École des Ponts ParisTech

THÈSE DE DOCTORAT

de l'École des Ponts ParisTech

Approche conjointe géométrique et sémantique pour la reconstruction de maquettes numériques de bâtiments

École doctorale N° 405, Mathématiques & sciences et technologies de l'information et de la communication

Informatique

Thèse préparée au sein du Laboratoire d'Informatique Gaspard Monge (LIGM), UMR 8049, équipe A3SI (Imagine, Ecole des Ponts)

Thèse soutenue le 14 Décembre 2021, par

Pierre-Alain Langlois

Composition du jury :

Pierre, ALLIEZ Directeur de recherche, INRIA Sophia-Antipolis	*Président, Rapporteur*
Fréderic, JURIE Professeur, ENSICAEN & Scientific expert, Safran	*Rapporteur*
Tania, LANDES Maîtresse de conférences, INSA de Strasbourg	*Examinatrice*
Peter, WONKA Professor, King Abdullah University of Science and Technology	*Examinateur*
Renaud, MARLET Chercheur, Ecole des Ponts ParisTech	*Directeur de thèse*
Alexandre, BOULCH Chercheur, valeo.ai	*Co-Directeur de thèse*
Bryan, RUSSELL Senior Research Scientist, Adobe Inc.	*Invité*

SEMANTIC-GEOMETRIC JOINT APPROACH FOR THE RECONSTRUCTION OF DIGITAL MODELS FOR BUILDINGS

Pierre-Alain Langlois

December 14th, 2021

A dissertation submitted for the degree of
Doctor of Philosophy from Ecole des Ponts ParisTech

École doctorale n°405 MSTIC
Mathématiques & sciences et technologies de l'information et de la communication

Renaud Marlet	Senior Researcher, Ecole des Ponts ParisTech	Supervisor
Alexandre Boulch	Researcher, Valeo.ai	Co-Supervisor
Pierre Alliez	Directeur de Recherche, INRIA Sophia-Antipolis	Reviewer
Fréderic Jurie	Professeur, ENSICAEN & SAFRAN	Reviewer
Peter Wonka	Professor, King Abdullah University of Science and Technology	Examiner
Tania Landes	Maîtresse de conférences, INSA de Strasbourg	Examiner
Bryan Russell	Senior Research Scientist, Adobe Inc.	Invited jury member

Colophon This document was typeset in LaTeX, using the beautiful `tufte-latex` class[1] and a hand-made overlay inspired by Aaron Turon's thesis *Understanding and expressing scalable concurrency* and Marie-Morgane Paumard's thesis, *Solving jigsaw puzzles with deep learning*. Firmin Didot's `GFS Didot` acts as the typeface. The bibliography is typeset using `biblatex`.

First printing, Month Year

Acknowledgments

Je remercie chaleureusement mes directeurs de thèse, Renaud et Alexandre, pour les longues heures passées à discuter d'algorithmes, de papier, et de projets.

Résumé

L'émergence du Building Information Model (BIM) dans les domaines de l'industrie du BTP et de la gestion de la ville révolutionnent notre manière de concevoir, construire, gérer et entretenir nos bâtiments. Les modèles BIM rassemblent à la fois des informations géométriques sur le bâtiment, mais également des informations sémantiques permettant d'identifier chaque composant logique (mur, dalle, fenêtre, etc.). Cette information est généralement créée manuellement à la conception d'un nouveau bâtiment; cependant, seul 1% du parc immobilier à reconstituer est renouvelé chaque année. La génération automatique de maquettes BIM à partir de capteurs tels que des appareils photos ou des LIDARs (générant directement des nuages de points) constitue dans ce contexte un besoin de plus en plus important. L'objectif de cette thèse est le développement de nouvelles méthodes pour la reconstruction de maquettes BIM, qui incluent à la fois les informations géométriques et sémantiques. Un important effort de recherche est déployé pour rendre les méthodes existantes plus robustes aux contraintes présentes dans les différents cas d'usage pratiques. La reconstruction 3D débute généralement par l'acquisition de nuages de points par LIDARs ou par photogrammétrie, c'est-à-dire la triangulation de points clés mis en correspondance entre différentes photographies d'un même lieu, avant la reconstruction de la surface sous-jacente. Dans le contexte du bâtiment, la triangulation est rendue difficile par la présence de zones sans texture dans lesquelles les algorithmes peinent à détecter des points-clés. De plus, certaines parties du bâtiment peuvent être absentes des données d'entrée en raison des occultations ou d'erreurs imputables à l'opérateur. En ce qui concerne la sémantique au sein des nuages de points, il existe d'importantes ambiguïtés entre les différentes classes: il peut par exemple être ardu de repérer sur un nuage de points la séparation entre une porte et un mur. Dans cette thèse, nous proposons trois contributions techniques afin de traiter ces limitations.

Tout d'abord, nous proposons la première méthode permettant la reconstruction planaire par morceaux de bâtiments à partir d'images utilisant des segments de droites comme primitives. En effet, bien que les vastes zones sans texture dans les bâtiments (murs blancs par exemple) rendent difficile la détection de points clés, les segments de droites restent visibles et détectables (par exemple grâce au changement de luminance à l'intersection entre deux murs). L'utilisation de ces segments de droites rend notre méthode robuste aux surfaces sans texture, et nous sommes en mesure de reconstruire des scènes pour lesquelles les méthodes basées sur les points-clé échouent.

La seconde contribