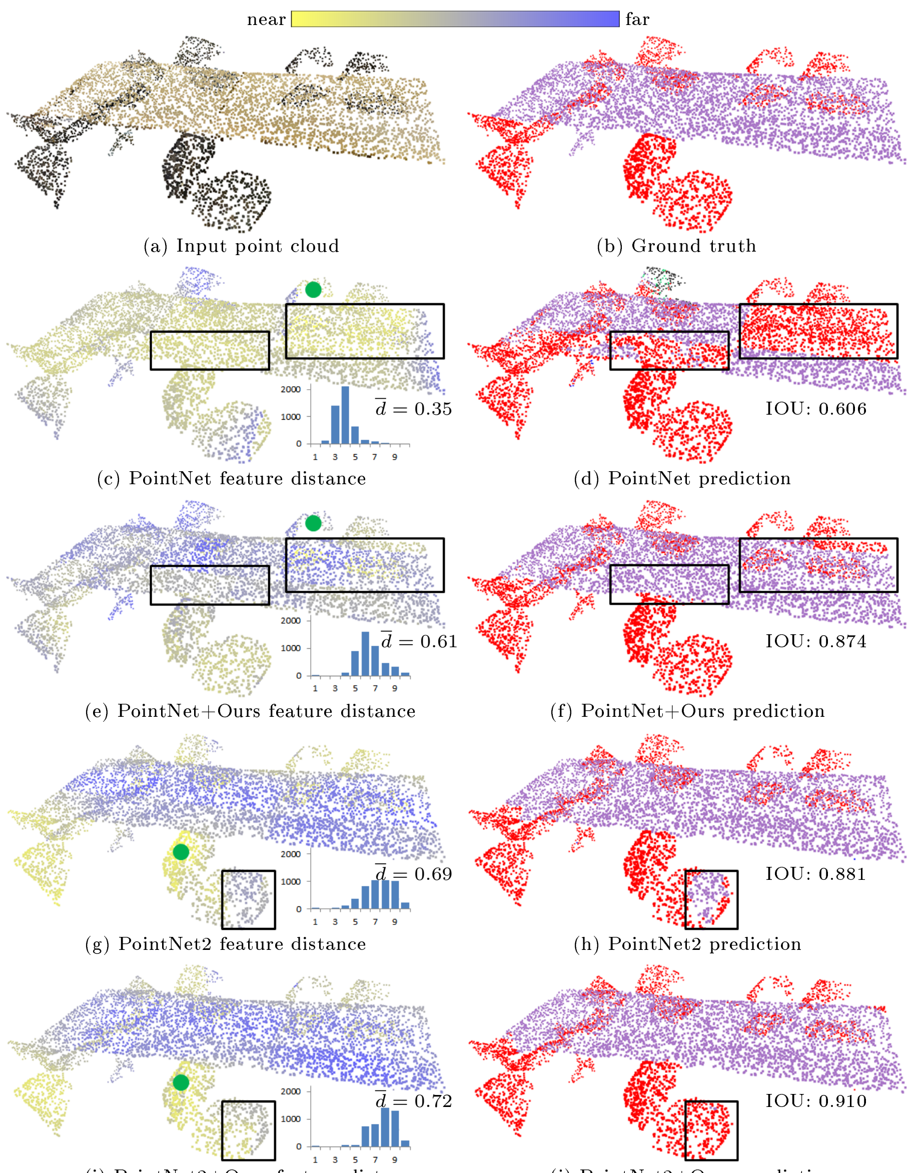
Next, we group enriched pointwise features of all pyramid scales together as multi-scale contextual features $\hat{F}^C = \hat{F}^1 \oplus \hat{F}^2 \oplus \cdots \oplus \hat{F}^L$, where $\oplus$ is the concatenation operator. We argue that the enriched pointwise features $\hat{F}$ capture multi-scale contextual information by aggregating features learned at different sub-regions with varied sizes. Finally, to not lose the pre-learned local information, we assemble contextual and local features together and get the final enriched pointwise features $\hat{F} = \hat{F}^C \oplus F$. Note that all the basic operators exploited in our 3d-PSPNet preserve the order-invariant property of the input point cloud. Since the main application of our 3d-PSPNet is large scale indoor and urban scene semantic segmentation, we exploit a typical *cross-entropy* loss function in the framework.

3.3 Feature analysis

Being equipped with our 3d-PSPNet, the state-of-the-art baseline models, i.e. PointNet and PointNet++, are capable of learning enriched pointwise feature by incorporating both local and multi-scale global information. Figure 3.3 analyzes the quality of learned features while employing our module. Given the point cloud of indoor scene shown in Figure 3.3a, we fed it into four networks: PointNet baseline, PointNet baseline+Ours 3d-PSPNet, PointNet2 (for clarity issue, we replace PointNet++ by PointNet2 in the following sections) baseline and PointNet2 baseline+Ours 3d-PSPNet. We output the features learned at the last layer of each model, and visualize extracted features as Euclidean distance from every point to a standard point with ground truth label *chair* (green spot shown in Figure 3.3c, 3.3e, 3.3g and 3.3i) in feature space. The color of each point varies from yellow to blue, representing a near-to-far feature distance from current point to the selected standard point. Besides, we compute the distribution of feature distance from points with label *table* to standard point with label *chair* (see blue histogram in Figure 3.3c, 3.3e, 3.3g and 3.3i).

As a result, Figure 3.3c shows that most of *table* points have a relatively close distance to *chair* point with a mean distance 0.35. This observation means that features learned by PointNet baseline fail to clearly discriminate points with ground truth label *table* and *chair*. By plugging-in our 3d-PSPNet after PointNet baseline, this mean distance increases to 0.61. In addition, the feature distance distribution shifts from lower bins to higher bins in Figure 3.3e compared with Figure 3.3c. Most points with label *chair* are near to standard point (green spot in Figure 3.3) while points with label *table* are far away from standard point in feature space. Consequently, the *Intersection Over Union* (IOU) criteria improves from 0.606 to 0.874 after employing our 3d-PSPNet (see comparison of prediction results in Figure 3.3d and 3.3f).

In another control experiment, since PointNet2 baseline produces richer local features than PointNet by exploiting a hierarchical approach, the mean feature distance from *table* points to standard point reaches a high value 0.69 (see Figure 3.3g). It indeed achieves more accurate segmentation results with 0.881 IOU. Yet, there are still a small part of points with label *chair* which are relatively far away from standard point in feature space (see points in black rectangle in Figure 3.3g). In addition, the network mislabeled them to *table* (see Figure 3.3h). While concatenating our 3d-PSPNet after PointNet2 baseline, these mislabeled points reach a closer feature distance to standard point with a mean feature distance 0.72 (see Figure 3.3i). Figure 3.3j illustrates a correction of prediction result for these mislabeled points, achieving 0.910 IOU.



(a) Input point cloud (b) Ground truth

(c) PointNet feature distance (d) PointNet prediction

(e) PointNet+Ours feature distance (f) PointNet+Ours prediction

(g) PointNet2 feature distance (h) PointNet2 prediction

(i) PointNet2+Ours feature distance (j) PointNet2+Ours prediction

Figure 3.3: Feature analysis on an indoor scene composed of two types of objects: *chair* (red) and *table* (purple). Given input scene (a), we extract features of output layer learned by each model and visualize the feature distance from each point to a standard point (green spot) with ground truth label *chair*. Models equipped with our 3d-PSPNet not only reduce the feature distance from points with label *table* to standard point (see shift of blue histograms from lower bins to higher bins in (e) and (i) compared with (c) and (g)), but also produces better prediction results than using the baseline only (see chair IOU in (d, f) and (h, j)).

According to previous observations, our 3d-PSPNet enriches pointwise features and improves the final segmentation results. We argue this gain of prediction accuracy comes from capturing appropriate multi-scale contextual information in sub-regions with different size.

3.4 Experiments

We evaluate our approach on three large scale datasets, Stanford Large-Scale 3D Indoor Spaces Dataset (S3DIS) [ASZ+16], ScanNet dataset [DCS+17] and vKITTI dataset [FTAB17], ranging from real world indoor scenes to large scale urban scenes. For each dataset, we compare the qualitative and quantitative results returned by two standard baselines, PointNet and PointNet2, with and without plugging-in our 3d-PSPNet. For a fair comparison, we fix all the parameters and hyperparameters throughout the evaluation procedure, where the only difference being injecting our 3d-PSPNet or not. All the experiments are performed on a NVIDIA GeForce GTX 1080 Ti GPU.

3.4.1 Implementation details.

Unified diagram. There are several hyperparameters involved in our network, i.e. number of pyramid scales L, dimension of feature aggregation fully connected layers MLP $(f_2, \dots, f_k)$, number of grids at each scale etc. Remind that our goal is not to achieve state-of-the-art performance on all the datasets, but to validate that our 3d-PSPNet is a generic module that increases the segmentation accuracy of state-of-the-art 3D neural network. Therefore, we keep all the parameters and hyperparameters constant and only compare the results with and without our module. As a result, we utilize one unified diagram throughout all experiments. First, we observe that the number of pyramid scales L is a trade-off between prediction accuracy of network and computational efficiency. Large L indeed provides more accurate segmentation results, but also increases the whole training time drastically. To balance this trade-off, we use a 4-level pyramid structure in all our experiments. We discuss the choice of L in Section 3.4.5. Second, empirical results illustrate that simply increasing the number of MLP in the sub-regional feature aggregating layer will not provide more promising results. We choose a 256-channel MLP to aggregate the global contextual information in each sub-region. This choice not only avoids bringing numerous parameters to be learned but also prevents the network from overfitting. Batch normalization and Relu activations are involved after each MLP. Finally, we set the number of grids in each dimension (x, y, z) at each scale l

as 2^{l-1} to preserve a multi-scale pyramid structure.

Training strategy. We follow the data-preparing process proposed by [QSMG17] for all datasets. All the individual scenes are first divided into a set of blocks with same size. Then, a fixed number of points are sampled from each block to make the training more efficient. Each block serves as a mini-batch for the end-to-end training. Finally, each trained model is applied to testing blocks for final semantic segmentation evaluation. Again, to illustrate a fair comparison, we adopt all the training details of [QSMG17] and [QYSG17]. More specifically, we use Adam optimizer with initial learning rate 0.001. The learning rate is divided by 2 every 300000 mini batches for S3DIS and ScanNet, 200000 mini batches for vKITTI. More details can be found in each following subsection.

Evaluation metrics. We evaluate the prediction results by both qualitative and quantitative comparisons. For quantitative measurement, we use three standard semantic segmentation evaluation metrics to validate: *Overall* Accuracy (OA), *mean Interscetion Over Union* (mIOU) and *mean Accuracy Over Classes* (mAcc). Readers can find exact formulation for these metrics from [TCA+17].

3.4.2 Semantic Segmentation on the S3DIS Dataset

We first evaluate our 3d-PSPNet on S3DIS dataset. The whole dataset contains 6 Areas including 271 rooms with 13 classes. Each point has an annotation label from 13 classes. As proposed in [QSMG17], in the training procedure, each room is split into blocks with size $1m\times 1m$ in direction x and y with stride $0.5m$. To make batch training available and accelerate the training process, we sample 4096 points in each block. Every sampled point contains 9 dimensional channels: $[x, y, z, r, g, b, \overline{x}, \overline{y}, \overline{z}]$, representing position, color and normalized position of each point in the current room. In the testing process, all rooms are split into non-overlapping blocks with size $1m\times 1m$. Following the 1-fold experimental protocol described in [ASZ+16], we test on Area 5 and train on the other Areas.

We manipulate two control experiments to validate of our 3d-PSPNet with PointNet and PointNet2 baseline. For PointNet baseline, we set batch size as 24 and assign each point a 9-dimensional input feature $[x, y, z, r, g, b, \overline{x}, \overline{y}, \overline{z}]$. For PointNet2 baseline, the batch size is also 24 but we assign a 3-dimensional channel $[x, y, z]$ as input feature for each point. This setting is designed to exhibit the generalization of our 3d-PSPNet in case of insufficient input features. No data augmentation technique has been employed in both experiments. Note again that all the hyperparameters are fixed in the two control experiments.

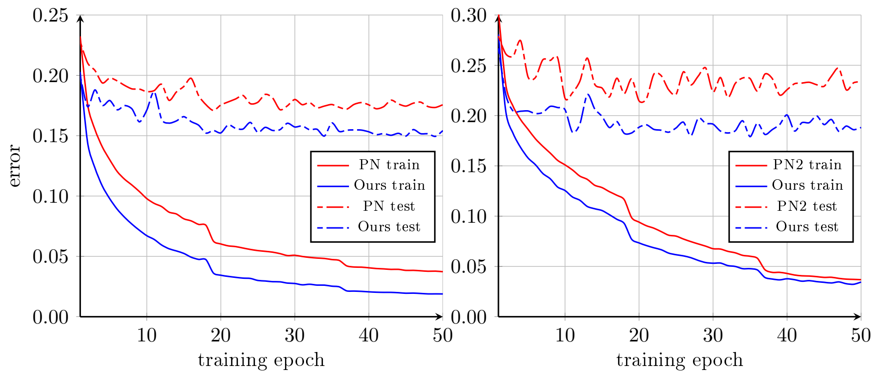


Figure 3.4: Training error and testing error on S3DIS dataset Area 5 of PointNet (abbreviated as PN, left image) and PointNet2 (abbreviated as PN2, right image) baseline with and without our 3d-PSPNet learned from scratch. Note that our 3d-PSPNet improves 2.27% and 3.78% OA for PointNet and PointNet2 respectively.

Method	mIOU	mAcc	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PointNet	41.02	48.51	89.41	**98.35**	69.14	0.04	**5.54**	45.20	**11.56**	58.69	53.21	2.90	42.63	23.09	33.47
PointNet+Ours	**45.54**	**54.08**	**92.05**	97.59	**70.60**	**0.43**	5.04	**49.83**	7.73	**64.82**	**68.71**	**11.36**	**47.28**	**38.35**	**38.27**
PointNet2	43.11	53.39	71.80	74.75	69.35	0.00	11.70	22.15	42.92	59.93	75.71	22.63	**51.74**	17.51	40.23
PointNet2+Ours	**48.07**	**58.21**	**79.99**	**84.15**	**73.32**	0.00	**20.21**	**32.69**	**50.25**	**62.02**	**78.25**	**31.02**	51.24	**21.04**	**40.68**

Table 3.1: Quantitative results on S3DIS dataset Area 5, including mIOU, mAcc and IOU for 13 classes.

Two baseline models with and without our 3d-PSPNet are trained from scratch for 50 epochs. We plot training and testing errors on Area 5 along all epoch for these 4 models to validate 3d-PSPNet. As shown in Figure 3.4, the gap between the baseline curve and our curve means that our 3d-PSPNet improves both training and testing accuracy along the whole training procedure. Although there is an oscillation along the curves of testing accuracy, our model finally converges to a optimal with lower testing error, which increases the generalization of two baseline models. Figure 3.5 shows visualization results on two indoor scenes. PointNet and PointNet2 baseline provides accurate prediction for most of the points. Our 3d-PSPNet succeeds in correcting prediction results for part of the mislabeled points, i.e. *window* and *table*. In addition, the quantitative segmentation measurement on S3DIS dataset Area 5 is given in Table 3.1. Our 3d-PSPNet improves 4.52% mIOU and 5.57% mAcc for PointNet baseline, 4.96% mIOU and 4.82% mAcc for

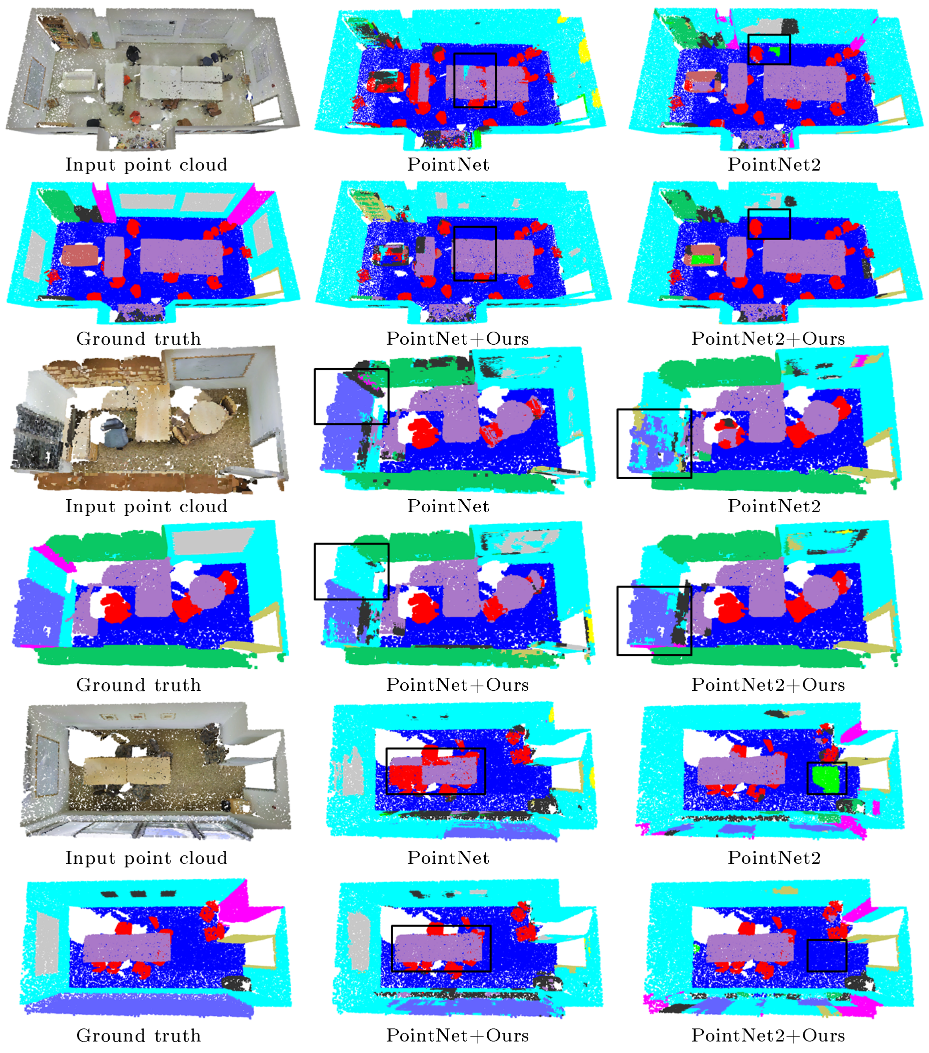


Figure 3.5: Qualitative results on S3DIS dataset. Our 3d-PSPNet produces better prediction results than using baseline model only (see the parts marked by black boxes).

PointNet2 baseline. Besides, IOU of 10 and 11 out of 13 classes are improved for PointNet and PointNet2 baseline respectively. According to qualitative and quantitative comparisons, our 3d-PSPNet indeed reinforces the generalization of baseline models, which thanks to the enrichment of multi-scale contextual feature captured in different sub-regions.

3.4.3 Semantic Segmentation on the ScanNet Dataset

We next evaluate our 3d-PSPNet on the ScanNet dataset. This large scale indoor dataset contains 1201 training rooms and 312 testing rooms with 21 classes including *unannotated* class. In the training procedure, unlike S3DIS dataset, this time we split each room into blocks of size $1m \times 1m$ in direction x and y with stride $1m$ instead of $0.5m$, where each block contains 4096 sampling points. This choice avoids consuming too much training time on large dataset. Again, every sample point contains 9-dimensional channel: $[x, y, z, r, g, b, \overline{x}, \overline{y}, \overline{z}]$ as defined in last Section. For testing, we apply the trained model on testing blocks with size $1m \times 1m$.

We follow the same experiment settings employed in Section 3.4.2 by concatenating our 3d-PSPNet after PointNet and PointNet2 baselines. However, considering the enormous size of training data of ScanNet dataset, we exploit a *fine tuning* strategy on the training phase to improve the efficiency of training. In practice, we first train the PointNet and PointNet2 baselines for 20 epochs and stop. To perform a fair comparison, we use the pre-trained models to initialize the weights of parameters in the network and continue the experiments along two different paths. First, we concatenate our 3d-PSPNet after the pre-trained models and fine tune the whole network for 20 epochs. Second, we continue training the pre-trained models without our 3d-PSPNet for 20 epochs.

Figure 3.7 exhibits the comparison of training and testing errors along the last 20 training epochs between these two control experiments. Utilizing PointNet and PointNet2 models only, the training curves gradually converge to the final optimal while testing curves increase along training process. This phenomenon is caused by overfitting of baseline models to the training set. However, plugging-in our 3d-PSPNet after baseline models improves the testing accuracy from first epoch of fine-tuning and converges in only a few epochs. This observation proves that our 3d-PSPNet raises the generalization of baseline model. For PointNet and PointNet2, our module improves OA by 1.26% and 2.19% respectively, including *unannotated* class. Figure 3.6 shows some qualitative results. Note that to perform a fair visualization

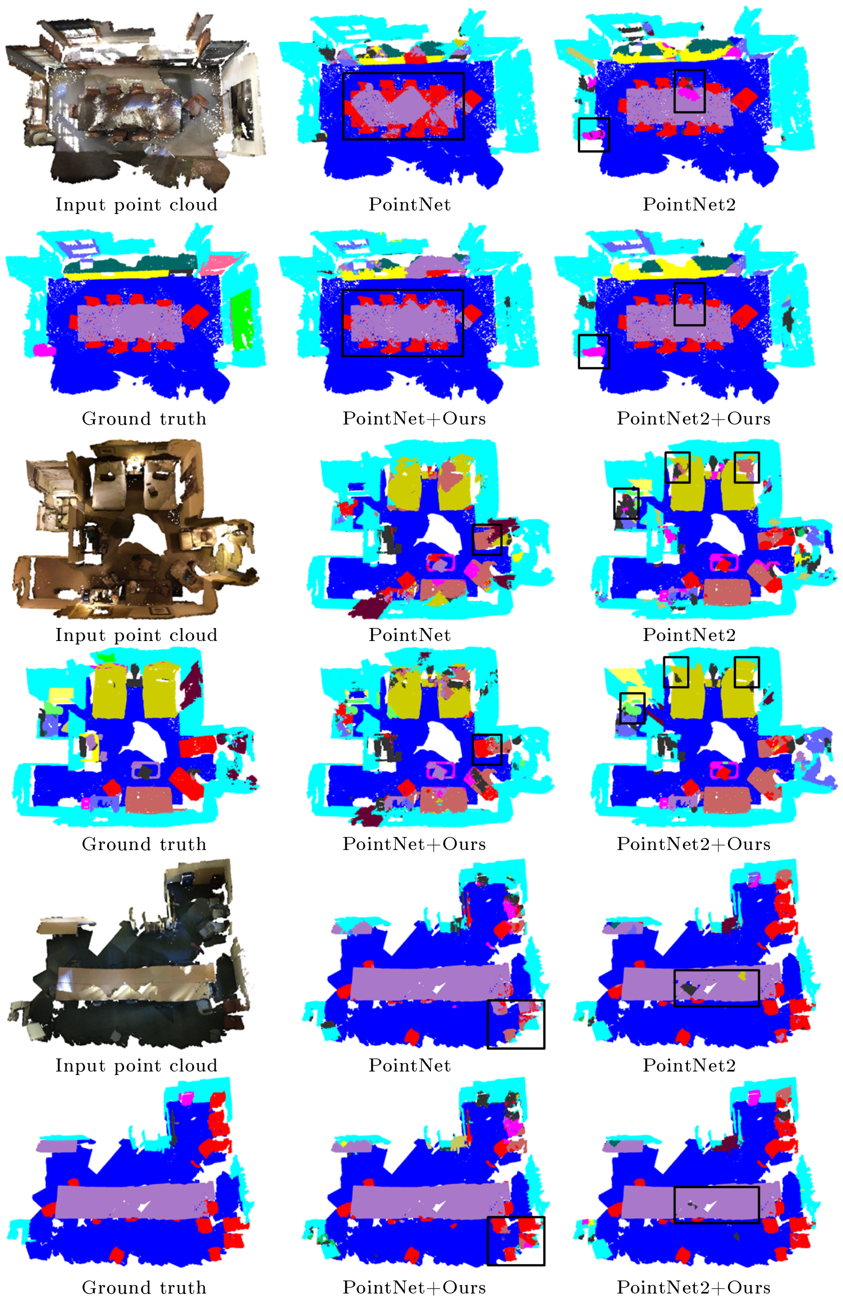


Figure 3.6: Qualitative results on ScanNet dataset. Our 3d-PSPNet improves the prediction results returned by baseline models via a fine tuning strategy (see changes in black boxes).

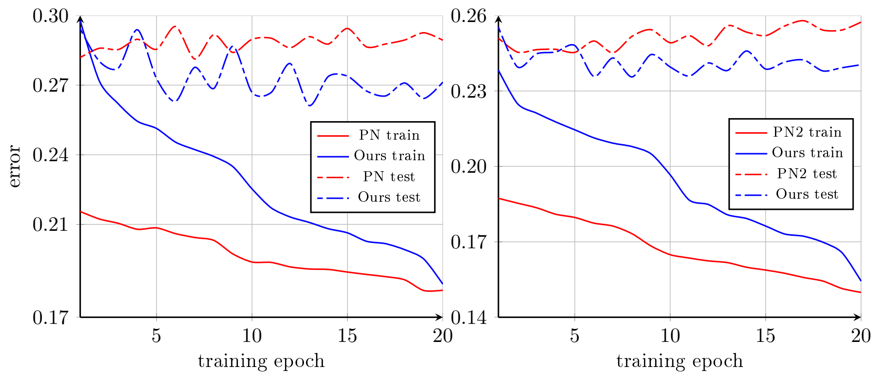


Figure 3.7: Training and testing error of PointNet (abbreviated as PN, left image) and PointNet2 (abbreviated as PN2, right image) baseline with and without our 3d-PSPNet on ScanNet dataset. Equipped with our 3d-PSPNet, the network improves OA by 1.26% and 2.19% for PointNet and PointNet2 respectively.

Method	mIOU	mAcc	wall	floor	chair	table	desk	bed	book-shelf	sofa	sink
PointNet	23.64	33.25	67.48	87.70	41.31	40.48	14.83	32.85	19.73	28.12	14.86
PointNet + ours	**26.81**	**38.27**	**69.33**	**89.38**	**44.24**	**44.25**	**16.34**	**35.85**	**27.47**	**29.88**	**19.64**
PointNet2	30.98	42.40	**71.55**	87.59	57.17	45.94	14.66	40.83	31.92	42.52	17.61
PointNet2 + ours	**33.02**	**48.20**	71.42	**88.75**	**57.67**	**48.09**	**18.87**	**41.91**	**36.77**	**46.53**	**21.68**
Method	bathtub	toilet	curtain	counter	door	window	shower curtain	refrid-gerator	picture	cabinet	other furniture
PointNet	24.19	22.27	8.49	11.64	12.05	9.02	4.31	**7.98**	0.87	17.02	**7.53**
PointNet + ours	**36.63**	**28.54**	**10.55**	**13.45**	**13.01**	**12.39**	**9.53**	7.90	**3.58**	**17.89**	6.44
PointNet2	**49.15**	**33.06**	25.72	14.54	13.37	8.07	18.05	15.85	**0.76**	18.35	12.85
PointNet2 + ours	48.27	32.51	**26.87**	**17.81**	**14.53**	**9.75**	**18.54**	**26.01**	0.47	**20.81**	**13.09**

Table 3.2: Quantitative results on the ScanNet Dataset, including mIOU, mAcc and IOU for 20 classes. Noted that *unannotated* class is not considered in this evaluation.

comparison, we show the results given by baseline models trained after first 20 epochs (the continued trained model is overfitting). Our 3d-PSPNet corrects some mislabeled labels returned by the baseline model and makes more consistent prediction results (see points in block rectangles). This modification benefits from fine-tuning training strategy where mislabeled points incorporate pyramid contextual information from its local regions. Table 3.2 presents the quantitative results on testing rooms. Network equipped with our 3d-PSPNet increases 3.17% mIOU and 5.02% mAcc for PointNet baseline, 2.04% mIOU and 5.80% mAcc for PointNet2 baseline. Besides, IOU of 18 and 16 out of 20 classes have been improved for PointNet and PointNet2 baseline respectively. Again, we argue these improvements come from the aggregation of multi-scale contextual feature enriched in different sub-regions.

3.4.4 Semantic Segmentation on the vKITTI Dataset

We finally evaluate our framework on the vKITTI dataset, a real world point cloud dataset obtained by Velodyne LiDAR scanners. The whole dataset contains 6 non-overlapping urban scenes with 13 classes. Unlike rooms contained in indoor scene dataset, outdoor urban scenes usually preserve objects with larger scalability, i.e. cars, buildings etc. Thus, we split each scene into non-overlapping blocks with size $5m \times 5m$ in direction x and y to make sure that the large scale objects are split into fewer number of blocks. Again, we sample 2048 points in each block as the mini training batch. Every sample point contains 9-dimensional channel: $[x, y, z, r, g, b, \overline{x}, \overline{y}, \overline{z}]$ as before. In the testing process, we apply the trained model on all the testing blocks. We follow the 6-fold cross validation protocol described in [FTAB17].

As previous experiments, we perform two pairs of comparisons with PointNet and PointNet2 baseline model. This time, we feed the same input points with 9-dimensional channels to both models for the reason that rgb-color information in urban scene plays an important role to characterize objects. In the training phase, we employed the same fine tuning strategy as proposed in Section 3.4.3. PointNet and PointNet2 baseline models are first trained on vKITTI dataset for 50 epochs. After that, we concatenate our 3d-PSPNEt at the end of the baseline models and fine tune the whole network for another 50 epochs. In the control experiment, we continue training baseline models for 50 epochs for comparison.

Training and testing errors of four experiments for the last 50 epochs are shown in Figure 3.9. The training curves of baseline models already converge to a minimum after the first 50 training epochs. Testing curves of our 3d-PSPNet oscillate in the first fine tuning 35 epochs but preserve a tendency of convergence in the last 15 epochs. Our 3d-PSPNet increases the generalization of baseline models by improving OA by 2.49% and 3.58% for PointNet and PointNet2 respectively. Qualitative comparisons are illustrated in Figure 3.8. PointNet and PointNet2 produced accurate prediction results for most of the points in the urban scene. Our 3d-PSPNet corrected some mislabeling for points of large scale objects, i.e. *building* (red) and *car* (blue). Table 3.3 shows the 6-fold quantitative results on vKITTI dataset. Our 3d-PSPNet improves 2.88% mIOU and 3.27% mAcc with PointNet baseline, 3.95% mIOU and 5.19% mAcc with PointNet2 baseline. Besides, the IOU of 13 and 11 out of 13 classes improves for PointNet and PointNet2

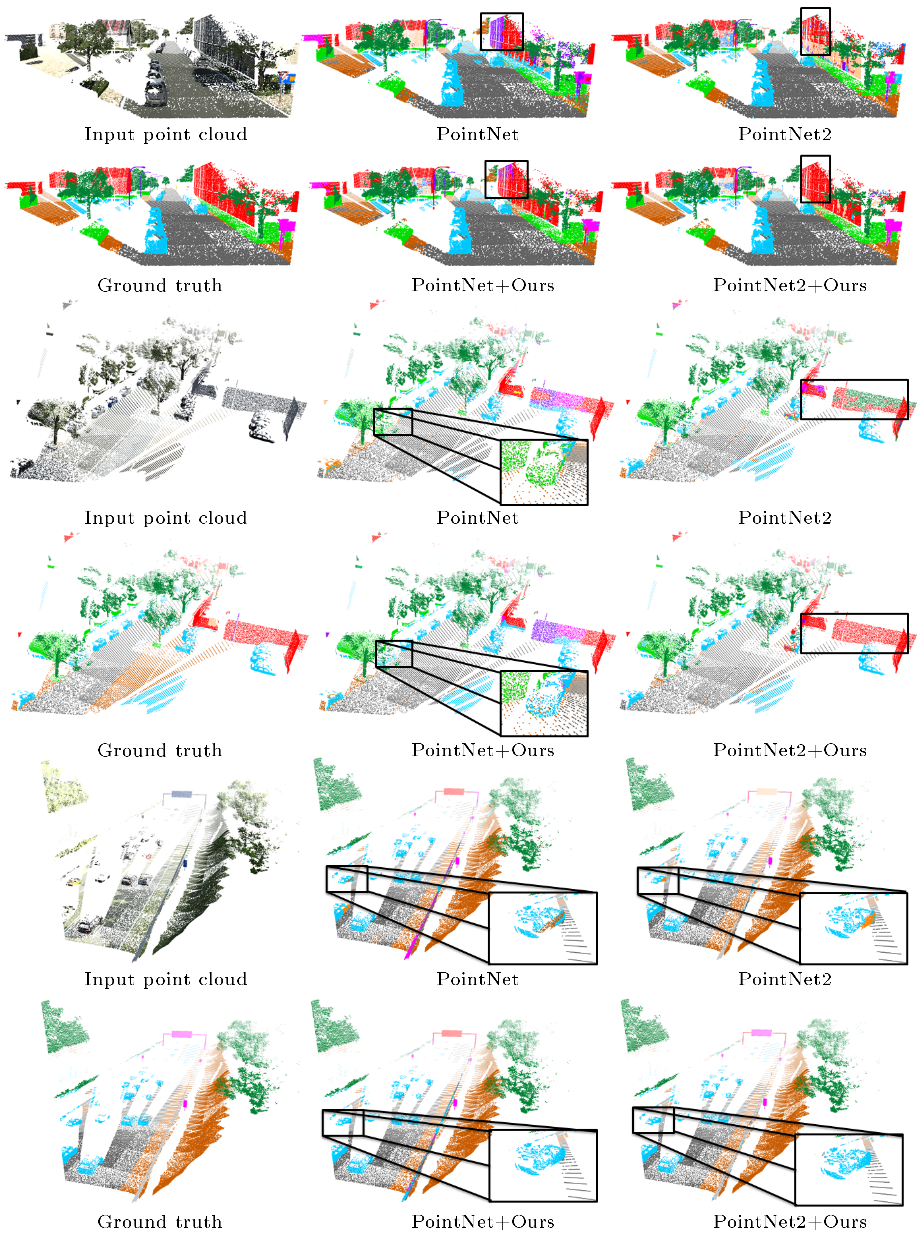


Figure 3.8: Qualitative results on vKITTI dataset. With respect to results produced by baseline models, our 3d-PSPNet succeed in correcting some mislabeled points.

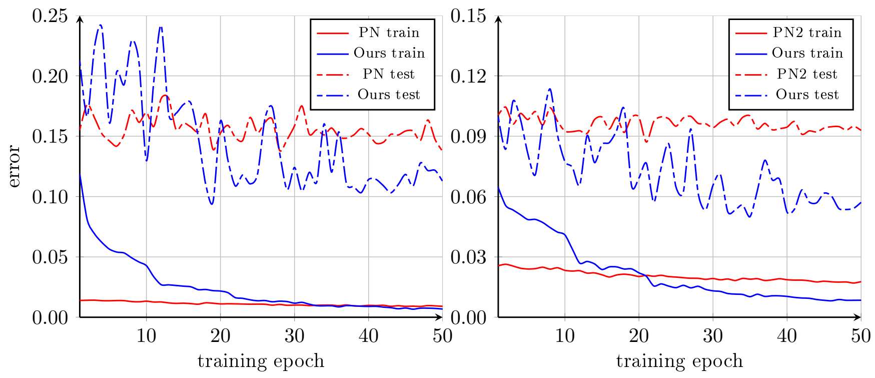


Figure 3.9: Training and testing errors of PointNet (abbreviated as PN, left image) and PointNet2 (abbreviated as PN2, right image) baseline with and without our 3d-PSPNet on Scene 6 of vKITTI dataset. Equipped with our 3d-PSPNet, the network improves OA by 2.49% and 3.58% for PointNet and PointNet2 respectively.

baseline respectively. Both qualitative and quantitative results demonstrate the improvement of our 3d-PSPNet by aggregating global contextual information from urban scene point clouds.

Method	mIOU	mAcc	terrain	tree	vegetation	building	road	guard rail	traffic sign	traffic light	pole	misc	truck	car	van
PointNet	28.43	38.65	54.19	84.43	19.43	29.16	59.77	12.27	20.31	2.81	10.74	1.98	8.82	44.39	21.29
PointNet + ours	**31.31**	**41.92**	**58.23**	**87.74**	**20.01**	**32.61**	**63.05**	**14.75**	**28.97**	**3.84**	**11.26**	**2.48**	**9.77**	**47.64**	**22.64**
PointNet2	30.94	40.09	56.41	81.32	24.94	27.07	58.34	19.99	25.10	**11.56**	12.54	1.40	**5.71**	54.62	23.25
PointNet2 + ours	**34.89**	**45.28**	**60.47**	**90.38**	**26.98**	**38.65**	**59.41**	**22.31**	**29.21**	8.89	**14.97**	**4.07**	5.68	**55.20**	**37.42**

Table 3.3: 6-fold quantitative results on the vKITTI Dataset, including mIOU, mAcc and IOU for 13 classes.

3.4.5 Network Architecture Design Analysis.

This section first analyzes the effect of some hyperparameters involved in our 3d-PSPNet and introduces our design choices. Then, we discuss the impact of input point features and transfer learning strategy on the performance of the whole network.

Number of pyramid scales L. We first analyze the impact of number of pyramid scales L on the whole network. To do so, we select $L \in \{1, 2, 3, 4, 5\}$ and perform 5 experiments under the same settings on S3DIS dataset using PointNet baseline equipped with our 3d-PSPNet. According to Figure 3.10, L plays a trade-off role between prediction accuracy and efficiency in the whole network. Increasing L promotes better mIOU results

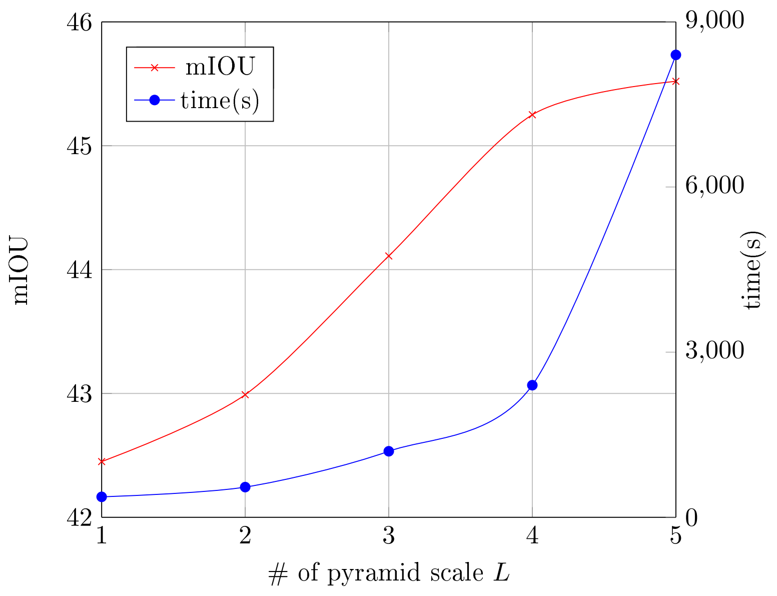


Figure 3.10: Analysis of *number of pyramid scales* L on prediction accuracy and efficiency of the framework. Noted that *time* means the average training time per epoch.

but also requires more training time. When L is smaller than 5, the curves of training time and mIOU grow in a quasi-quadratic pattern. However, when L reaches 5, the computational time explodes but mIOU only improves slightly. The explosion of computational time comes from imposing all operators on the large number of grids (16×16×16). Therefore, to balance this trade-off, we use a 4-level pyramid structure for all experiments. However, we can also choose $L = 3$ to reduce training time at the expense of accuracy.

Number of channels and size of MLP. We then study the impact of MLP on the accuracy and computational cost of the network. Note that the only extra parameters to be trained in our 3d-PSPNet are sequences of MLPs (fully connected layers) involved at each pyramid scale. These MLPs learn to aggregate information along the feature channel. To analysis the impact of MLPs on our framework, we performed 4 control experiments on S3DIS dataset with our 3d-PSPNet involving various MLPs. The segmentation results and model size are reported in Table 3.4. On one hand, increasing the output channels of fully connected layer slightly raises OA but also increases the complexity of models to be learned. On the other hand, according to the fourth control experiment, simply increasing the dimensions of MLPs will not bring any benefit to the prediction accuracy and increase the model complexity. We believe that a more carefully designed composition of fully connected layers could achieve better prediction results. However, it is a time-consuming task to manually tune the best MLP architecture while bringing more parameters to be trained. Therefore, we impose a generic architecture by simply assigning a 256-dimensional fully connected layer in MLP at each pyramid scale.

MLP	OA(%)	mIOU(%)	Model size (MB)
128	81.16	44.34	**31.21**
256	81.28	**45.54**	40.23
512	**81.45**	45.46	58.27
[512,256,128]	80.87	44.34	56.84

Table 3.4: Study on dimensions of *MLP* on S3DIS dataset with PointNet baseline model equipped with our 3d-PSPNet. Note that *model size* refers to the number of parameters to be learned in the whole framework.

Grid shape. As described in Section 3.2, there are two choices to divide the 3D scene space into sub-regional space. The first choice is the regular cubic grid which is used throughout our experiments. This structure is efficient to pool each point into its corresponding sub-regional grid by considering its spatial position in the 3D space. An alternative way is to employ the *sampling layer* of PointNet2 by selecting a subset of distant points and pool each original point into its corresponding distant point. This solution divides 3D scene into irregular grids according to the density of point clouds. We evaluate these two architectures on all three datasets with PointNet2 baseline under same training settings. The evaluation results are illustrated in Table 3.5. Our regular cubic grid version performs better on all three experiments. The main difference comes from density-invariance of regular cubic grids, where the pooling grid of each point is only dependent on its spatial coordinates.

Method	S3DIS Area 5	ScanNet	vKITTI Scene 6
Ours V1	**80.15**	**76.21**	**93.28**
Ours V2	78.79	75.81	93.09

Table 3.5: *Grid shape analysis* with PointNet2 baseline model equipped with our 3d-PSPNet. V1 is the regular-grid method, V2 is the adaptive-grid method mentioned in Section 3.2. OA(%) is reported for both version.

Input features. We also analyze how the input feature channels impact our approach. We separately feed input points with 3-dimensional channels $[x, y, z]$ and 9-dimensional channels $[x, y, z, r, g, b, \overline{x}, \overline{y}, \overline{z}]$ into both PointNet and PointNet models with and without 3d-PSPNet. We followed the control experiment setting proposed in Section 3.4.2. Table 3.6 presents a quantitative evaluation for four control experiments. Our 3d-PSPNet increases prediction accuracy in all pairs of control experiments. However, when the input features are insufficient, i.e. when containing only spatial position $[x, y, z]$, our 3d-PSPNet performs better than by feeding 9-dimensional features. The main reason is that pointwise features learned from rich input point features by baseline models are discriminative enough to reach accurate prediction results. In that case, the room for improvement is smaller than training on 3-dimensional input features. In summary, our 3d-PSPNet preserves better generalization with insufficient input features.

Method	$[x, y, z]$		$[x, y, z, r, g, b, \overline{x}, \overline{y}, \overline{z}]$	
	OA(%)	Improvement	OA(%)	Improvement
PointNet	77.12	2.20	79.49	1.79
PointNet+Ours	**79.32**		**81.28**	
PointNet2	76.37	3.78	83.11	1.54
PointNet2+Ours	**80.15**		**84.65**	

Table 3.6: *Input features analysis* on the S3DIS dataset.

Fine tuning. Finally, we study how fine tuning strategy helps accelerating the training procedure on ScanNet and vKITTI dataset. In Section 3.4.3, we train the baseline model for 20 epochs and fine-tune the network equipped with our 3d-PSPNet for another 20 epochs. In Section 3.4.4, we follow this strategy and use the pre-trained model as the staring point and fine tune the whole network for another 50 epochs. To analyze the gain of the transfer learning strategy, in the control experiments, we train the PointNet2 baseline model equipped with our 3d-PSPNet from scratch with random initialization. We follow all the same experimental settings and train the whole network for exactly the same number of epochs, i.e. 40 and 100 respectively. Evaluation results are reported in Table 3.7. After the same number of epochs, transfer learning enables the whole network to converge to a better optimal than training from scratch. In other words, transfer learning accelerates the training phase.

Method	ScanNet		vKITTI Scene 6	
	OA(%)	mIOU(%)	OA(%)	mIOU(%)
Learning from scratch	74.73	30.32	87.69	40.45
Fine-tuning	**76.21**	**33.02**	**93.28**	**45.48**

Table 3.7: *Learning from scratch* vs *transfer learning.*

3.5 Conclusion

This chapter proposes a pyramid structured network to aggregate multi-scale contextual information in point clouds. This generic module can be concatenated after any state-of-the-art pointwise feature learning network. It enriches local features with multi-scale sub-regional global clues. Experimental results on different common datasets illustrated that the enriched pointwise features are more discriminative for each objects in the complex 3D scene and produce more accurate semantic segmentation predictions.

Chapter 4

Planar shape detection

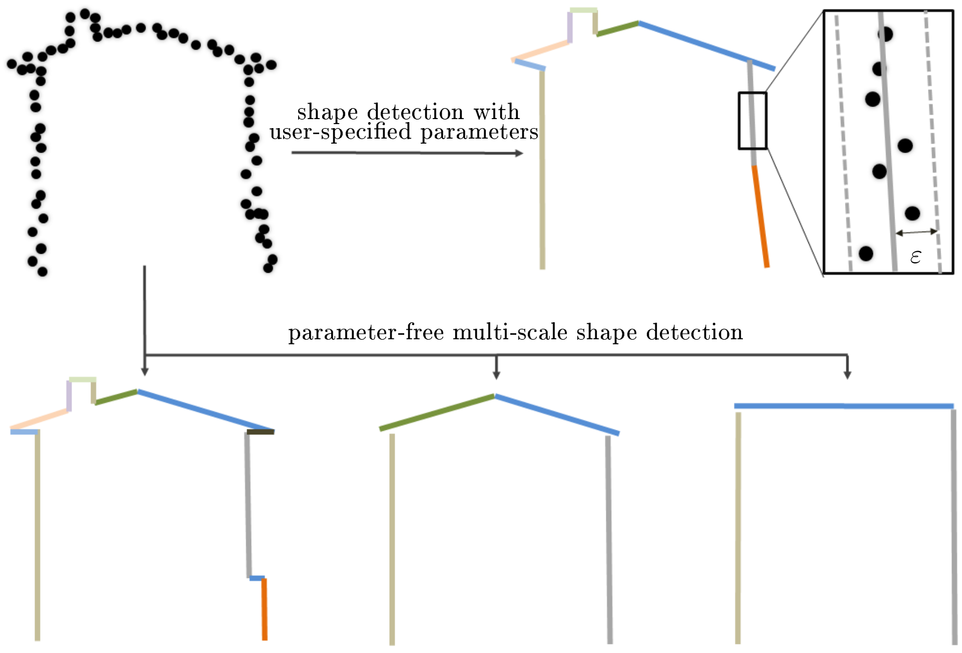

shape detection with
user-specified parameters
ε
parameter-free multi-scale shape detection

Figure 4.1: Multi-scale shape detection. Data measurements (top left) give different geometric representations of an object depending on the scale we observe it. Existing shape detection algorithms represents an object by geometric shapes given user-specified parameters as the fitting tolerance ε (top right). Instead, our algorithm extracts multiple representations of shapes at archetypical structural scales without tedious parameter tuning (bottom).

4.1 Introduction

Interpreting 3D data such as point clouds or surface meshes depends heavily on the scale of observation. Yet, existing algorithms for shape detection rely on trial-and-error parameter tunings to output configurations representative of a structural scale. We present a framework to automatically extract a set of representations that capture the shape and structure of man-made

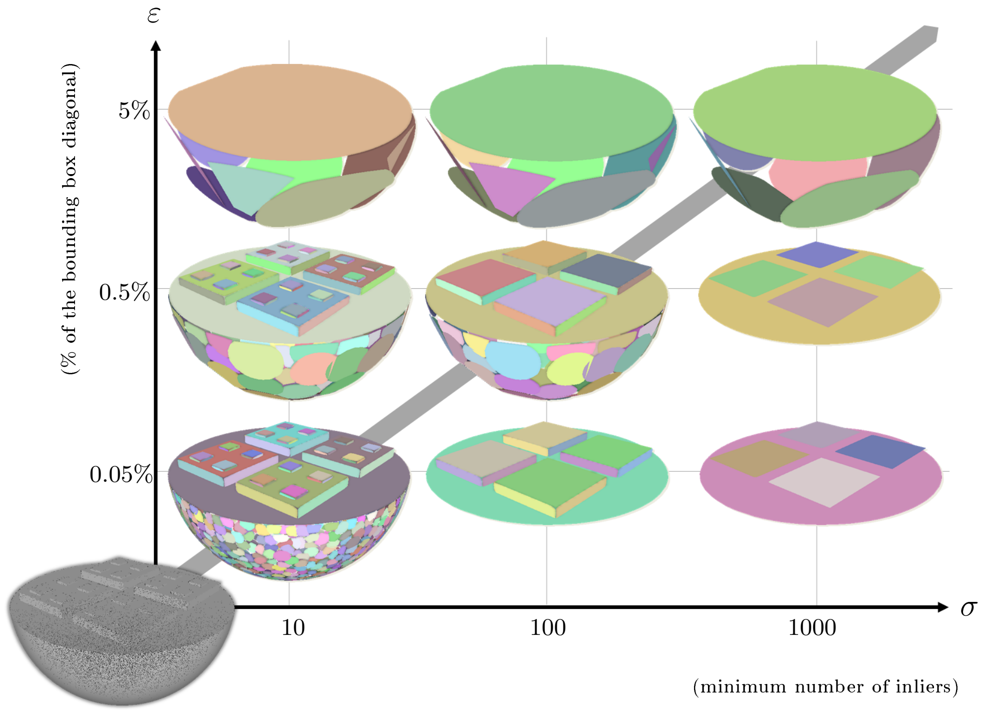


Figure 4.2: *Influence of shape detection parameters.* A point sampled object partially piecewise-planar (bottom left) is turned into a set of planar elements by region growing [RvDHV06] given a fitting tolerance ε and a minimal shape size σ. Increasing σ for a fixed ε progressively removes the smallest planar elements. Simplifications that are most representative of a key structural scale are located along the bottom-left to top-right diagonal: above (resp., below), planar regions (resp., free form parts) disappear too fast.

objects at different key abstraction levels as illustrated in Figure 4.1. A shape-collapsing process first generates a fine-to-coarse sequence of shape representations by exploiting local planarity. This sequence is then analyzed to identify significant geometric variations between successive representations through a supervised energy minimization. Our framework is flexible enough to learn how to detect both existing structural formalisms such as the CityGML Levels Of Details, and expert-specified levels of abstraction. Experiments on different input data and classes of man-made objects, as well as comparisons with existing shape detection methods, illustrate the strengths of our approach in terms of efficiency and flexibility.

The motivation behind our work is to explore the (ε, σ) space of shape approximation for a given input 3D scene, where ε quantifies the geometric tolerance to data and σ defines the minimum number of inliers: its geometric relevance to the issue of shape and scale detection has been repeatedly con-

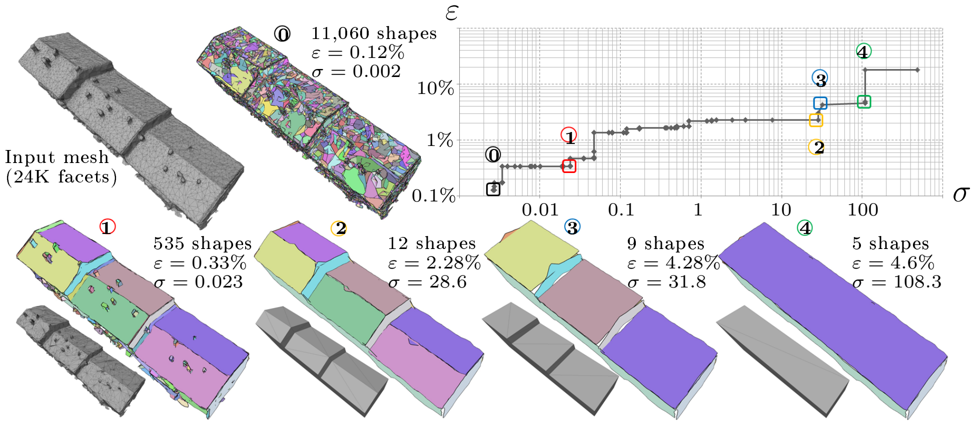


Figure 4.3: *Overview.* Starting from 3D data (here a dense mesh generated by MultiView Stereo, top left), our algorithm produces a set of high-level representations with planar primitives (representations 1–4) describing the object at different representative structural scales (bottom). By progressively merging planar regions of an initial state (representation 0), one creates a sequence of representations whose further analysis allows for the extraction of a few structurally relevant representations (top right). Such shape representations can be used, for instance, as input for piecewise-planar reconstruction [CLP10] (see grey compact meshes). Note that each shape is displayed as a colored polygon computed as the α-shape of its inliers projected onto the shape; we use this visualization of inliers in all following figures.

firmed (see, e.g., [RvDHV06]). Yet, it may appear at first sight that finding meaningful abstractions of input shapes by exploring this (ε, σ) space is simply intractable: even a greedy search through discrete sampling is unlikely to find the few key structural scales that we seek. We observe, however, that for a vast range of 3D objects (including man-made shapes), the meaningful structural scales are likely to be well captured along the (bottom-left to top-right) diagonal of the parameter space (ε, σ) as illustrated in Figure 4.2. This property has an important practical consequence: we can turn this two-parameter exploration task into a simple 1D exploration along this diagonal—a far more tractable task.

We are left with two issues to address: (i) how to sample efficiently the shape configurations along the parameter space diagonal which are likely to cross the different structural scales, and (ii) how to detect structural scales robustly.

To address (i), we propose a shape-collapsing procedure described in Section 4.2 that merges progressively pairs of planar shapes from an initial configuration with both low ε and σ, i.e., a configuration at the bottom left of the parameter space of Figure 4.2. Since merging two planar shapes cannot decrease the maximal distance to an inlier or the minimum shape size, repeated shape merging will generate a sequence of shape representations near the diagonal of the parameter space, as illustrated in Figure 4.3. Such a procedure is very efficient, and returns a fine discretization of abstractions roughly along the diagonal of our two-parameter space: starting from n planar shapes, we produce a sequence of n shape configurations called a *trajectory* in the parameter space.

As structural scales correspond to arbitrary levels of abstraction, solving (ii) by tracking and quantifying the geometric changes along this diagonal is not a reliable approach to detect them. Instead, we adopt an efficient strategy detailed in Section 4.3 that consists in learning the geometric characteristics of structural scales from a training set. The latter is typically created by a manual assignment of structural scales to the configurations of trajectories obtained by our shape collapsing procedure on a few test datasets. This training strategy offers the advantage to be fast compared to a greedy exploration of the 2D parameter space, and consistent with the way planar shapes are sampled during the testing.

4.2 Shape collapsing

Our shape-collapsing process iteratively merges two planar shapes from a current shape abstraction. This approach relies on two key ingredients: a merging operator specifying how to create a new planar shape from two existing ones, and a priority policy that orders the shape pairs to merge. Pseudocode is shown in Algorithm 1.

Initialization. We start by extracting an initial configuration of planar shapes from input data, be it a 3D point cloud or a surface mesh. A region growing algorithm [RvDHV06] is used with low parameter values, typically $\varepsilon = 0.05\%$ of the bounding box diagonal, and $\sigma = 10$ inliers. As preprocessing, we compute an adjacency graph between the detected shapes based on spatial proximity: for surface meshes, two planar shapes are considered as adjacent if at least a pair of their respective inlier facets shares a common edge in the input mesh; for a point cloud instead, two shapes are adjacent if at least a pair of their respective inlier points are mutual neighbors in the k-nearest neighbor graph of the input points (we use $k = 20$ in all our experiments).

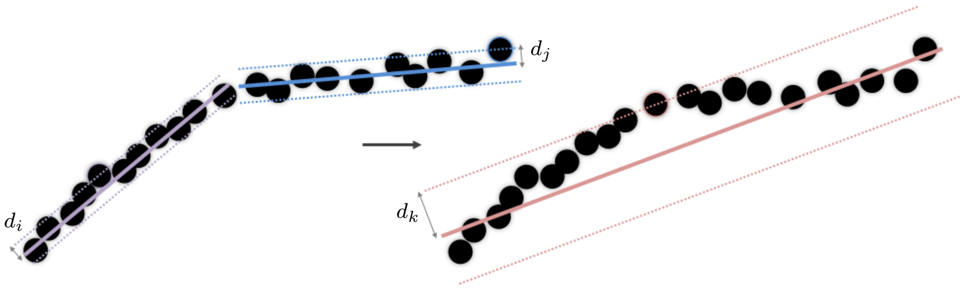


Figure 4.4: *Merging operator.* Two adjacent shapes i and j are merged into the shape k that minimizes the Euclidean distance to their joint sets of inliers. If d_i denotes the distance between shape i and its furthest inlier, note that $d_k \geq \max(d_i, d_j)$.

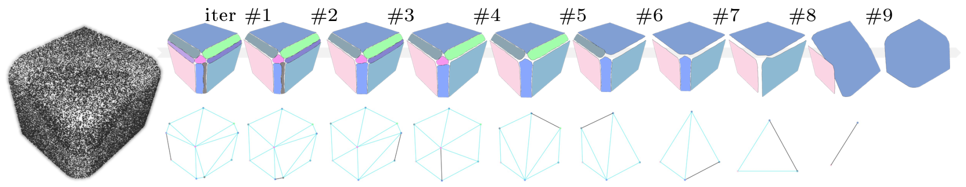


Figure 4.5: *Shape collapsing.* Iteratively merging adjacent planar elements creates a sequence of shape representations, some of which being structurally representative, e.g., representations obtained after iterations #4 and #7 (top). At each iteration, the black edge in the adjacency graph (bottom) indicates the edge with the lowest weight, i.e. the next edge to be collapsed.

Merging operator. This operator is applied on the edges of the adjacency graph. It merges two adjacent planar elements into the planar shape that minimizes the Euclidean distance to their joint sets of inliers, as illustrated in Figure 4.4. The optimal planar shape is trivially found via Principal Component Analysis.

Priority policy. In order to choose the next pair of planar shapes to merge, a weight is assigned to each edge of the adjacency graph. Merging is then performed on the edge with the lowest weight. Different metrics can be considered for specifying the weights, e.g., deviation of the normal vectors of the two planes, or area of the smallest of the two shapes. After an experimental evaluation of several metrics, we chose the Euclidean distance between input points to planar shapes as it offers the best compromise between accuracy and performance. In particular, this choice limits drifts during shape collapsing because it relies on a direct measurement to input

Algorithm 1 Shape collapse

```
Input: initial extracted planes and iteration $T$
Output: planes after collapses
Initialization: adjacency graph with weight $w_{ij}$ on each edge; $t \leftarrow 1$
while $t \leq T$ do
    - find edge with minimum weight $w_{ij}$;
    - merge plane $j$ into plane $i$ (assume plane $i$ is larger than plane $j$);
    - update adjacency graph and local edge weight;
    - $t \leftarrow t + 1$;
```

data. Formally, we define the weight w_{ij} between planar shapes i and j as

$$w_{ij} = \sqrt{\frac{1}{\sigma_i + \sigma_j} \sum_{p_k \in I_{ij}} d(p_k, P)^2} \tag{4.1}$$

where σ_i is the size of shape i, I_{ij} is the joint set of inliers from shapes i and j, and P is the optimal planar shape computed by the merging operator. At each iteration, we choose the pair of shapes with the lowest weight as the candidates to be merged. After merging two shapes, the adjacency graph as well as the weights are updated. Note that this update is local as only edges with the planar shapes adjacent to the two merged shapes are impacted. Figure 4.5 illustrates this procedure.

Semantic constraint. More special manipulation could be involved in our proposed priority policy while processing objects with well-defined structure scales, i.e. LOD of *buildings* proposed by cityGML formalism [GP12]. According to this conception, LOD1 of building is represented as block model such that *roof* is approximated by a flat plane. In this case, semantic information of each plane serves as a crucial prior knowledge to exploit the scale space such that the shape collapse trajectory passes through the well-defined LOD1 plane configuration. To achieve that, we propose a new coefficient c_{ij} considering semantic clue of two adjacent planes i and j in form of

$$c_{ij} = \begin{cases} +\infty & \text{if } s_i \neq s_j \textbf{ and } \{k \in N(j) \mid s_k = s_j\} \neq \emptyset \\ 1 & \text{otherwise} \end{cases} \tag{4.2}$$

where s_i is semantic label of each plane computed with method of [VLA15], $N(j)$ refers to the adjacent planes set of plane j and here we assume plane j will be merged into plane i. The final priority policy is updated by $w_{ij} \times c_{ij}$. The intuition behind this coefficient is that planes with different semantic labels are not supposed to be merged together unless the smaller one (which is

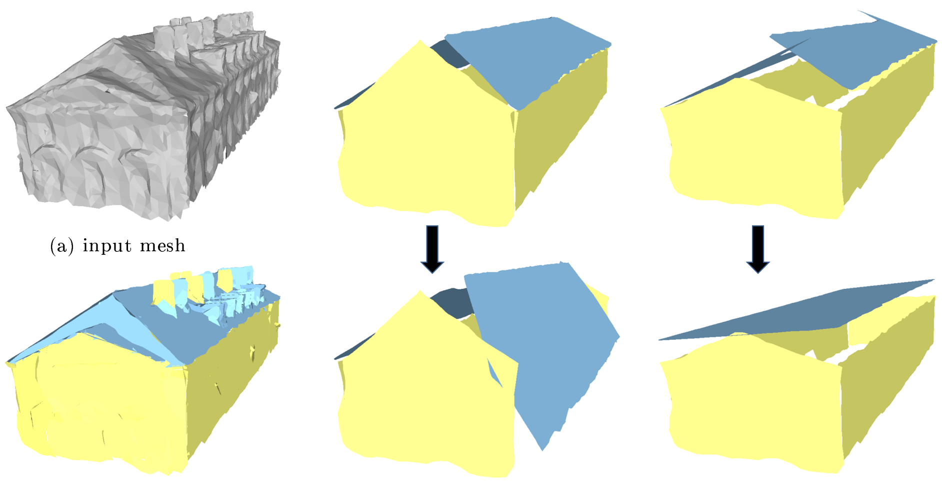


Figure 4.6: *Semantic constraint.* Starting from input mesh (a) we first extract 405 planes (b) and assign each of them a semantic label as *roof* (blue) or *facade* (yellow) using method of [VLA15]. (c) illustrates the shape collapse from 6 planes to 5 using our priority policy without semantic constraint. Two planes with semantic labeling *roof* and *facade* are merged together, which leads to skipping over the LOD1 representation of current building. However, taking the semantic constraint into consideration permits us reaching LOD1 configuration by avoiding merging planes with different semantic labels (d).

plane j in our assumption) has different semantic labels than all of its neighbors. Figure 4.6 illustrates how this semantic constraint impacts the shape collapse procedure. Note that we only employ this constraint to *buildings* throughout our experiments.

4.3 Detection of structural scales

Given a roughly-diagonal trajectory in parameter space, our goal is now to detect structural scales by analyzing the geometric evolution of the shape representations along the trajectory. For an object with simple structure, the problem can be solved in a unsupervised manner by detecting strong geometric variations between two successive piecewise-planar representations. However, in mosts cases, structural scales are levels of abstraction that cannot be reliably detected without learning from training samples. We thus formulate the detection of structural scales as a supervised labeling problem by assigning a structural scale to each shape configuration of the trajectory.

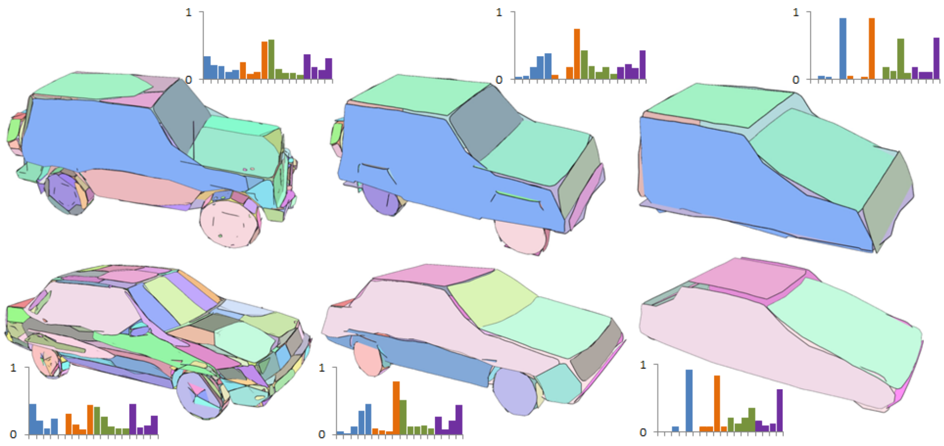


Figure 4.7: *Feature vector.* Feature vectors (see histograms) can discriminate between shape representations that capture different structural levels of man-made object, here cars. Four bins are used for both normal alignment (orange) and z-axis deviation (navy), and five bins for centroid distance (blue) and area variation (green).

Feature vector. We define a feature vector in order to characterize a configuration of planar shapes from a geometric point of view. Four different geometric descriptors are used:

- *Centroid distance* that computes the Euclidean distance between the barycenters of two adjacent shapes;
- *Normal alignment* measuring $|\mathbf{n}_i \cdot \mathbf{n}_j|$ between the normals $\mathbf{n}_i$ and $\mathbf{n}_j$ of two adjacent shapes;
- *Area variation* that computes $1 - |\sigma_i - \sigma_j|/|\sigma_i + \sigma_j|$ from the sizes σ_i and σ_j of two adjacent shapes;
- *z-axis deviation* that compares the relative orientation of two adjacent shapes with the z-axis $\mathbf{n}_z$ through the expression $|\,|\mathbf{n}_i \cdot \mathbf{n}_z| - |\mathbf{n}_j \cdot \mathbf{n}_z|\,|$.

For each descriptor, we create an histogram describing the distribution over all the pairs of adjacent shapes. We then normalize each histogram and concatenate them into a 18-bin feature vector, as illustrated in Figure 4.7. We denote by $\mathbf{f}_i$ the feature vector of shape representation i. Such a simple feature vector summarizes the main geometric characteristics of a shape representation as mutual position, orientation, size and alignment of pairs of adjacent shapes.

Energy minimization. Recall that from an initial configuration composed of n planar shapes, repeated collapsing generates a trajectory with $n-1$ shape representations. Given a finite set of structural scales $\mathcal{L} = \{1, 2, ..., K\}$, we consider a random variable $l_i \in \mathcal{L}$ that associates a structural scale to the i^{th} shape configuration of the trajectory. The quality of a label assignment $l = (l_i)_{i\in[1,n]}$ over a trajectory is measured through an energy U of the form

$$U(l) = \sum_{i=1}^{n} \psi_i(l_i) + \gamma \sum_{i=1}^{n-1} \varphi_{i,i+1}(l_i, l_{i+1}) \tag{4.3}$$

where $\psi_i(l_i)$ is a unary data term, $\varphi_{i,i+1}(l_i, l_{i+1})$ is a pairwise potential that accounts for temporal consistency between two successive shape representations, and $\gamma > 0$ is a weight balancing the two terms. In all our experiments, γ has been fixed to 0.5. Note that this formulation is basically a Hidden Markov model, so the configuration that minimizes energy U is found by dynamic programming using the Viterbi algorithm [Vit67].

Choice of ψ_i. The unary data term of shape representation i is formulated using a classifier trained by Random Forests [Bre01]. It is expressed by:

$$\psi_i(l_i) = -\frac{1}{|\mathcal{T}|} \sum_{t\in\mathcal{T}} \log(P_t(l_i|\mathbf{f}_i)), \tag{4.4}$$

where $\mathcal{T}$ denotes a set of decision trees, $|\mathcal{T}|$ the number of trees, and P_t the prediction probability of the label l_i for the decision tree t.

Choice of $\varphi_{i,i+1}$. The pairwise potential promotes temporal consistency along the trajectory: it penalizes scale changes between successive representations when geometrically too similar. This potential is defined through

$$\varphi_{i,i+1}(l_i, l_{i+1}) = w_{i,i+1} \cdot T(l_i, l_{i+1}) \tag{4.5}$$

where $w_{i,i+1} = \exp(-d_{\text{EM}}(\mathbf{f}_i, \mathbf{f}_{i+1})/2)$ is a weight measuring the similarity between feature vectors $\mathbf{f}_i$ and $\mathbf{f}_{i+1}$. The distance d_{EM} is defined as the L_2 norm of the Earth Mover distances for each descriptor histogram using a L_1 ground distance. This weight favors high geometric variation between two successive representations with different labels. The term $T(l_i, l_{i+1})$ measures jump coherence from scale l_i to scale l_{i+1}, and is defined as

$$T(l_i, l_{i+1}) = \begin{cases} 0 & \text{if } l_{i+1} = l_i \\ 1 & \text{if } l_{i+1} = l_i + 1 \\ +\infty & \text{otherwise} \end{cases} \tag{4.6}$$

The role of $T(l_i, l_{i+1})$ is to weakly penalize a jump between two successive scales while preventing other jumps in scale.

The resulting labeled sequence assigns a same label to a whole range of representations. The *first* shape representation with a given label is selected as representative of the object structure at this scale. With this choice, every planar shape is a relevant component of the object structure.

4.4 Experiments

We tested our method on three datasets with (i) different man-made objects (buildings, cars, sofa and indoor scenes), and (ii) different input data including synthetic/real-world surface meshes and point clouds. We only considered three scales in all our experiments: one scale with fine details, one with general structure and no fine details, and one with an overly-simplified general shape; but any (typically small) number of scales can be used.

- *CAD dataset.* The Princeton Shape database [SMKF04] is used to generate noise-free input point clouds that uniformly sample CAD models. Models are mainly composed of free-form shapes, including cars and sofas. The three structural scales are levels of abstraction that were specified by an expert.

- *MultiView Stereo dataset.* We created a dataset of buildings represented by dense surface meshes generated from MultiView Stereo (MVS [VKLP12]). These dense meshes contain fine details such as chimneys, but have a high amount of defects in the form of noise, holes and erroneous topology. We trained the algorithm to recognize the Levels Of Details 1, 2 and 3 defined by the cityGML formalism [GP12] as structural scales.

- *RGB-D dataset.* We also evaluated our algorithm on point clouds generated by RGB-D cameras from the Sun3D database [XOT13] and datasets from [LBF14]. These 3D point sets correspond to indoor scenes, each representing a room with walls, floor and furniture. Inputs are defect laden with variable noise, heterogeneous spatial density and

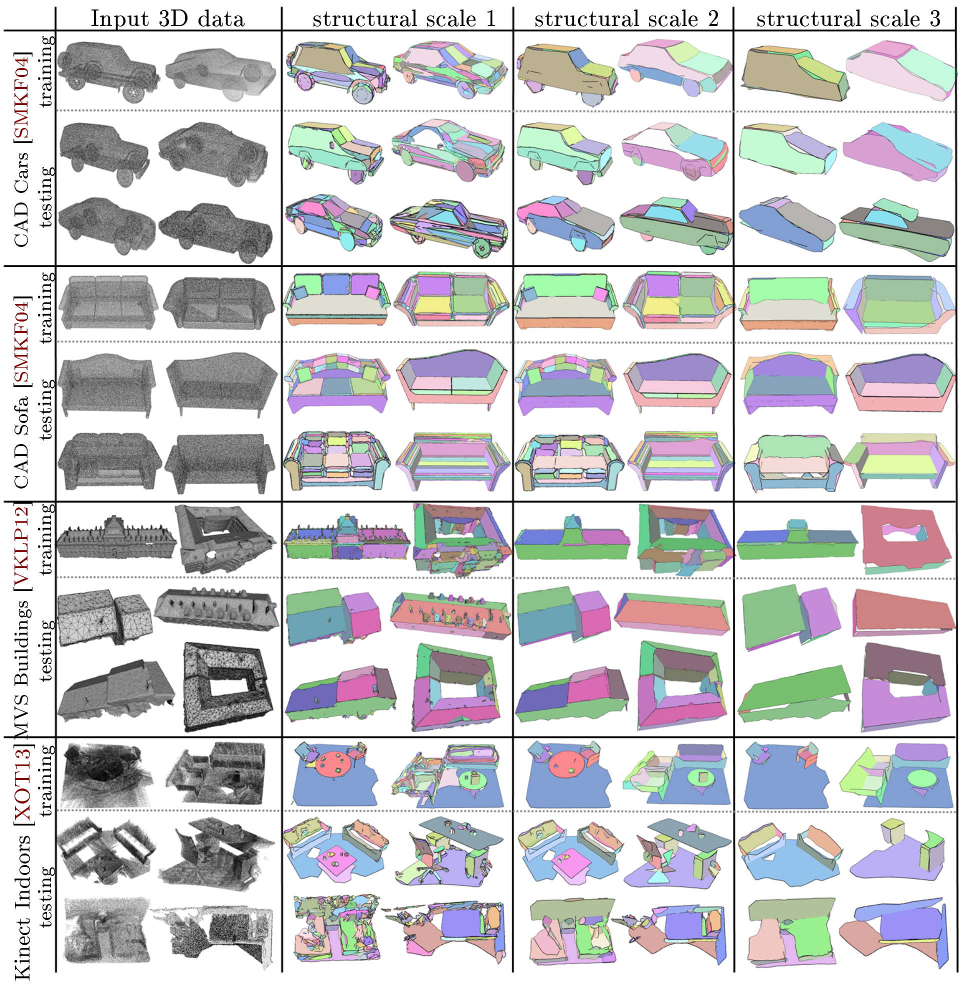


Figure 4.8: *Results on different man-made objects.* The shape representations archetypical of each structural scale generated by our algorithm on testing examples have similar structures to the training samples. In particular, our algorithm is able to learn the CityGML formalism and produce meaningful shape representations of buildings at different LODs. For indoor scenes, both furniture and permanent elements such as floor and walls exhibit the same level of detail at a given scale. Even for less structured objects such as cars or sofas, the level of abstraction conveyed by planar elements remains consistent between training and testing. Note in particular how cars at scale 1 have their bonnet described by many elements, which turn into a single element at scale 2, before merging with the windshield at scale 3.

Object class	#training samples	#testing samples	training accuracy	testing accuracy
CAD car	5K	12K	98.53%	82.88%
CAD sofa	3K	4K	97.60%	85.88%
MVS building	9K	12K	99.61%	99.30%
RGB-D indoor	20K	26K	96.90%	80.60%

Table 4.1: Accuracy of scale labeling on training and testing sets for different object classes.

severe occlusions. The three structural scales are levels of abstraction that were specified by an expert.

For each class of man-made objects, we randomly selected one third of the models for training, and the two remaining third for testing. To create planar configurations at representative structural scales for the training set, we created sequences of configurations by our automatic shape collapsing process and then assigned a scale label to each configuration by visual inspection. To speed-up the annotation, we visually detect the pairs of successive configurations where the scale changes, and then automatically annotate the configurations in between. Such a training procedure is (i) fast, *i.e.,* from 30 minutes (Multiview dataset) to 2 hours (RGB-D dataset) to create the full training set, and (ii) consistent with our two-step strategy since training samples are also generated from shape collapsing.

Qualitative and quantitative evaluation. Figure 4.8 presents some qualitative results on small portions of the three datasets. We observe that the computed representative shapes for each structural scale on testing examples are structurally similar to those in the training samples. Our framework is flexible enough to learn shape detection from both existing formalisms such as the CityGML LODs for representing buildings, and expert-specified levels of abstraction of man-made objects. Table 4.1 demonstrates that our resulting scale labeling is fairly accurate. One may note that accuracy on the MultiView Stereo dataset is much higher than for the other datasets; two main reasons explain this difference: buildings are less free-form than cars or furnitures, and levels of abstraction for building are less subjective. Once trained on a specific class of object, the classifiers do not generalize particularly well when tested on other object categories: accuracy typically decrease proportionally to the similarity between objects, *e.g.* applying the "Car" classifier on the "Sofa" dataset decreases accuracy from 86% to 63%.

Robustness to data defects, object size and initialization. As scale detection is performed using normalized features, our algorithm is only weakly

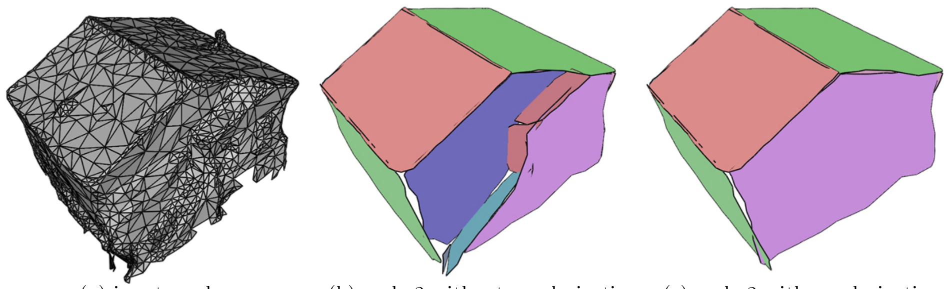

(a) input mesh
(b) scale 2 without regularization
(c) scale 2 with regularization

Figure 4.9: Regularization term. Given input mesh (a), we employ our algorithm on the shape collapse results and get structure scale 2 detection configuration (b) without regularization term and (c) with regularization term. Note that some small structures exist on the facade in (b). Involving the pairwise term considering shape similarity between consecutive configurations, we obtain (c) which preserves a more meaningful structure as defined in the training set, i.e. flat facade.

affected by noise: adding 1% random noise in the car dataset only decreases the general accuracy by 1.8%. Initialization can be an issue if we start with too large ε and σ values that are located after the first scale. In practice, there is no accuracy difference on the MVS meshes if we start with $\varepsilon = 0.05\%$ and $\varepsilon = 0\%$, *i.e.*, with each triangular facet as a shape. Since histograms of descriptors are normalized, our classifier is robust to object size variability as well: while the buildings in Figure 4.8 have quite different sizes (from small cottages to entire blocks), their shape representations are consistent at each scale.

Regularization term. Figure 4.9 illustrates how regularization term defined in Equation 4.5 improves the final classification result. Using the pre-trained random forest classifiers produces accurate labeling for most of the plane configurations along the trajectory, but still mislabel configurations near the jump points of structure scales. Inserting our pairwise term solves this issue by measuring the shape similarity between two configurations.

Timings. Learning the classifier on the different datasets requires from 5 seconds for the MultiView Stereo dataset (9K training samples) to 2.5 minutes for the RGB-D dataset (20K training samples) for a random forests training with 100 trees and 25 levels. Table 4.2 details timings for testing on one representative sample of each object class. Shape collapsing is the most time-consuming step, whereas the timing for scale detection is negligible and independent of the input complexity.

Object class	Input complexity	Initialization	Shape collapse	Scale detection
CAD car	143K pts	4.05s	10.7s	0.24s
CAD sofa	142K pts	4.79s	21.6s	0.16s
MVS mesh	3.3K facets	0.31s	0.54s	0.22s
RGB-D indoor	1.15M pts	114s	12min	0.72s

Table 4.2: Running times for testing on one representative sample of each object class (see the first testing model for each class in Figure 4.8). Experiments have been done on a single-core Intel Core i7 processor clocked at 2GHz.

Comparisons with shape detection methods. We compared our algorithm to an advanced Ransac-based method [SWK07], and the Rapter labeling mechanism [MMBM15]. A fair comparison must consider three main evaluation criteria: geometric fidelity, coverage and output complexity. We chose as measures the root mean square distance of detected shapes to inliers, the ratio of points assigned to shapes, and the number of shapes respectively. Contrary to our algorithm, these other methods required tuning some parameters as the fitting tolerance. Table 4.3 presents the evaluation scores from two input point clouds representing complex buildings, whereas Figure 4.10 shows visual results with error distributions. Our output shape representations at three different scales better capture the structure of the buildings while remaining competitive with existing methods in terms of geometric fidelity, coverage and output complexity.

	RMS	coverage	#planes
Ransac [SWK07]	0.034	0.808	128
Rapter [MMBM15]	0.042	0.817	163
Ours (scale 1)	0.017	0.816	239
Ours (scale 2)	0.29	0.816	40
Ours (scale 3)	1.03	0.816	9

Table 4.3: Comparisons on *Empire* in terms of Root Mean Square distance (RMS) of detected shapes to inliers (unit expressed as % of the bounding box diagonal), coverage (ratio of inliers) and number of shapes. Note that the shape collapsing process guarantees an identical coverage for outputs at different scales.

Application to surface reconstruction. By connecting our algorithm to a polyhedral surface reconstruction method [CLP10], we can generate compact piecewise-planar 3D models of bulidings at different LODs from

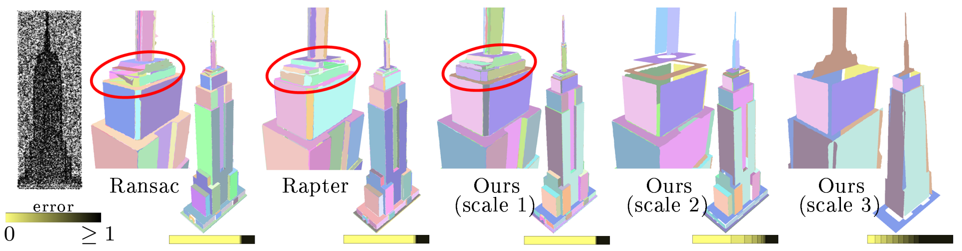


Figure 4.10: *Comparisons on Empire.* The result from Rapter [MMBM15] (courtesy of the authors) finds a visually-significant configuration of planar shapes to describe the building, whereas the one from Ransac [SWK07] was obtained by manual parameter tuning to obtain a result as close as possible as our scale 1. While Ransac and Rapter exhibit similar error distributions with respect to input points (see color histograms from yellow to black), our algorithm produces three output representations that strongly differ in terms of geometric accuracy and number of planar elements, while guaranteeing a similar coverage. Our representation at scale 1 is more meaningful than those obtained by these two methods. In particular, Ransac and Rapter omit fine planar components on the top of the tower.

dense defect-laden meshes. As shown on Figure 4.11, we outperform the state-of-the-art method of [VLA15] in terms of geometric accuracy and output complexity while conforming to the LOD CityGML formalism. Although [VLA15] is specialized in producing LOD models of buildings, our learning strategy allows us to generate meaningful configurations of planes without explicitly specifying the rules of this LOD formalism.

Design choice for priority policy. We chose the Euclidean distance as priority metrics after extensive experimental evaluation on various objects as illustrates in Figure 4.12. In particular, we tested how many times the structural scales were missed in the trajectories on a set of 30 buildings: Euclidean distance exhibited a much better score (2/90) than normal deviation (55/90) and shape area (47/90). Although these two last metrics are fast to compute, they are not direct metrics to input data, leading often to drifts during shape collapsing. We also tested a weighted sum of these three metrics. In this case, weights in front of each metrics were learned from the feature vectors of the trained samples to better adapt the tracking of scale i once scale i-1 was detected. However, such a mechanism was extremely costly as shape collapsing and scale detection were no longer performed serially, and the accuracy gain compared to the Euclidean distance was negligible (0.06% gain on buildings dataset). Euclidean distance is, at the end, a good compromise between accuracy and performance. Note also

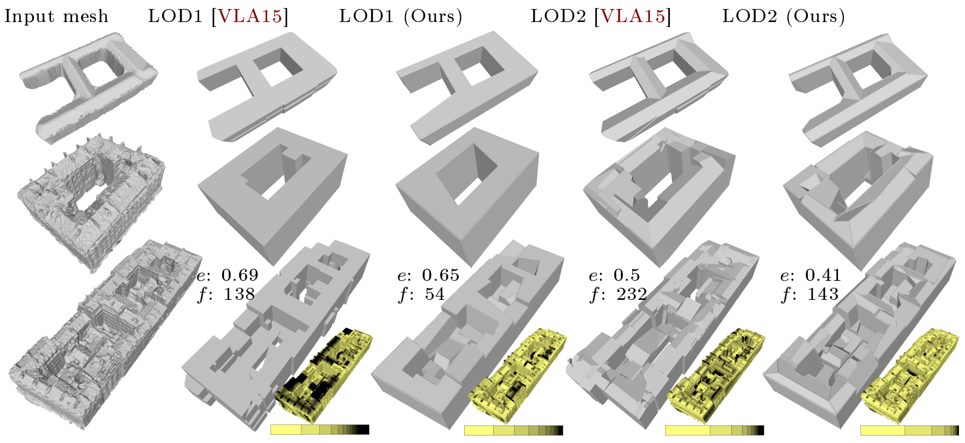


Figure 4.11: *Application to reconstruction of LOD models of buildings.* Our algorithm combined with a piecewise planar reconstruction algorithm [CLP10] produces compact LOD1 and LOD2 models from dense defect-laden meshes that outperform those delivered by a building-specific LOD generation method [VLA15] in terms of both geometric accuracy —as shown using color histograms from yellow (0 meter error) to black ($\geq$ 2 meter error)— and output complexity, where e refers to geometric error and f is number of output facets.

that, as the scales are learned by annotated samples obtained by the same collapsing mechanism than during testing, trajectories in the testing stage are less likely to miss the structural scales in practice.

Limitations. Although our framework is designed to be flexible, the choice of the metric (Equation 4.1) that specifies the priority weights during shape collapsing is independent of the object's category. As suggested by Table 4.1, our choice is relevant in the case of buildings for exploring LODs, but not always optimal for more free-form objects such as furniture. Ideally, this metric should be learned from a training set of trajectories. This variant would however be very costly in practice as shape collapsing and scale detection are no longer performed serially. Additionally, our algorithm does not discover and preserve geometric regularities such as parallelism, orthogonality or symmetry of shapes contrary to recent shape detection methods as [MMBM15]. This does not affect geometry fidelity and coverage, but may lead to suboptimal shape abstractions that fail to respect these specific features.

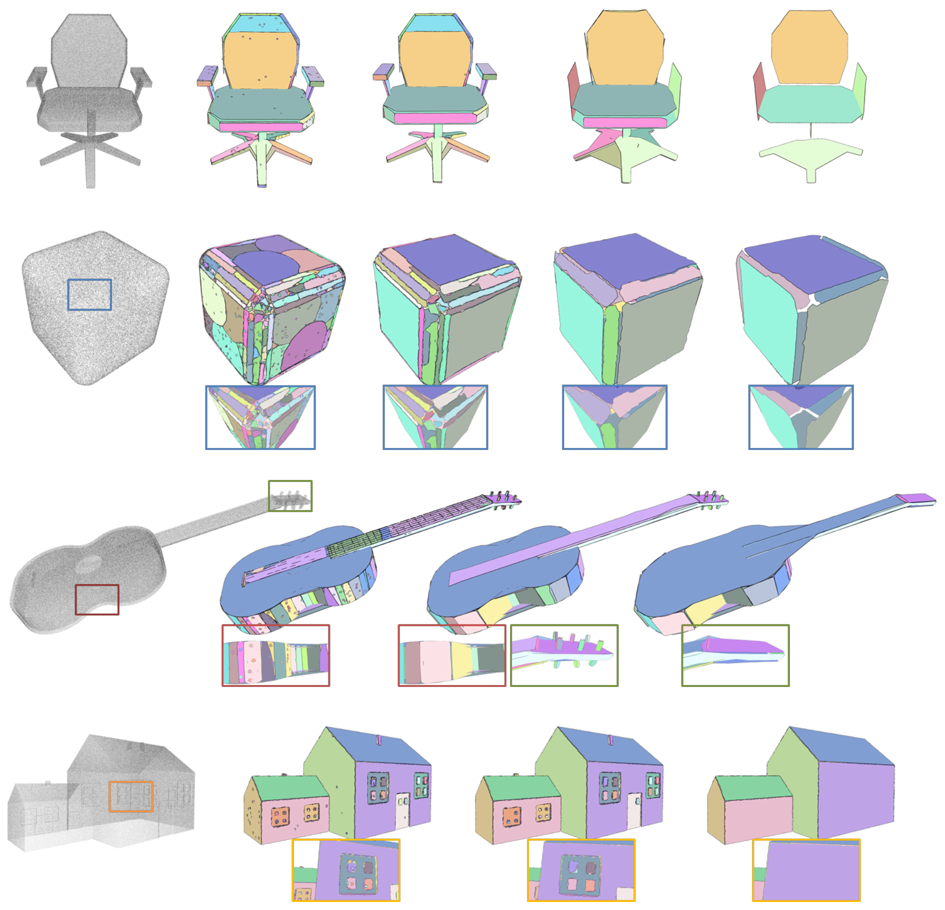


Figure 4.12: Shape collapse results on different objects with Euclidean distance priority metrics. This robust metric choice hierarchically merges planes in a structure-aware way (from left to right: shape collapse direction), which is the essential criteria to generate trajectory along the parameter space.

4.5 Conclusion

Our work provides a parameter-free algorithm for detecting piecewise-planar shapes from 3D data. Contrary to existing methods that require tedious parameter tuning, our algorithm extracts multiple representations of an input shape at key structural scales whose characteristics are learned from a training set. Our framework is flexible enough to learn both existing structural formalism such as the CityGML Levels Of Details for representing buildings, and expert-specified levels of abstraction on man-made objects. Experiments demonstrate the added value of our approach with respect to existing shape detection methods, as well as its potential to help with surface reconstruction and approximation.

Chapter 5

Piecewise-planar reconstruction

5.1 Introduction

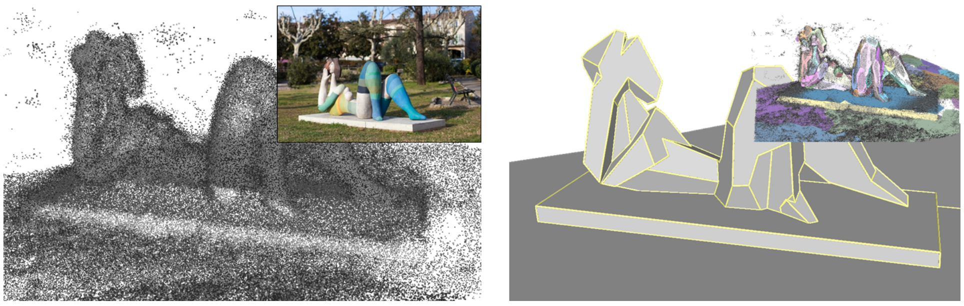


Figure 5.1: Objective of our algorithm. Given raw 3D data as a point cloud generated from Multi-View Stereo (left), our algorithm assembles a set of planar shapes into a compact polygonal mesh (right). The algorithm is particularly adapted for describing piecewise planar man-made objects and scenes, but can also be used to approximate free-form shapes as, here, a statue composed of curved and thin volumes.

Robust polygonal surface reconstruction algorithms typically operate by slicing a 3D domain with planes detected from input points. This operation generates a set of polyhedra and facets from which output surface is then extracted. Because this slicing operation is computationally costly, the best existing algorithms can laboriously handle more than one hundred planes under reasonable times. In this chapter, we specifically tackle this scalability issue. The core idea consists in slicing a 3D domain in a more flexible and scalable manner than existing mechanisms. We propose a data-structure which is i) spatially-adaptive in the sense that a plane slices a restricted number of relevant planes only, and ii) composed of components with different structural meaning. We also propose a surface extraction mechanism that delivers intersection-free and 2d-manifold surface meshes from such partitioning data-structures. Our experiments on a variety of objects and sensors

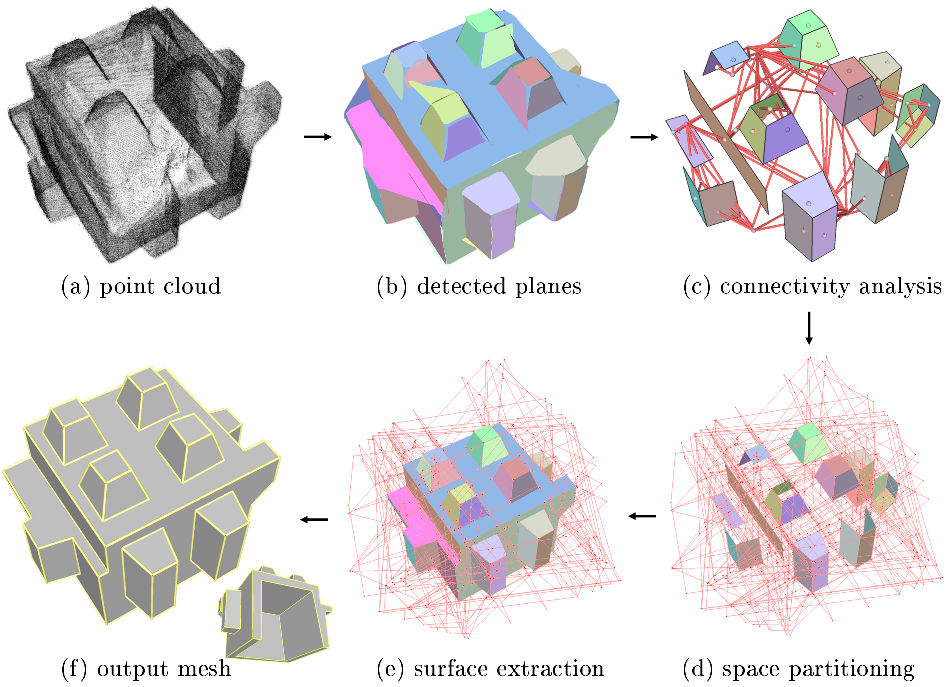


Figure 5.2: Overview of our approach. Our algorithm starts from a point cloud (a) and a set of primitives whose α-shapes are represented by colored polygons (b). By analyzing the connectivity graph of primitives (see red edges), we extract some structurally-valid facets represented by colored polygons with black edges (c). This quick connectivity analysis allows us to treat 35 of the 60 primitives on the shown example. We then build the partitioning data-structure (see the pink wireframe) by slicing the spatially-close unprocessed primitives while embedding the structurally-valid facets found in the previous step (d). The last step selects a subset of polygonal facets from the partition data-structure (e). The output is a 2d-manifold polygonal mesh in which each facet is a polygon supported by one of the primitives (f).

show the versatility of our approach as well as its competitiveness with respect to existing methods.

More specifically, our algorithm takes as input a point cloud or a dense mesh and returns as output a polygonal mesh which is 2d-manifold, watertight and intersection-free. Figure 5.1 shows the goal of our approach. Optionally, the user can relax these geometric guarantees. We first extract from the input 3D data a set of primitives by standard methods [RvDHV06, SWK07]. For each detected primitive, we compute (i) a rough approximation of its boundaries using α-shape [EKS83], and (ii) an oriented 2D bounding box, *i.e.* the smallest rectangle lying on the detected plane that contains all its projected inliers. We call a $\varepsilon-$*bounding box*, the oriented 2D bounding box scaled up by an offset ε.

The algorithm operates in three steps illustrated in Figure 5.2. First, the connectivity relations between primitives are analyzed in order to search for structurally-valid surface components. This step, presented in Section 5.2, allows us to quickly process a part of the input primitives and solve obvious assembling situations before slicing operations. We then build the partitioning data-structure in Section 5.3 by slicing the spatially-close unprocessed primitives while embedding the structurally-valid components found in the previous step. Finally, the output surface is recovered by selecting a subset of polygonal facets from the partition data-structure using an energy minimization formulation presented in Section 5.4.

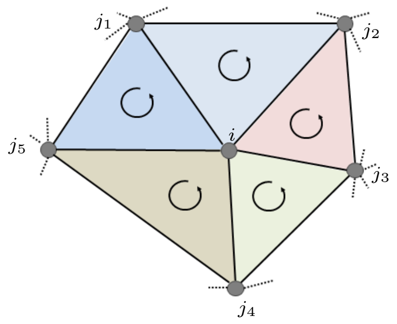
5.2 Connectivity analysis

The objective of the first step is to quickly solve obvious local assemblings of some primitives by analyzing the connectivity relations between them.

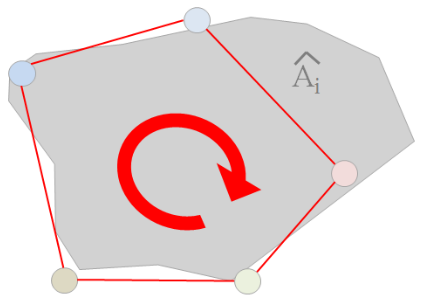
We define the notion of strong connectivity for characterizing primitives that are spatially very close. When detected from point clouds, two primitives are said *strongly-connected* if at least two inlier points fitted each to one of the two primitives are mutual neighbors in the k-nearest neighbor graph of the input points. In case of input meshes, two primitives are strongly-connected if at least one inlier facet from the first primitive share an edge with an inlier facet of the second primitive. We operate our analysis on the *connectivity graph* where each node is associated with a primitive, and each edge with a pair of strongly-connected primitives. From real-world data, such a graph usually contains errors with missing and invalid connections.

Our strategy is to search for structurally-valid facets in this graph.

Extracting corners, creases and border polygons. We first detect all the 3-cycles in the connectivity graph, *i.e.* triples of primitives that are mutually connected. The point located at the intersection of the three corresponding planes is called a *corner* if it is close from the α-shapes of the three primitives. In practice, we impose a maximal distance of 5% of the 3D bounding box diagonal. This condition allows us to ignore a corner positioned far away from its primitives, which typically occurs when primitives are nearly parallel. We then detect *creases*, *i.e.* the line-segments linking pairs of corners which have exactly two primitives in common. Finally, we extract *border polygons* of each primitive, *i.e.* the simple cycles of creases lying on the primitive.



(a) Connectivity graph



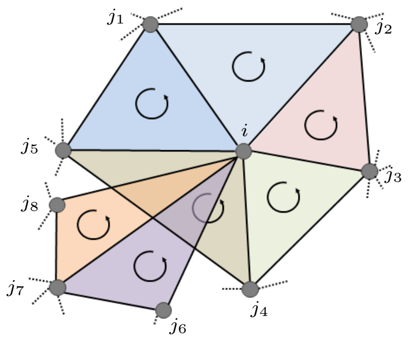
(b) Corners and creases on plane i

Figure 5.3: Data consistency condition. Primitive i is strongly connected to 5 primitives $j_1, .., j_5$. In the connectivity graph (a), we detect 3-cycles (black curved arrows) that correspond to (corner) points at the intersection of 3 planes in the 3D space (see colored dots in (b)). We then detect creases (red edges) by searching the pairs of corners which have exactly 2 primitives in common. A close sequence of creases (see red curved arrow) is a border polygon. The latter is not a structural facet on the shown example because the data consistency condition is not valid: the facet does not overlap well with the α-shape of plane i (see grey polygon $\widehat{A_i}$).

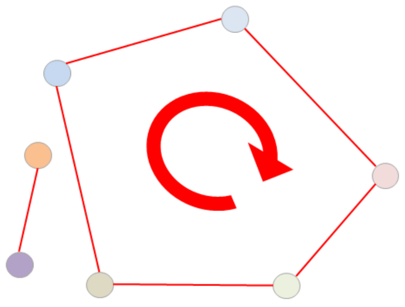
Extracting structural facets. A primitive with border polygons hosts a facet which is potentially a good candidate to be part of the output surface. In presence of one border polygon, this facet is simply defined as its inside surface. When two border polygons are nested, ie one of these polygons is contained in the second one, we define the facet as the surface in between the two polygons. When border polygons intersect, we do not create facet to avoid non-manifold degeneracies. Such a facet is called a *structural facet* if two conditions are respected:

- *Data consistency*: The facet must strongly overlap with the α-shape of the primitive,
- *Structural validity*: all the creases lying on a primitive must belong to the border polygons of that primitive.

The first condition checks whether the facet is well recovered by the α-shape of the primitive as illustrated in Figure 5.3. In practice, we impose an overlapping ratio higher than 0.9 between the facet and the α-shape of the primitive. The second condition guarantees that the facet is unique and connect in a 2d-manifold way with facets induced by the other primitives. It thus prevents from structural degeneracies, in particular the crossing of the facet by another primitive. An example of configuration that does not fulfill this condition is illustrated in Figure 5.4.



(a) Connectivity graph



(b) Corners and creases on plane i

Figure 5.4: Structural validity. Continuing on the example of Figure 5.3, 3 more primitives j_6, j_7 and j_8 are strongly connected to primitive i. The structural validity condition is not respected here because the left isolated crease does not belong to the border polygon.

Structural facets connect between each others to form 2d-manifold polyhedral surface components that partially describe the observed object. The border edges of these components necessarily lie on the remaining primitives: we call them *anchor edges*. We impose the structural facets to be part of the final output mesh and discard their corresponding primitives for the subsequent steps. We denote by $\mathcal{P}$, the set of remaining primitives.

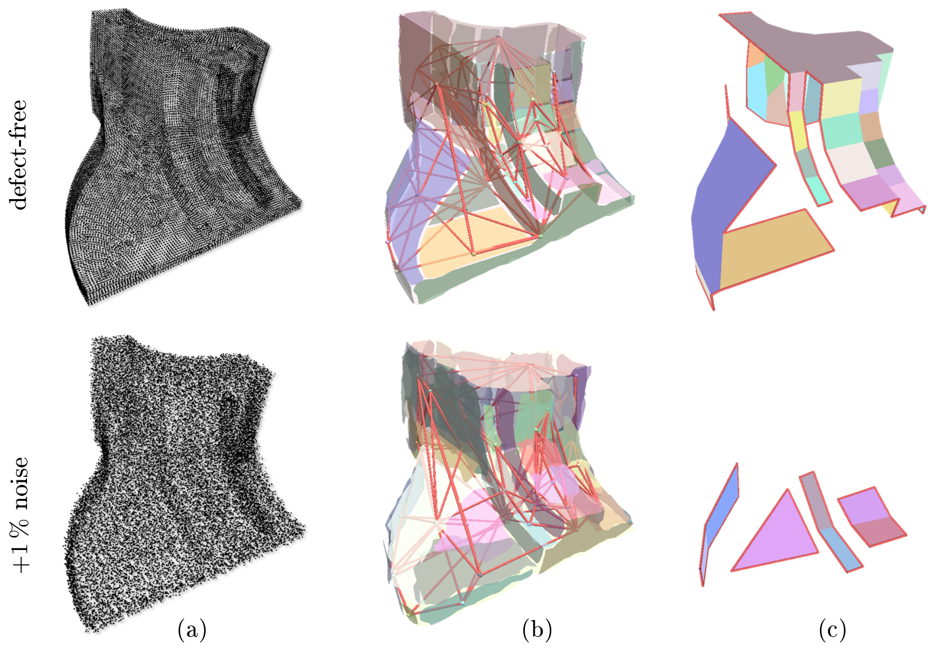
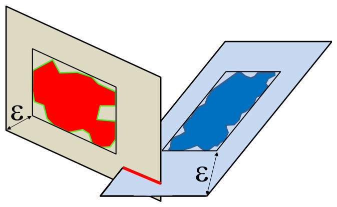


Figure 5.5: Connectivity analysis on *Fandisk*. Primitives and associated connectivity graph (b) are typically accurate when input data (a) is clean (top). Our quick connectivity analysis allows us to process 36 of the 45 initial primitives on the top example, leading to the reconstruction of 36 structural facets (c). When data is defect-laden, for instance highly noisy (bottom), the connectivity analysis is less efficient: connectivity graph contains many ambiguities that restricts the number of structural facets. Nevertheless the 7 structural facets recovered on the bottom example are all relevant. Anchor edges are colored in red in (c).

This mechanism solves obvious plane assembling situations to lighten the time-consuming slicing operations that come next. It is less efficient in presence of defect-laden data where few structural facets are extracted in practice, as illustrated in Figure 5.5. That said, the recovered structural facets are relevant as the data consistency and structural validity conditions are strict and highly selective. Choosing to detect more structural facets, potentially wrong ones, and to let following steps selecting them is a more robust alternative, but it would lead to a much more complex partitioning data structure where all primitives should be inserted. This would significantly reduce scalability and increase running times with respect to our strategy.

5.3 Space partitioning

Primitive slicing is usually performed in a greedy manner in the literature. Typically, one first computes the *slicing domain* of each primitive, *i.e.* the polygon lying on the primitive plane and bounded by the 3D bounding box of the observed object. Then, the 3D bounding box is divided into polyhedra by inserting one per one each slicing domain in an arbitrary order: the first slicing domain splits the 3D bounding box into 2 polyhedra, the second slicing domain typically splits the these two polyhedra into four polyhedra, etc. Because such a slicing strategy considers the intersection of all pairs of slicing domains, the number of polyhedra increases exponentially with respect to the number of primitives. In practice, only a small portion of these intersections is relevant. To reduce the computational burden of this operation, we restrict the pairs of primitives to be sliced. We define the notion of soft-connectivity to avoid intersecting slicing domains whose primitives are not close enough. Two primitives are said *softly-connected* if their ε-bounding boxes intersect inside the 3D bounding box of the observed object. This connectivity relationship is fast to compute and less restrictive than strong-connectivity.



As illustrated in Figure 5.6, this strategy allow us to strongly reduce the complexity of the partitioning data-structure when combined with the structural facets extracted in Section 5.2. Note that more advanced connectivity relationships inspired from collision detection problems could be used to better match primitives, but this would be more time-consuming than a direct distance between 3D rectangles.

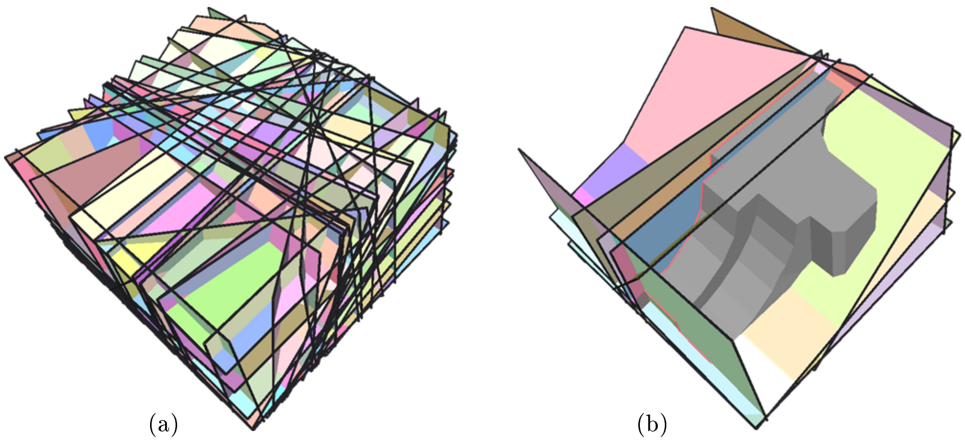


Figure 5.6: Primitive slicing with and without structural facets. Intersecting the 45 primitives of the defect-free version of the *Fandisk* model (see Figure 5.5) produces a complex partition composed of nearly $2K$ facets (a). By embedding the 36 structural facets (grey mesh), the complexity of the partition with the 9 remaining primitives drops to less than a hundred facets (b).

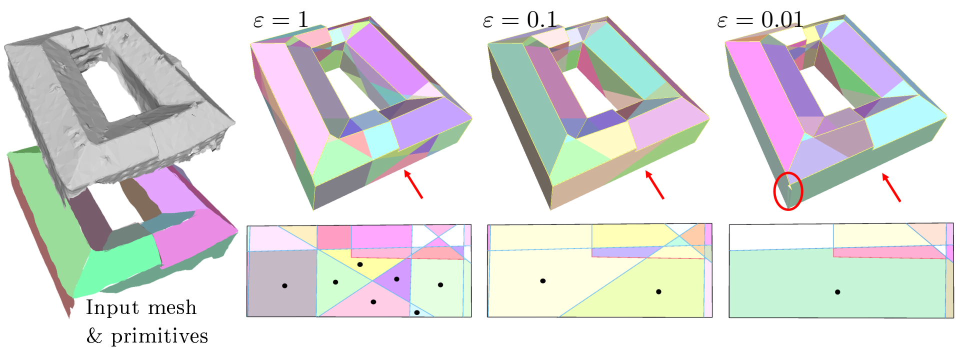


Figure 5.7: Soft-connectivity. When all the remaining primitives intersect with each others ($\varepsilon = 1$), the 2D partition of the front facade of the building is over-fragmented (see colored polygons on the top right frame with the anchor edges in red and the intersection lines in blue; polygons with a black dot indicate they belong to the output surface on the left). Decreasing ε reduces the complexity of 2D partitions. In presence of holes in the input mesh, primitive intersections can be missed when ε is too low (see the missing intersection between the front and left facade in the case where $\varepsilon = 0.01$). ε is expressed as a ratio of the bounding box diagonal of the scene.

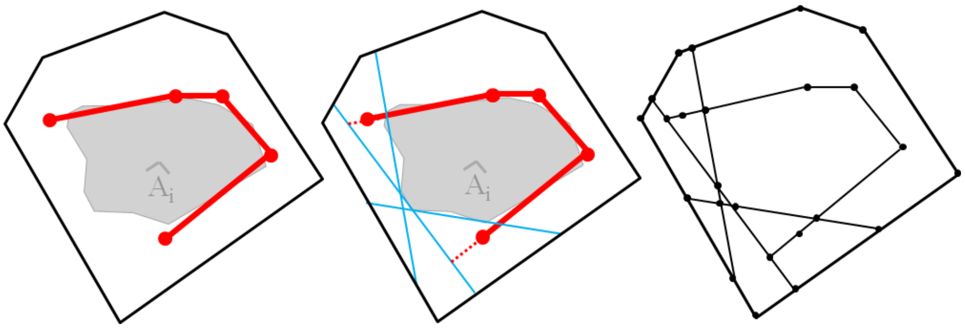


Figure 5.8: Slicing operations. (a): we first compute the slicing domain (back lines) of primitive i and insert the anchor edges associated with this primitive (red segments). (b): we then insert line-segments defined as the intersection with the slicing domains of softly connected primitives (blue lines) and extend anchor edges whose extremities are not connect to other anchor edges (dashed red lines). (c): the intersections of these different lines and edges give us the 2D partition of polygonal facets associated with primitive i.

In practice, we first intersect the slicing domains of softly-connected primitives to form a 2D partitions of polygonal facets. Potential anchor edges lying on primitives are then inserted into the corresponding 2D partitions. The anchor edges whose extremities are not connected to other anchor edges are extended until meeting an intersection line or the border of the slicing domain. We finally split edges that cross anchor edges. Figure 5.8 illustrates these different slicing operations. Note that such a strategy generates a set of polygonal facets which can possibly intersect between each others without necessarily sharing an edge.

The value of ε controls the complexity of the partitioning data-structure, as illustrated in Figure 5.7. Choosing a low ε value gives a set of light 2D partitions, low running time and low memory consumption, but is less likely to be robust to missing data.

We denote by $\mathcal{F}$, the set of polygonal facets contained in all the 2D partitions, and $\mathcal{E}$, the set of edges. Note that edges are typically adjacent to four facets, except in case of anchor edges and rare situations where at least three primitives intersect along the same line.

5.4 Surface extraction

Contrary to existing methods [CLP10, NW17, VLA15], our set of polygonal facets $\mathcal{F}$ and edges $\mathcal{E}$ does not necessarily constitute a regular partition of polyhedral cells in the sense that cells can overlap and polygonal facets can intersect between each others. Traditional polyhedron labeling methods by Graph-Cut [CLP10, VLA15] not being applicable to our partition, we adopt a more flexible facet selection approach inspired by the integer programming formulation of [NW17]. In particular, such a formulation allows us to impose some geometric constraints on the expected solution, *e.g.* the intersection-free guarantee. Contrary to [NW17], our energy model (i) operates on an irregular partition that requires additional linear constraints and (ii) relies on a new data term that does not directly depend on time-consuming measurements to input data.

We denote by $x_i = \{0, 1\}$ the activation state of facet $i \in \mathcal{F}$, and by $\mathbf{x} = (x_i)_{i \in \mathcal{F}}$ a configuration of activation states for all facets in $\mathcal{F}$. The set of active facets, *i.e.* so that $x_i = 1$, constitutes the polygonal facets of the output surface.

Energy. We measure the quality of a configuration $\mathbf{x}$ with a two-term energy of the form

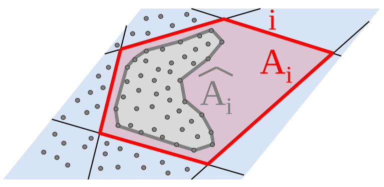
$$U(\mathbf{x}) = (1 - \lambda)D(\mathbf{x}) + \lambda V(\mathbf{x}) \tag{5.1}$$

where $D(\mathbf{x})$ and $V(\mathbf{x})$ are terms living in $[0, 1]$ measuring data consistency and surface complexity. $\lambda \in [0, 1]$ is a parameter balancing these two terms.

The linear term $D(\mathbf{x})$ encourages facets recovered by inliers to be activated as:

$$D(\mathbf{x}) = \beta \left(1 - \sum_{i \in \mathcal{F}} \frac{A_i}{A} x_i\right) + (1 - \beta) \left(\sum_{i \in \mathcal{F}} \frac{A_i - \widehat{A_i}}{A - \widehat{A}} x_i\right) \tag{5.2}$$

where A_i is the area of facet i, $\widehat{A_i}$ the area of α-shape of the inliers falling in facet i, A the sum of areas of all facets, $\widehat{A}$ the sum of areas of all primitive α-shapes. The first part of the expression encourages the activation of facets homogeneously recovered by data. Because the cost of non-activation is null, the second part of the expression is required to penalize the non-activation of facets. β is a parameter living in $[0, 1]$ that allows these the two opposite forces to be



counter-balanced. It acts as a trade-off between local correctness of facets and global coverage. In our experiments, we set β to 0.5, except for inputs with missing data where the value is increased to 0.7.

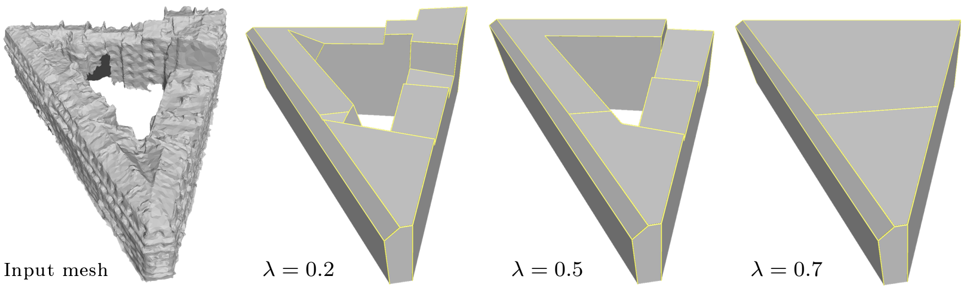


Figure 5.9: Impact of parameter λ. Increasing λ reduces the complexity of the output model. At $\lambda = 0.7$, only a small portion of the 21 detected primitives plays a role in the output model. Note how the inner courtyard disappears.

The quadratic term $V(\mathbf{x})$ favors low complexity output surface in a similar way than the one proposed by [NW17]

$$V(\mathbf{x}) = \frac{1}{|\mathcal{E}_\sim|} \sum_{(i,j)\in\mathcal{E}_\sim} 1_{\{i \bowtie j\}} x_i x_j \tag{5.3}$$

where $\mathcal{E}_\sim$ is the set of pairs of facets in $\mathcal{F}$ that share an edge, $|\mathcal{E}_\sim|$ its cardinality, $1_.$ the Heaviside function, and $i \bowtie j$ the geometric relationship which is true when facets i and j are not coplanar. This term favors output surfaces with large facets by penalizing the presence of creases, as illustrated in Figure 5.9.

Constraints. We introduce three linear constraints in order to impose some geometric guarantees on the output surface.

- *Structural constraint* imposes the structural facets to be active, *i.e.* part of the output surface (Eq. 5.4):

$$x_i = 1, \quad \forall i \in \mathcal{F}_s \tag{5.4}$$

  where $\mathcal{F}_s$ corresponds to the set of structural facets.

- *2d-manifold and watertight constraint* traditionally imposes each edge to be shared by zero or two facets. As input points have often missing parts on their 3D bounding box (see for instance *Church* and *Face* in Figure 5.11), we relax the watertight constraint on edges lying on the 3D bounding box. This allow us to avoid either shrinking the output surface or increasing computational complexity by adding facets of the six sides of the 3D bounding box in $\mathcal{F}$. Note that, for a strict watertightness, such border edges can be easily filled in as post-processing. We formulate this constraint as

$$\sum_{k \in F_e} x_k = 0 \text{ or } 1, \quad \forall e \in \mathcal{E}_{border} \tag{5.5}$$

$$\sum_{k \in F_e} x_k = 0 \text{ or } 2, \quad \forall e \in \mathcal{E}_{\overline{border}} \tag{5.6}$$

where F_e is the set of facets adjacent to edge e, $\mathcal{E}_{border}$ is the set of edges lying on one of the six sides of the 3D bounding box, and $\mathcal{E}_{\overline{border}}$ its complementary set in $\mathcal{E}$.

- *Intersection-free constraint.* As $\mathcal{F}$ can contain facets that intersect, we impose such pairs not to be active at the same time

$$x_i + x_j = 0 \text{ or } 1 \;\; \forall (i,j) \in \mathcal{I} \tag{5.7}$$

where $\mathcal{I}$ is the set of pairs of facets in $\mathcal{F}$ that intersect. Note that when ε is set to 1, this constraint is not necessary as the partition is guaranteed to be free of intersecting facets by construction.

Figure 5.10 shows the impact of these constraints on the output solution. The activation of the 2d-manifold and watertight constraint is required in most cases, unless the end-user is satisfied with a rough polygon soup. The activation of the intersection-free constraint is required only when input data contained defects as noise and outliers.

Optimization. We search for the configuration $\mathbf{x}$ that minimizes the energy U while imposing Eq. 5.4, 5.5, 5.6 and 5.7 to be true. We solve this quadratic optimization problem under linear constraints using a standard integer programming library [GO16]. In practice, we turn it into a linear optimization problem by inserting the extra-variables $y_k = x_i x_j$.

5.5 Experiments

The algorithm has been implemented in C++, using the Computational Geometry Algorithms Library [The17] which provides the basic geometric tools for mesh-data structures.

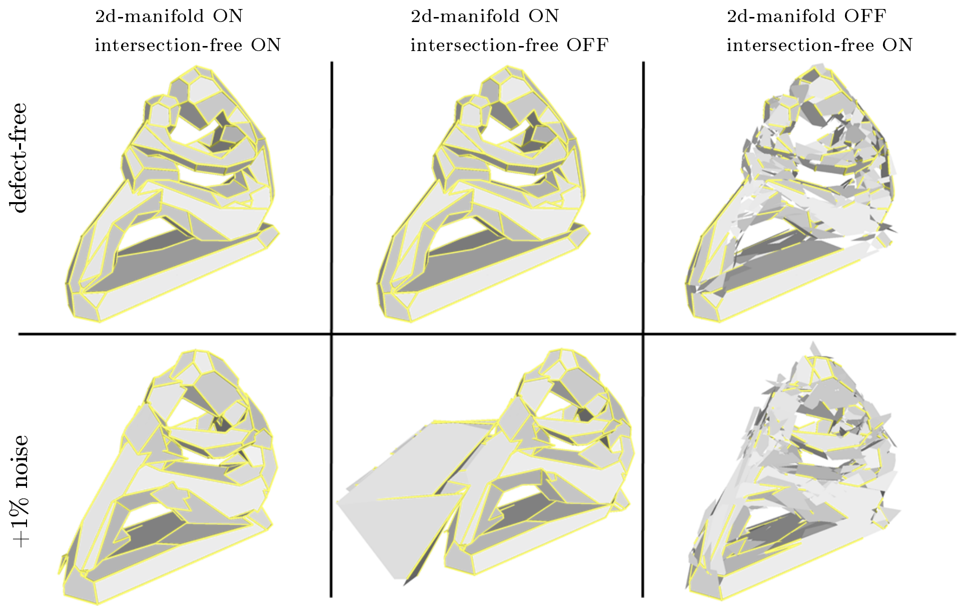


Figure 5.10: Impact of constraints. Without activating the 2d-manifold and watertight constraint, the output surface exhibits poorly connected facets as well as holes and edges adjacent to more than two facets (right). Deactivating the intersection-free constraint has no impact on the quality of the output mesh when input data is defect-free (top-middle) but tends to make the output surface too complex with groups of facets that self-intersect (bottom-middle).

Flexibility. The algorithm has been tested on a variety of data from urban and indoor structures to mechanical pieces through free-form objects (see Figure 5.11 and Figure 5.12 for visual results). Although it performs best on piecewise-planar objects and scenes, our algorithm can handle a large number of primitives necessary to approximate free-form shapes at different levels of detail, as illustrated in Figure 5.14. Different types of acquisition systems have also been used to generate the datasets, including Laser, *e.g. Euler*, *Hand* and *Lans*, multi-view stereo, *e.g. Cottage*, *Building block*, *Capron*, *Block 1* and *Block 2*, sampling points from CAD model, *e.g. Chair* and Kinect, *e.g. Rubbish bin* and *Couch*. Because the data term of our energy measures surface consistency with respect to primitives directly, our algorithm is weakly affected by the type of acquisition systems as long as primitives fit well to input data. Moreover, we also perform our approach on LOD generation of urban scene as shown in Figure 5.13. The output polygonal mesh of each building at LOD1 and LOD2 are highly consistent with CityGML formalism [GP12].

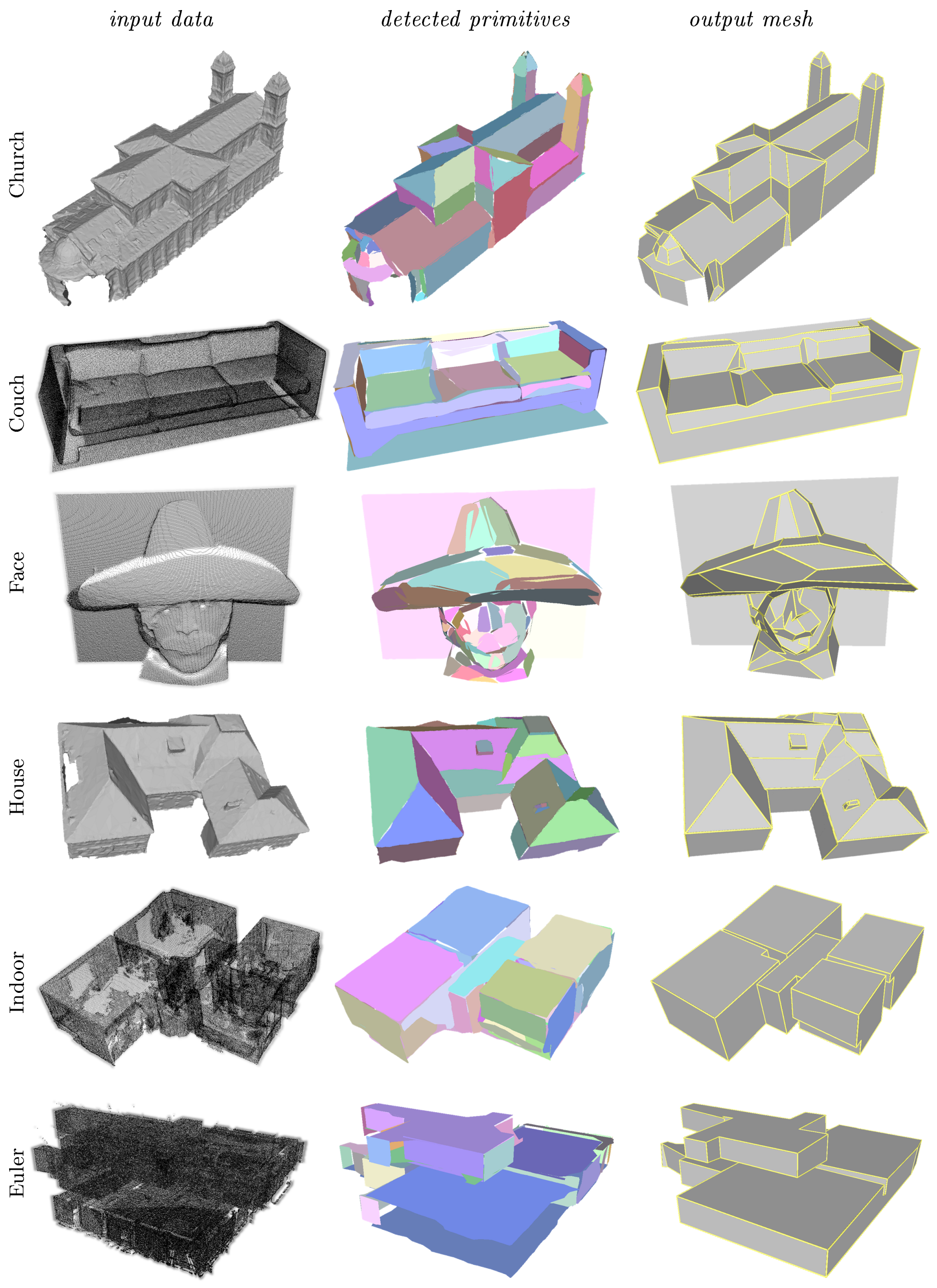


Figure 5.11: Results on different man-made objects and urban scenes. Our algorithm offers a good versatility by operating on different types of objects and scenes without any specific geometric assumption. Note, in particular, that our configurations of primitives are neither regularized nor filtered.

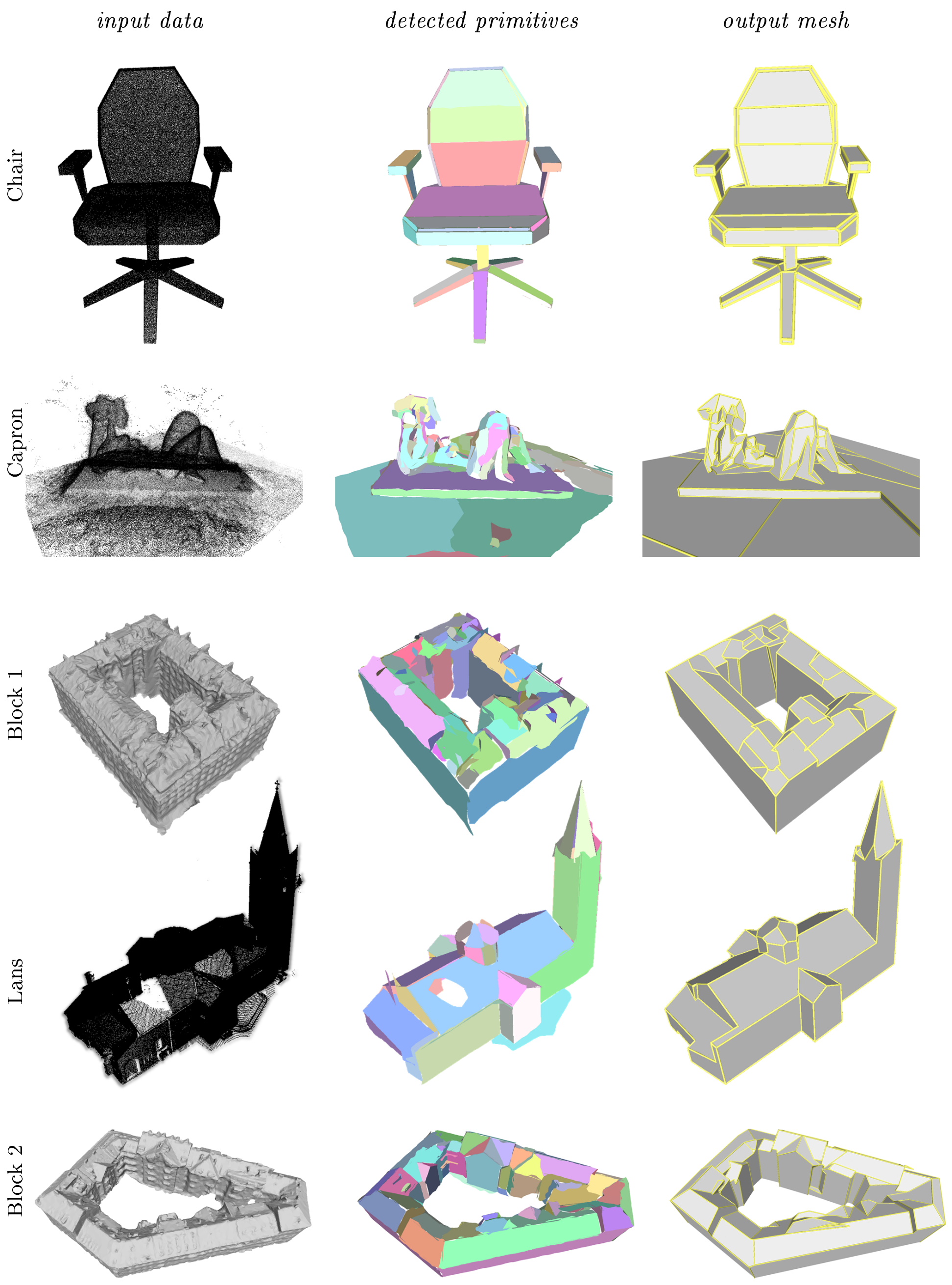
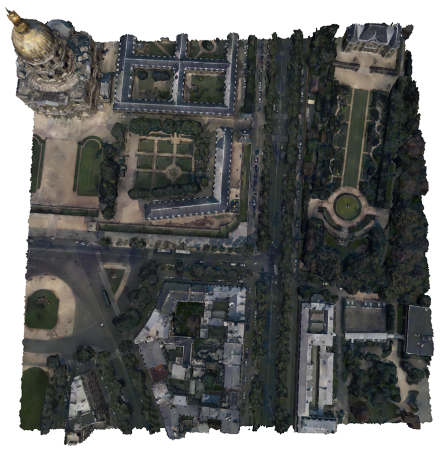
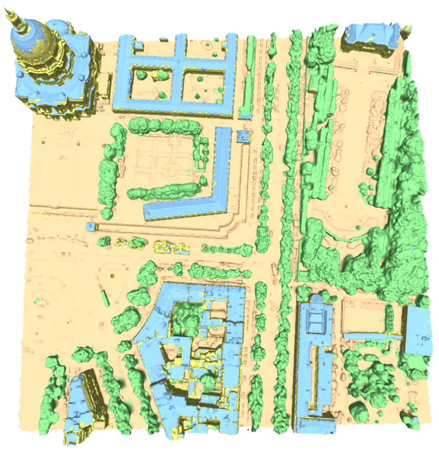
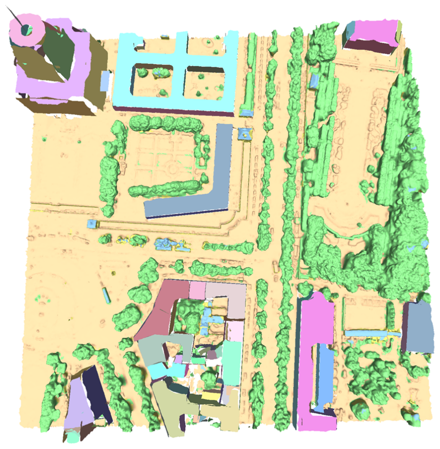
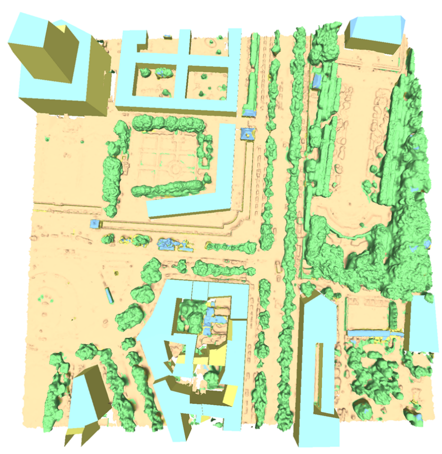


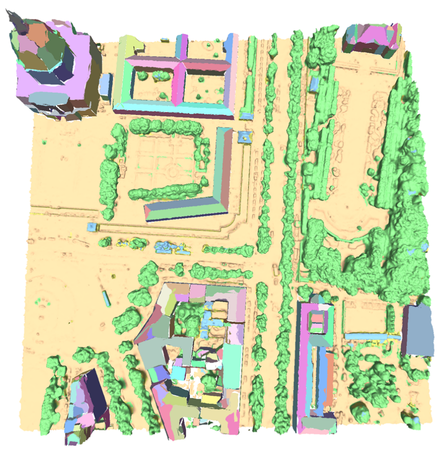
Figure 5.12: More visualization results on different man-made objects. Our algorithm successfully assembling the detected planes into a polygonal mesh, where each polygon preserves a meaningful part of the object.


(a) input mesh


(b) segmentation result by [VLA15]


(c) LOD1 plane detection


(d) LOD1 mesh generation


(e) LOD2 plane detection

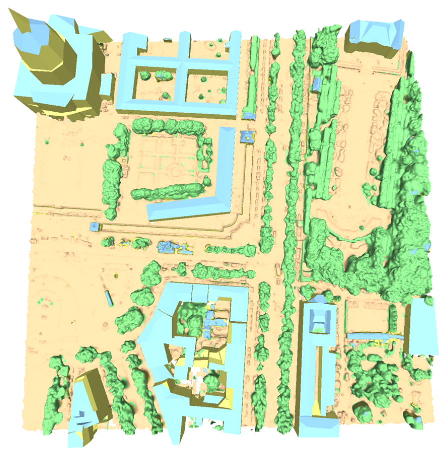

(f) LOD2 mesh generation

Figure 5.13: LOD generation of *urban scene*. Starting from original mesh (a), we first get segmentation results (b) and extract the individual buildings composed of adjacent facets with label *roof* or *facade*. We then detect planes of each building at LOD1 and LOD2 and assemble them as the corresponding polygonal meshes.

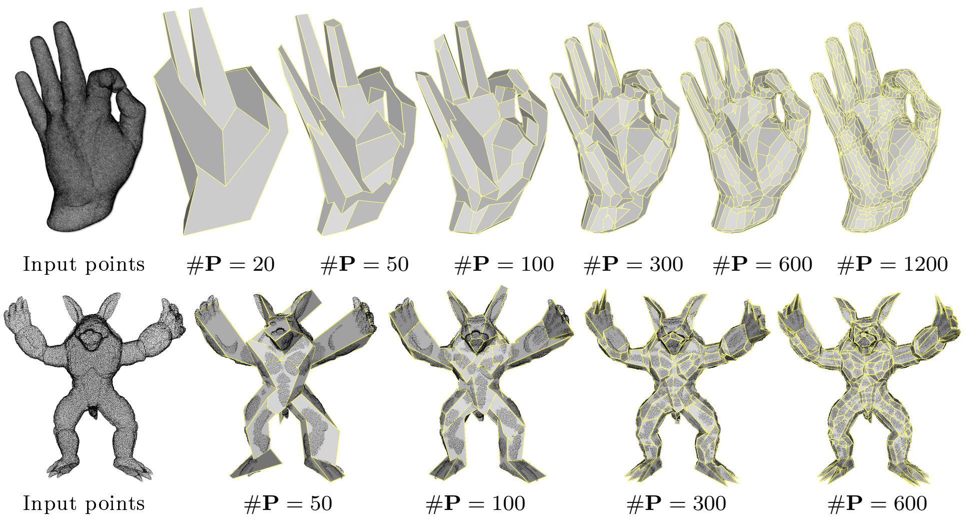


Figure 5.14: Reconstruction of a free-form object at different levels of details (top: *hand*, bottom: *Armadillo*). Our algorithm can be used to approximate free-form objects by piecewise planar representations. By detecting primitives with an increasing precision, we produce a set of polygonal meshes at different levels of detail. The evolution of the geometric error of to input of *hand* in function of the output complexity is given in Figure 5.15.

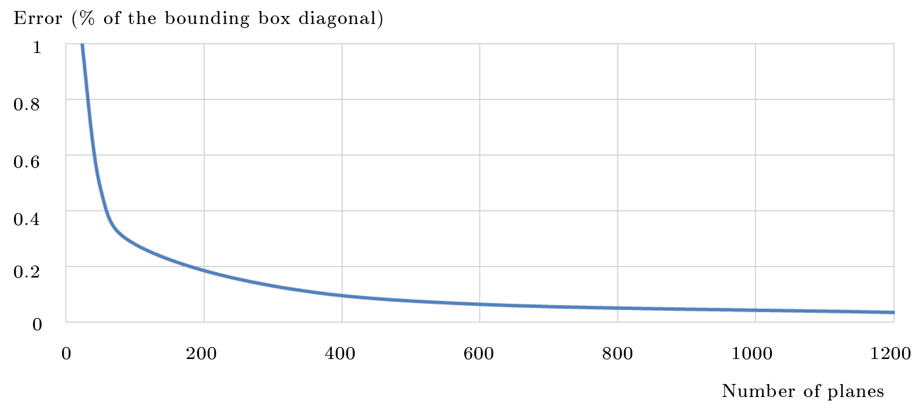


Figure 5.15: Error vs complexity on the *Hand* model (Figure 5.14). We obtain a good trade-off between geometric error and output complexity when approximatively one hundred primitives are detected. The error is measured as the Hausdorff distance from input points to output surface.

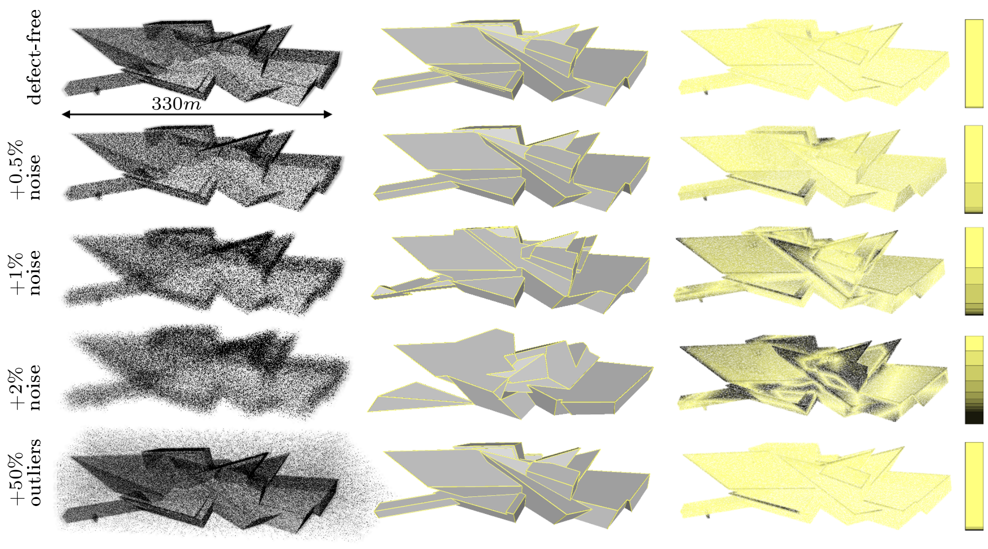


Figure 5.16: Robustness to noise and outliers on dataset *Museum*. Our output surface meshes (middle) are weakly affected by noise as long as primitive detection can capture the main planar components of the object. When adding 2% of noise (expressed w.r.t. the 3D bounding box diagonal), primitives are no longer correctly detected. Yellow-to-black colored points (bottom) represent the Hausdorff distance from the defect-free point cloud to output surface (yellow $= 0m$, black $\geq 8m$).

Robustness. As illustrated in Figure 5.16, our algorithm is relatively robust to noise as long as primitives can be decently detected. When data contain holes and missing areas as in *Euler* in Figure 5.11, the connectivity analysis typically returns few structural facets, but the subsequent slicing mechanism achieves to fill in the missing areas. In practice, our algorithm cannot handle large holes for which an extension of detected primitives is not sufficient to describe the missing part. Globally speaking, the exploitation of defect-laden data requires to increase the value of ε from 0.1 (default value) to typically 0.3.

Performance. Our algorithm is designed to be scalable and fast through two keys ingredients: a connectivity analysis to quickly process obvious assembling situations, and a slicing mechanism operated on softly-connected primitives only. Figure 5.17 shows the impact of these two ingredients on output complexity and running time. Used simultaneously, they allows us to strongly reduce running time from 43 minutes to 7 seconds on the shown

	Mechanical	*Fandisk*	*Church*	*Euler*	*Indoor*	*Couch*	*Face*
input size	382K	27K	18K	4M	186K	460K	144K
#primitives	60	45	80	45	50	50	70
#structural facets	35	36	5	1	10	3	3
#facets in $\mathcal{F}$	2.2K	213	1.9K	11.2K	3.1K	2.6K	6.6K
output complexity	298	89	291	1.1K	401	381	876
connectivity analysis (sec)	1.8	1.2	0.9	9	1	1.2	1.4
space partitioning (sec)	2.5	1	1.8	3	2	1.7	3
surface extraction (sec)	2.4	2	15	684	108	18	23
memory peak (Mb)	63	36	186	738	201	194	269

	House	*Museum*	*Hand-20*	*Hand-100*	*Hand-300*	*Hand-1200*	*Armadillo 600*
input size	16K	129K	369K	369K	369K	369K	173K
#primitives	51	75	20	100	300	1200	600
#structural facets	5	10	0	8	23	84	31
#facets in $\mathcal{F}$	3.1K	5.5K	429	7.3K	39K	71K	51.3K
output complexity	376	637	79	875	3747	7759	7281
connectivity analysis (sec)	0.3	1.6	1.5	2.7	11.1	21.9	13.2
space partitioning (sec)	2.2	0.9	0.6	4.2	21	81.8	50.4
surface extraction (sec)	62	5	10	45	286	529	282
memory peak (Mb)	201	167	45	141	546	741	1376

Table 5.1: Performance on some reconstructed models in terms of running time and memory consumption. The output complexity is expressed in number of active facets returned by the surface extraction solver.

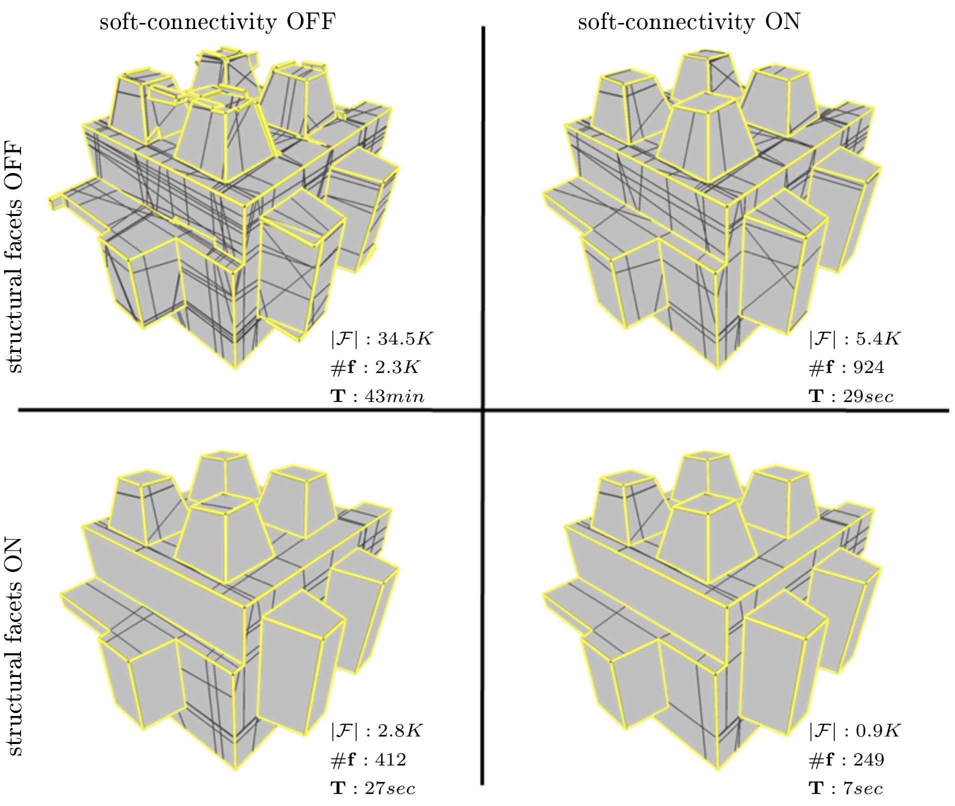


Figure 5.17: Ablation study. Without soft-connectivity ($\varepsilon = 1$) and connectivity analysis step, the number of facets $|\mathcal{F}|$ in the partition is huge, leading to high running times $\mathbf{T}$ and a complex output surface likely to contain artifacts (top left). Activating soft-connectivity ($\varepsilon = 0.1$) reduces the complexity of the partition while improving the quality of the output surface (top right). When a fair number of structural facets are detected during the connectivity analysis step (here 35 structural facets over 60 primitives), the partition is even more compact as only a part of primitives are sliced (bottom right). The facets in the output sufaces are represented through yellow (border) and black (internal) edges, and their number is given by #$\mathbf{f}$.

example. As illustrated in Table 5.1, running time does not depend only on the size of input data and the number of primitives, but also on the amount of structural facets. The latter is high typically on data weakly corrupted by defects and with few free-form components. In such cases, the algorithm is faster.

	Cottage				*Stanford bunny*			
	#**P**	$\|\mathcal{F}\|$	#**f**	**T**(s)	#**P**	$\|\mathcal{F}\|$	#**f**	**T**(s)
Structuring [LA13]	19	272K	34K	21	100	63K	12K	6
PCC [CLP10]	21	3.7K	288	8	100	165K	10.5K	58
Polyfit [NW17]	23	1.8K	112	19	100	147K	5.6K	2449
Ours	21	0.9K	83	8	100	5.7K	0.6K	22

	Rubbish bin				*Building block*			
	#**P**	$\|\mathcal{F}\|$	#**f**	**T**(s)	#**P**	$\|\mathcal{F}\|$	#**f**	**T**(s)
Structuring [LA13]	100	115K	20K	11	150	360K	43K	7
PCC [CLP10]	100	211K	7.9K	91	150	651K	22K	387
Polyfit [NW17]	30	3.9K	0.5K	57	54	6.4K	0.9K	1267
Ours	100	8.7K	1.2K	14	150	12.4K	1.2K	126

Table 5.2: Quantitative evaluation for models presented on Figures 5.18 and 5.19. #**P**, $|\mathcal{F}|$ #**f** and **T** refer to the number of primitives, the number of candidate facets in $\mathcal{F}$, the number of facets in the output model, and the running time respectively.

Comparisons. We compared our algorithm with the connectivity-based method Structuring [LA13] and the slicing-based methods Polyfit [NW17] and Polyhedral Cell Complex (PCC) [CLP10]. For the latter, no primitive has been artificially added along vertical and horizontal axes in order to fairly compare the assembling mechanisms. As illustrated in Figure 5.18 and Table 5.2, we deliver similar results than Polyfit and PCC on simple examples requiring few primitives as Cottage, while being slightly faster. On more challenging datasets where hundred primitives are necessary to decently approximate the objects as *Rubbish bin*, our algorithm performs better in terms of visual quality, output complexity and running time. Structuring is fast and scalable but the mixture of large polygonal facets with fine triangular meshes leads to complex output models which is not a simple assembling of planes. Polyfit and PCC which rely on greedy slicing mechanisms are relatively slow and have memory consumption problems. In particular, we reduced the number of primitives for Polyfit on *Rubbish bin* so that the algorithm could run in reasonable time. PCC and Polyfit produce models

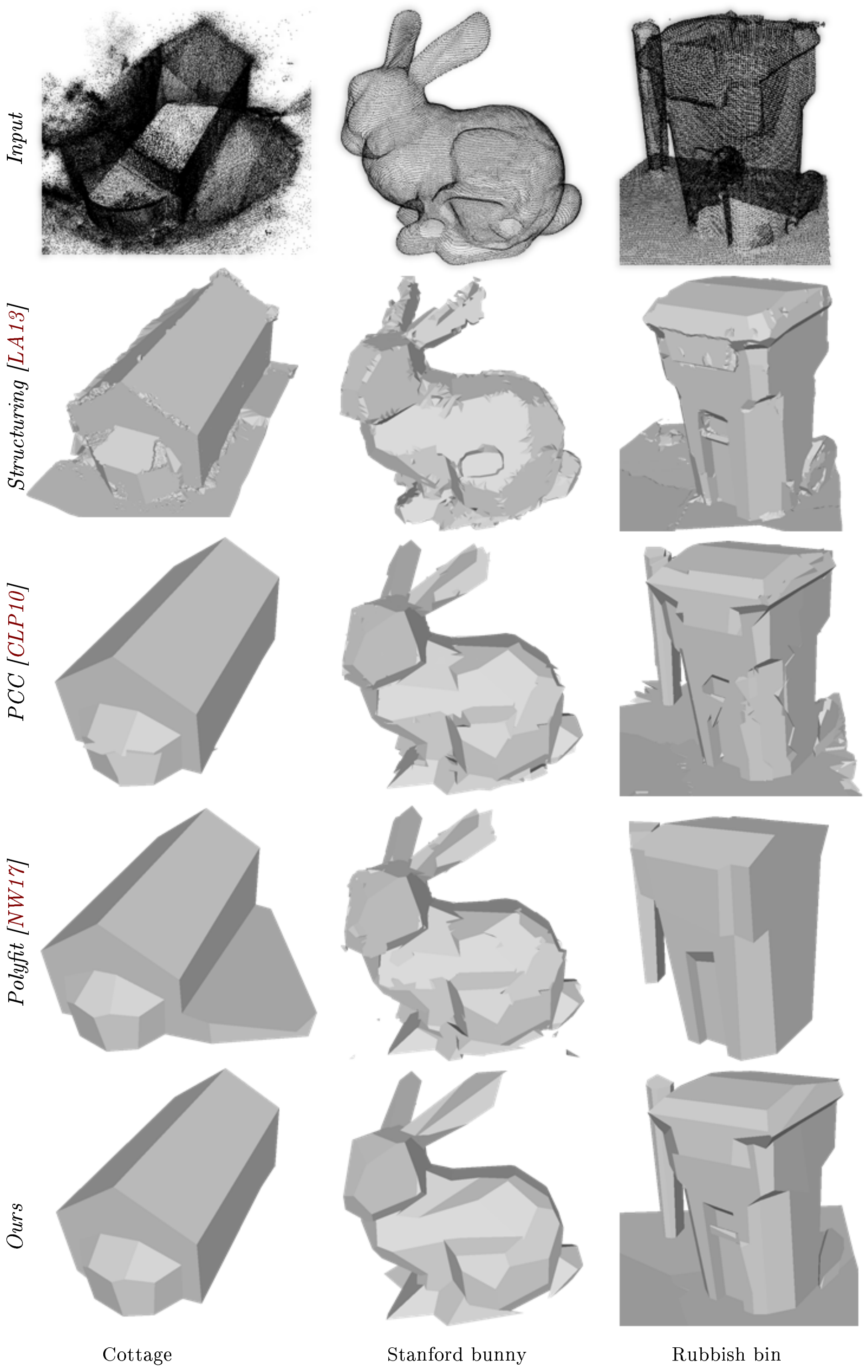


Figure 5.18: Visual comparison with state-of-the-art methods. Given similar configurations of primitives, our algorithm produces artifact-free models with a lower complexity in shorter running times than Structuring, PCC and Polyfit as illustrated in Table 5.2.

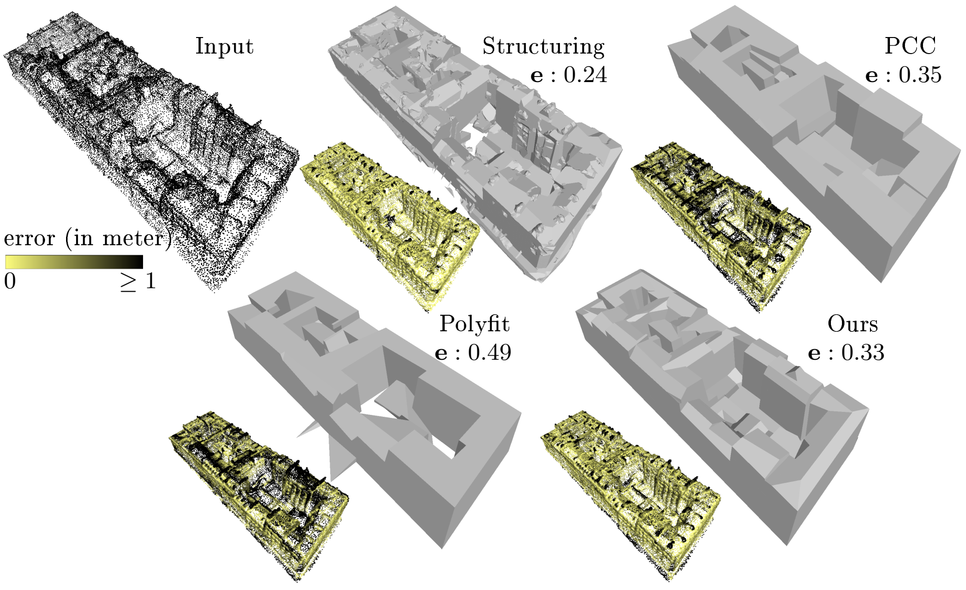


Figure 5.19: Geometric accuracy on dataset *Building block*. The yellow-to-black colored points represents the Hausdorff distance from the input points to output surface. Structuring obtains the best RMS error **e**, but the model is not compact as underlined in Table 5.2. Our error is the second best, outclassing PCC and Polyfit which are penalized by dense space partitions and data consistency terms affected by noisy input points.

with visual artifacts when approximating the free-form shapes of *Standford bunny*. On such an object, many planes share almost straight angles, leading to a mislabeling of facets or cells when the set of candidates is huge (the number of candidate facets for PCC and Polyfit is approximatively 30 times higher than with our method) and the data consistency term of the labeling energy does not rely on primitives only. Yet, these methods do not offer a special treatment to scalability, but rather focus on improving the quality of primitives. Figure 5.19 shows the geometric accuracy of these methods on a complex block of buildings. Our algorithm outclasses Polyfit and PCC while being faster and delivering a more compact output model. More precise, we argue that our algorithm improves the performance and the quality of output from three aspects: (i) connectivity analysis recovers the border of certain planes in a short time (ii) soft-connectivity highly decreases the number of candidate facets in the solution space (iii) the designed constraints in Eq. 5.6 and 5.7 help exploring the solution space by avoiding the solver returning unpromising solutions.

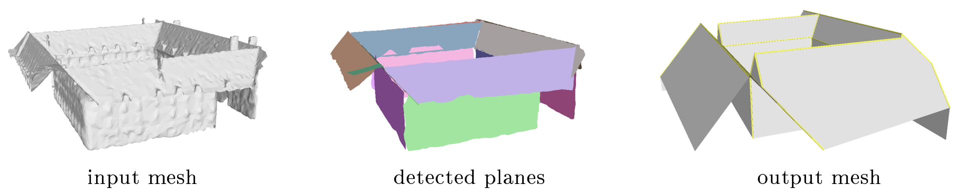


Figure 5.20: Failure case. Given the planes detected from an input mesh with huge missing parts (i.e. facades in front view), our algorithm fails recovering the missing parts.

Limitations. Our work focuses on primitive assembling, not on primitive detection or primitive completion. As a result, if primitives are badly detected from input points, we do not offer a special treatment to repair them, contrary to PCC or Polyfit. This typically happens when inputs contain large missing parts. When no detected primitive can decently fill in the missing parts, our algorithm typically shrinks the surface as shown in Figure 5.20. PCC which artificially adds primitives on these parts along vertical and horizontal directions is then a more suitable choice. Similarly, Polyfit delivers more regularized surfaces for simple objects thanks to its filtering of primitives. Yet, the primitive treatments of these two methods could be also employed with our work in order to rectify our input primitives. Also, the connectivity analysis step typically retrieves less structural facets from defect-laden data, as shown in Table 5.1. The performances of our method are then reduced in this case. Globally speaking, one could choose to detect more structural facets, potentially wrong ones, and to let the surface extraction solver selecting them is a more robust alternative. This would lead to a much more complex partitioning data structure where all planes should be inserted. This would thus significantly reduce scalability and increase running times with respect to our strategy.

5.6 Conclusion

We proposed a polygonal surface reconstruction algorithm from 3D data that specifically addresses the scalability issue existing in the field. The algorithm is built on several key technical ingredients that allows us to operate on an efficient and compact partitioning data-structure. We proposed (i) the principle of soft-connectivity that avoids slicing improbable pairs of primitives, (ii) an analysis of the connectivity of primitives in order to quickly solve obvious primitive assembling situations, and (iii) a surface extraction energy which estimates the quality of a solution without operating time-consuming measurements to input 3D data. Our algorithm outperforms state-of-the-art methods on challenging input data in terms of performance and output complexity.

CHAPTER 6

Conclusion and perspectives

6.1 Conclusion

In this thesis, we investigated the problem of generating CAD-style models from raw 3D data by proposing 3 contributions.

Semantic segmentation of 3D data. In Chapter 3, we developed a pyramid structure network for deep feature learning from raw point cloud. Being concatenated after state-of-the-art baseline models, our 3d-PSPNet aggregated multi-scale sub-regional contextual features with local features and produce better semantic segmentation results than only using the baseline models on three public datasets. Experimental results proved our idea that enlarging the receptive field of each point is a crucial step to enrich the pointwise feature. We also investigated several ablation studies and chose the best hyperparameters for our architecture to balance segmentation accuracy and computational cost. In summary, this step enables us to extract individual objects from complex scenes and work on each of them afterwards.

The deep learning techniques applied on raw point cloud is a new field in 3D Computer Vision. We believe more sophisticated architectures linking the points at different neighborhoods will capture richer contextual clues and produce better prediction results. So far, we did not test our methodology on dense urban mesh, where current main obstacle is the lack of public datasets with accurate ground truth label of each triangle facet. We believe that with the availability of more public datasets, the study of deep learning techniques on dense mesh will be more commonly used in the future.

Shape detection on multiple structural scales. In Chapter 4, we explored a mechanism to automatically extract a set of plane configurations that capture the shape and structure of man-made objects at different key abstraction levels. The intuition of the whole framework is that the structural plane configurations are more likely to be located along the diagonal of a two-dimensional parameter space. Following this observation, we first proposed a simple but robust priority metrics to generate a trajectory along the diagonal direction of this parameter space by shape collapse. Second, we designed a learning-based technique to characterize each configuration and detect existing structural formalisms such as the CityGML Levels Of Details. The extracted higher-level geometric shapes not only decrease data complexity but also provides a set of clean intermediate representation for further use.

Our current framework is designed based on a collapse-then-detection strategy. In the shape collapse procedure, we chose the Euclidean distance as priority metrics after extensive experimental evaluation as a good compromise between accuracy and performance. This priority metrics works well on highly-structured objects such as buildings. However, it might miss some structural scales for objects that are not so-well structured such as free-form objects. This leads to a wrong scale detection result if the collapsing trajectory is not correct. Hence, we would like to utilize scale detection results to instruct the shape collapse process in an on-the-fly way. More precisely, a possible solution is to learn a metrics from a training set of trajectories that can enable us to explore and track the structural scales in the parameter space in a more refined way.

Polyhedral surface reconstruction. In Chapter 5, we designed a hybrid algorithm to assemble the isolated planar shapes into a CAD-style model which is compact and structure-aware. We mainly addressed the scalability issue of the proposed method with respect to input planes. We firstly processes partial input planes by analyzing the corresponding adjacency graph. This efficient step enabled us to quickly recover the border shapes of several planes and embed them as geometric constraints to reconstruct the final surface model. After that, we sliced the remaining planes by a soft-connectivity mechanism and formulated the surface reconstruction approach as a constrained integer programming problem. Experimental results illustrated that the whole pipeline generates high quality CAD-style models ranging from free-form shapes to man-made objects in an efficient way. We argued that the improvement of performance comes from two aspects: (i) efficient graph analysis processed partial primitives (ii) soft-connectivity mechanism highly reduced the number of variables for optimization problem.

In this part, the parameter ε specifying the soft-connectivity relationship plays an important role in controlling how far the connected primitives can be located from each other to guarantee a correct reconstruction. This mechanism recovers the missing intersection information efficiently and produces light partitioning results for the downstream processing. In future work, we would like to investigate on the automatic selection of ε for each planes. We also wish to understand the hierarchical relationships between primitives in order to detect and utilize high order structural information as symmetry to avoid some meaningless intersection calculation.

6.2 Perspectives

The contributions of this PhD work constitute only a tiny step towards the automatic reconstruction of objects in the form of a CAD-style model. The quality of output models delivered in this work is still far from outperforming the one of real CAD models. Several research directions can be explored to reduce this gap.

Mixing data modalities. Our current framework is designed to handle 3D raw data collected by different kinds of sensors. This general pipeline mainly focus on geometric attributes in each step. However, some data acquisition techniques also record other attributes such as color information from RGB-D sensors and intensity from LiDAR scanners. Adapting those features into our current framework will improve both shape detection and surface reconstruction step. For instance, an appropriate balance between both geometric and photometric contributions to the priority metrics in Chapter 4 will decrease the uncertainty while exploring the scale space. We can also improve the data consistency term of each candidate facet defined in 5.2 while considering the photometric distance between each point to the corresponding facet.

Mixing shape modalities. Our system is proposed according to the assumption that the surface of man-made objects can be approximated by linear geometric shapes. This restrictive use of geometric primitives facilitates the design of geometric modeling algorithm and makes it efficient to process large number of planes. However, there are large number of objects preserving free-form shapes, which can be approximated by other geometric primitives like cylinders, spheres or even more complex parametric functions such as NURBS. Detecting and modeling the geometric relationships between different kinds of primitives is the key problem to be addressed. In addition, current 3D arrangement solutions can not handle non-convex shapes.

3D object detection. Another way to extract objects-of-interest from complex scenes can be done by 3D object detection. This hot topic has attracted more attention recently with the development of advanced deep learning techniques. The main difficulty is to extract relevant features from sparse point clouds and missing parts in case of object occlusions. Those issues impact gravely the feature learning across different neighborhoods. One potential future work is to design more intelligent network structures that can recover the missing knowledge by discovering the relationships between points in each sub-region. We believe this direction will promote rapidly with the availability of more public datasets.

More efficient space partitioning methods. We proposed a soft-connectivity mechanism in Chapter 5 to detect the collision between planar primitives. This choice decreases the computational cost compared with typical hard-connectivity strategy and leads to a lighter data-structure. However, current space partitioning mechanism loses magic facing large scale objects which can be approximated by millions of planes. Designing more intelligent collision detection approach is a future path to handle this large number of planes.

Joint semantic and geometric reconstruction. In this thesis, we reconstruct the surface of scanned data mainly in a geometric level, resulting in a lack of semantic information. However, semantic-aware reconstruction results bring lots of benefits to downstream applications. For instance, robots will interact with its environments better if it can recognize the semantic meaning of objects around it. To achieve this goal, existing approaches [HZC+13, SHP+16] are designed to integrate semantic information with geometric modeling and reconstruct the world in a joint way. However, these methods can not handle large scale urban scenes and output dense meshes that are not structure-aware. We believe extended work can be done by incorporating both scalability and structure-aware issues.

List of publications:

Planar shape detection at structural scales [FLD18]
Hao Fang, Florent Lafarge, Mathieu Desbrun
IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018

Fast and scalable assembling of planar primitives into polygonal meshes.
Hao Fang, Florent Lafarge
Submitted to IEEE Transaction on Pattern Analysis and Machine Intelligence

Pyramid scene parsing network in 3D: improving semantic segmentation of point clouds with multi-scale contextual information
Hao Fang, Florent Lafarge
Submitted to ISPRS Journal of Photogrammetry and Remote Sensing.

Bibliography

[ASF$^+$13] M. Arikan, M. Schwarzler, S. Flory, M. Wimmer, and S. Maierhofer. O-snap: Optimization-based snapping for modeling architecture. *Trans. on Graphics*, 32(1), 2013. (Cited on pages 5, 10 and 23.)

[ASZ$^+$16] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on pages 15, 33 and 34.)

[BdLGM14] Alexandre Boulch, Martin de La Gorce, and Renaud Marlet. Piecewise-planar 3d reconstruction with edge and corner regularization. *Computer Graphics Forum*, 33(5), 2014. (Cited on pages 9, 10 and 24.)

[BGLSA18] Alexandre Boulch, Joris Guerry, Bertrand Le Saux, and Nicolas Audebert. Snapnet: 3d point cloud semantic labeling with 2d deep segmentation networks. *Computers & Graphics*, 71:189–198, 2018. (Cited on pages 8 and 16.)

[BKC15] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015. (Cited on pages 8 and 16.)

[BLS16] Filip Biljecki, Hugo Ledoux, and Jantien Stoter. An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59:25–37, 2016. (Cited on pages 6 and 7.)

[BMP01] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *Advances in neural information processing systems*, pages 831–837, 2001. (Cited on page 13.)

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. (Cited on page 55.)

[BSVG15] H. Bodis-Szomoru, Riemenschneider and L. Van Gool. Superpixel meshes for fast edge-preserving surface reconstruction. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2015. (Cited on page 25.)

[BTS+17] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gael Guennebaud, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. A survey of surface reconstruction from point clouds. *Computer Graphics Forum*, 36(1), 2017. (Cited on page 7.)

[BZSL13] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. (Cited on page 17.)

[CC08] J. Chen and B. Chen. Architectural modeling from sparsely scanned range data. *IJCV*, 78(2-3), 2008. (Cited on pages 5, 10, 20 and 23.)

[CF14] R. Cabral and Y. Furukawa. Piecewise planar and compact floorplan reconstruction from images. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2014. (Cited on page 25.)

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996. (Cited on page 22.)

[CLP10] A.-L. Chauve, P. Labatut, and J.-P. Pons. Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *CVPR*, 2010. (Cited on pages 9, 10, 24, 49, 60, 62, 74, 85 and 86.)

[CMW+17] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *IEEE CVPR*, volume 1, page 3, 2017. (Cited on page 19.)

[CPK+18] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018. (Cited on pages 8 and 16.)

[CSAD04] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. In *Siggraph*, 2004. (Cited on page 23.)

[CSM12] Peter Carr, Yaser Sheikh, and Iain Matthews. Monocular object detection using 3d geometric primitives. In *ECCV*, 2012. (Cited on page 9.)

[CY00] J. Coughlan and A. Yuille. The manhattan world assumption: Regularities in scene statistics which enable bayesian inference. In *NIPS*, 2000. (Cited on page 25.)

[DBI18] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on page 16.)

[DCS+17] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on page 33.)

[DH72] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972. (Cited on page 20.)

[DN18] Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. *arXiv preprint arXiv:1803.10409*, 2018. (Cited on page 16.)

[EKS83] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *Trans. on Information Theory*, 29(4), 1983. (Cited on page 67.)

[FB81] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. (Cited on page 19.)

[FCSS09] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Reconstructing building interiors from images. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 80–87. IEEE, 2009. (Cited on page 22.)

[FLD18] Hao Fang, Florent Lafarge, and Mathieu Desbrun. Planar shape detection at structural scales. In *Proc. of Computer*

Vision and Pattern Recognition (CVPR), 2018. (Cited on page 95.)

[FMMCAJ13] Eduardo Fernandez-Moral, Walterio Mayol-Cuevas, Vicente Arevalo, and Javier Gonzalez Jimenez. Fast place recognition with plane-based maps. In *ICRA*, 2013. (Cited on page 9.)

[FTAB17] Engelmann Francis, Kontogianni Theodora, Hermans Alexander, and Leibe Bastian. Exploring spatial context for 3d semantic segmentation of point clouds. In *Proc. of International Conference on Computer Vision (ICCV), Workshop*, 2017. (Cited on pages 33 and 40.)

[GCO06] Ran Gal and Daniel Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Transactions on Graphics (TOG)*, 25(1):130–150, 2006. (Cited on page 14.)

[GFK$^+$18] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation. *arXiv preprint arXiv:1802.05384*, 2018. (Cited on page 18.)

[GFP10] D. Gallup, J.-M. Frahm, and M. Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2010. (Cited on page 25.)

[GKF09] Aleksey Golovinskiy, Vladimir G. Kim, and Thomas Funkhouser. Shape-based recognition of 3D point clouds in urban environments. In *ICCV*, 2009. (Cited on page 14.)

[GO16] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2016. (Cited on page 76.)

[GP12] G. Groger and L. Plumer. Citygml – interoperable semantic 3d city models. *Journal of Photogrammetry and Remote Sensing*, 71, 2012. (Cited on pages 52, 56 and 77.)

[HBL15] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. (Cited on page 17.)

[HFBM13] Peter Henry, Dieter Fox, Achintya Bhowmik, and Rajiv Mongia. Patch volumes: Segmentation-based consistent mapping with rgb-d cameras. In *3D Vision-3DV 2013, 2013 International Conference on*, pages 398–405. IEEE, 2013. (Cited on page 5.)

[HFL12] Ruizhen Hu, Lubin Fan, and Ligang Liu. Co-segmentation of 3d shapes via subspace clustering. In *Computer graphics forum*, volume 31, pages 1703–1713. Wiley Online Library, 2012. (Cited on page 15.)

[HJS+14] J. Huang, T. Jiang, Z. Shi, Y. Tong, H. Bao, and M. Desbrun. l_1-based construction of polycube maps from complex shapes. *Trans. on Graphics*, 33(3), 2014. (Cited on page 25.)

[HK12] Martin Habbecke and Leif Kobbelt. Linear analysis of nonlinear constraints for interactive geometric modeling. In *Computer Graphics Forum*, volume 31, 2012. (Cited on page 25.)

[HKLP09] Vu Hoang Hiep, Renaud Keriven, Patrick Labatut, and Jean-Philippe Pons. Towards high-resolution large-scale multi-view stereo. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1430–1437. IEEE, 2009. (Cited on pages 3 and 5.)

[Hou62] Paul VC Hough. Method and means for recognizing complex patterns, December 18 1962. US Patent 3,069,654. (Cited on page 20.)

[HSKK01] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212. ACM, 2001. (Cited on page 14.)

[HSL+17] Timo Hackel, Nikolay Savinov, Lubor Ladicky, Jan D Wegner, Konrad Schindler, and Marc Pollefeys. Semantic3d. net: A new large-scale point cloud classification benchmark. *arXiv preprint arXiv:1704.03847*, 2017. (Cited on pages 8 and 16.)

[HSSM14] Rostislav Hulik, Michal Spanel, Pavel Smrz, and Zdenek Materna. Continuous plane detection in point-cloud data based on 3d hough transform. *Journal of Visual Communication and Image Representation*, 25(1):86–97, 2014. (Cited on page 20.)

[HWN18] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3d segmentation of point clouds. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on pages 8 and 18.)

[HWS16] Timo Hackel, Jan D Wegner, and Konrad Schindler. Fast semantic segmentation of 3d point clouds with strongly varying density. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 3(3), 2016. (Cited on page 14.)

[HZC$^+$13] Christian Hane, Christopher Zach, Andrea Cohen, Roland Angst, and Marc Pollefeys. Joint 3d scene reconstruction and class segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 97–104, 2013. (Cited on page 94.)

[HZvK$^+$15] Ruizhen Hu, Chenyang Zhu, Oliver van Kaick, Ligang Liu, Ariel Shamir, and Hao Zhang. Interaction context (icon): towards a geometric functionality descriptor. *ACM Transactions on Graphics (TOG)*, 34(4):83, 2015. (Cited on page 15.)

[IKH$^+$11] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011. (Cited on page 22.)

[ISS17] Hamid Izadinia, Qi Shan, and Steven M Seitz. Im2cad. In *CVPR*, 2017. (Cited on page 26.)

[JH99] Andrew E Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5):433–449, 1999. (Cited on pages 13 and 14.)

[KAMC17] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3d shape segmentation with projective convolutional networks. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on pages 8 and 16.)

[KAZB18] A. Kaiser, J. Alonso, Y. Zepeda, and T. Boubekeur. A survey of simple geometric primitives detection methods for captured 3d data. *Computer Graphics Forum*, 37, 2018. (Cited on page 19.)

[KBH06] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Symposium on Geometry Processing*, 2006. (Cited on page 22.)

[KH13] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):29, 2013. (Cited on page 22.)

[KHS10] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3d mesh segmentation and labeling. *ACM Transactions on Graphics (TOG)*, 29(4):102, 2010. (Cited on page 14.)

[KL17] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proc. of International Conference on Computer Vision (ICCV)*, 2017. (Cited on page 17.)

[KML$^+$17] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3d proposal generation and object detection from view aggregation. *arXiv preprint arXiv:1712.02294*, 2017. (Cited on page 19.)

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2012. (Cited on page 16.)

[LA13] Florent Lafarge and Pierre Alliez. Surface reconstruction through point set structuring. In *Computer Graphics Forum*, volume 32, 2013. (Cited on pages 5, 10, 20, 23, 85 and 86.)

[LBF14] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3d scene labeling. In *ICRA*, 2014. (Cited on page 56.)

[LC87] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987. (Cited on page 22.)

[LM11] Florent. Lafarge and Clément. Mallet. Building large urban environments from unstructured point data. In *Proc. of International Conference on Computer Vision (ICCV)*, 2011. (Cited on page 25.)

[LM12] Florent Lafarge and Clément Mallet. Creating large-scale city models from 3d-point clouds: a robust approach with hybrid representation. *International journal of computer vision*, 99(1):69–85, 2012. (Cited on pages 8, 14 and 21.)

[LNSW16] Minglei Li, Liangliang Nan, Neil Smith, and Peter Wonka. Reconstructing building mass models from uav images. *Computers & Graphics*, 54:84–93, 2016. (Cited on page 5.)

[LPK07] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007. (Cited on page 22.)

[LPK09a] Patrick. Labatut, J.-P. Pons, and R. Keriven. Hierarchical shape-based surface reconstruction for dense multi-view stereo. In *ICCV workshops*, 2009. (Cited on page 23.)

[LPK09b] Patrick Labatut, J-P Pons, and Renaud Keriven. Robust and efficient surface reconstruction from range data. In *Computer graphics forum*, volume 28, pages 2275–2290. Wiley Online Library, 2009. (Cited on page 22.)

[LS18] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018. (Cited on page 18.)

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2015. (Cited on pages 8 and 16.)

[LWC$^+$11] Y. Li, X. Wu, Y. Chrysanthou, A. Sharf, D. Cohen-Or, and N. Mitra. Globfit: Consistently fitting primitives by discovering global relations. *Trans. on Graphics*, 30(4), 2011. (Cited on page 21.)

[LWN16] Minglei Li, Peter Wonka, and Liangliang Nan. Manhattan-world urban reconstruction from point clouds. In *Proc. of European Conference on Computer Vision (ECCV)*, 2016. (Cited on page 25.)

[LWYU18] Ming Liang, Shenlong Wang, Bin Yang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 641–656, 2018. (Cited on page 19.)

[MBBV15] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015. (Cited on page 17.)

[MGA$^+$17] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph*, 36(4):71, 2017. (Cited on page 17.)

[MMBM15] Aron Monszpart, Nicolas Mellado, Gabriel J Brostow, and Niloy J Mitra. Rapter: Rebuilding man-made scenes with regular arrangements of planes. *Trans. on Graphics*, 34(4), 2015. (Cited on pages 21, 60, 61 and 62.)

[MPM$^+$14] Oliver Mattausch, Daniele Panozzo, Claudio Mura, Olga Sorkine-Hornung, and Renato Pajarola. Object detection and classification from large-scale cluttered indoor scans. In *Computer Graphics Forum*, volume 33, pages 11–21. Wiley Online Library, 2014. (Cited on page 15.)

[MS15] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Proc. of International Conference on Intelligent Robots and Systems (IROS)*, 2015. (Cited on pages 8 and 16.)

[MWZ$^+$13] Niloy Mitra, Michael Wand, Hao Richard Zhang, Daniel Cohen-Or, Vladimir Kim, and Qi-Xing Huang. Structure-aware shape processing. In *SIGGRAPH Asia 2013 Courses*, page 1. ACM, 2013. (Cited on page 7.)

[MZL$^+$09] Ravish Mehra, Qingnan Zhou, Jeremy Long, Alla Sheffer, Amy Gooch, and Niloy J Mitra. Abstraction of man-made shapes. *ACM trans. Graph.*, 28(5), 2009. (Cited on page 9.)

[NBW12] Abdul Nurunnabi, David Belton, and Geoff West. Robust segmentation in laser scanning 3d point cloud data. In *Proc. of International Conference on Digital Image Computing Techniques and Applications (DICTA)*, 2012. (Cited on page 8.)

[NN07] J. Novatnack and K. Nishino. Scale-dependent 3D geometric features. In *ICCV*, 2007. (Cited on page 14.)

[NW17] L. Nan and P. Wonka. Polyfit: Polygonal surface reconstruction from point clouds. In *ICCV*, 2017. (Cited on pages 9, 10, 25, 74, 75, 85 and 86.)

[OLA16a] Sven Oesau, Florent Lafarge, and Pierre Alliez. Object classification via planar abstraction. In *Proc. of the ISPRS congress*, 2016. (Cited on pages 9, 14 and 20.)

[OLA16b] Sven Oesau, Florent Lafarge, and Pierre Alliez. Planar Shape Detection and Regularization in Tandem. *Computer Graphics Forum*, 35(1), 2016. (Cited on page 21.)

[PY09] C. Poullis and S. You. Automatic reconstruction of cities from remote sensor data. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2009. (Cited on page 25.)

[QLW$^+$17] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017. (Cited on page 18.)

[QSMG17] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on pages 8, 17, 18, 27, 28, 29 and 34.)

[QSN$^+$16] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on pages 8 and 16.)

[QYSG17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2017. (Cited on pages 8, 18, 28, 29 and 34.)

[QZN14] Rongqi Qiu, Qian-Yi Zhou, and Ulrich Neumann. Pipe-run extraction and reconstruction from point clouds. In *European Conference on Computer Vision*, pages 17–30. Springer, 2014. (Cited on page 20.)

[RBB09] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. Citeseer, 2009. (Cited on page 13.)

[RLA17] Mohammad Rouhani, Florent Lafarge, and Pierre Alliez. Semantic segmentation of 3D textured meshes for urban scene analysis. *ISPRS Journal of Photogrammetry and Remote Sensing*, 123, 2017. (Cited on page 15.)

[RUG17] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on page 17.)

[RVDH05] Tahir Rabbani and Frank Van Den Heuvel. Efficient hough transform for automatic detection of cylinders in point clouds. *Isprs Wg Iii/3, Iii/4*, 3:60–65, 2005. (Cited on page 20.)

[RvDHV06] T Rabbani, F van Den Heuvel, and G Vosselman. Segmentation of point clouds using smoothness constraint. *ISPRS*, 36(5), 2006. (Cited on pages 21, 48, 49, 50 and 67.)

[SDK09] Ruwen Schnabel, Patrick Degener, and Reinhard Klein. Completion and reconstruction with primitive shapes. In *Computer Graphics Forum*, volume 28, 2009. (Cited on page 24.)

[SFF11] F. Schindler, W. Forstner, and J.-M. Frahm. Classification and reconstruction of surfaces from point clouds of man-made objects. In *ICCV Workshops*, 2011. (Cited on pages 10 and 23.)

[SHP$^+$16] Nikolay Savinov, Christian Häne, Marc Pollefeys, et al. Semantic 3d reconstruction with continuous regularization and ray potentials using a visibility consistency constraint. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5460–5469. IEEE, 2016. (Cited on page 94.)

[SJS$^+$18] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on page 18.)

[SK17] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3693–3702, 2017. (Cited on page 17.)

[SMKF04] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. In *Shape Modeling International*, 2004. (Cited on pages 56 and 57.)

[SMKLM15] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2015. (Cited on pages 8 and 16.)

[SSBB15] Mojtaba Valinejad Shoubi, Masoud Valinejad Shoubi, Ashutosh Bagchi, and Azin Shakiba Barough. Reducing the operational energy demand in buildings using building information modeling tools and sustainability approaches. *Ain Shams Engineering Journal*, 6(1):41–55, 2015. (Cited on page 1.)

[SSS09] Sudipta N Sinha, Drew Steedly, and Richard Szeliski. Piecewise planar stereo for image-based rendering. In *ICCV*, 2009. (Cited on pages 9 and 25.)

[SWK07] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, 2007. (Cited on pages 19, 60, 61 and 67.)

[SWWK08] Ruwen Schnabel, Raoul Wessel, Roland Wahl, and Reinhard Klein. Shape recognition in 3d point-clouds. 2008. (Cited on page 19.)

[TB97] Alain Tremeau and Nathalie Borel. A region growing and merging algorithm to color segmentation. *Pattern recognition*, 30(7):1191–1203, 1997. (Cited on page 21.)

[TCA$^+$17] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *Proc. of International Conference on 3D Vision (3DV)*, 2017. (Cited on pages 8, 16 and 34.)

[The17] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.11 edition, 2017. (Cited on page 76.)

[TM14] L. Teran and P. Mordohai. 3D interest point detection via discriminative learning. In *ECCV*, 2014. (Cited on page 14.)

[TMT10] A. Toshev, P. Mordohai, and B. Taskar. Detecting and parsing architecture at city scale from range data. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010. (Cited on page 25.)

[Vit67] A.J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. on Information Theory.*, 13(2), 1967. (Cited on page 55.)

[VKH06] V. Verma, R. Kumar, and S. Hsu. 3D building detection and modeling from aerial LIDAR data. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2006. (Cited on page 25.)

[VKLP12] H. Vu, R. Keriven, P. Labatut, and J.P. Pons. High accuracy and visibility-consistent dense multi-view stereo. In *PAMI*, volume 34, 2012. (Cited on pages 56 and 57.)

[vKvLV11] M. van Kreveld, T. van Lankveld, and R. Veltkamp. On the shape of a set of points and lines in the plane. *Computer Graphics Forum*, 30, 2011. (Cited on page 23.)

[VLA15] Yannick Verdie, Florent Lafarge, and Pierre Alliez. LOD Generation for Urban Scenes. *Trans. on Graphics*, 34(3), 2015. (Cited on pages 7, 10, 15, 24, 52, 53, 61, 62, 74 and 80.)

[VTC05] George Vogiatzis, Philip HS Torr, and Roberto Cipolla. Multi-view stereo via volumetric graph-cuts. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 391–398. IEEE, 2005. (Cited on page 22.)

[WJM13] Martin Weinmann, Boris Jutzi, and Clément Mallet. Feature relevance assessment for the semantic interpretation of

3d point cloud data. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 5:W2, 2013. (Cited on page 8.)

[WPM$^+$14] Oliver J Woodford, Minh-Tri Pham, Atsuto Maki, Frank Perbet, and Björn Stenger. Demisting the hough transform for 3d shape recognition and registration. *International Journal of Computer Vision*, 106(3):332–341, 2014. (Cited on page 20.)

[WSK$^+$15] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2015. (Cited on pages 8 and 16.)

[WSL$^+$18] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018. (Cited on pages 8 and 18.)

[WYHN18] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on page 18.)

[XOT13] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3D: A database of big spaces reconstructed using sfm and object labels. In *ICCV*, 2013. (Cited on pages 56 and 57.)

[YLF$^+$18] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on page 18.)

[ZBKB08] L. Zebedin, J. Bauer, K.F. Karner, and H. Bischof. Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In *Proc. of European Conference on Computer Vision (ECCV)*, 2008. (Cited on page 25.)

[ZJM12] Zihan Zhou, Hailin Jin, and Yi Ma. Robust plane-based structure from motion. In *CVPR*, 2012. (Cited on page 9.)

[ZJRP$^+$15] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proc. of International Conference on Computer Vision (ICCV)*, 2015. (Cited on page 16.)

[ZLHW17] Qing Zhu, Yuan Li, Han Hu, and Bo Wu. Robust point cloud classification based on multi-level semantic relationships for urban scenes. *ISPRS Journal of Photogrammetry and Remote Sensing*, 129:86–102, 2017. (Cited on page 15.)

[ZN12] Qian-Yi Zhou and Ulrich Neumann. 2.5d building modeling by discovering global regularities. In *CVPR*, 2012. (Cited on pages 9 and 25.)

[ZSN+17] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on page 16.)

[ZSQ+17] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on pages 9 and 28.)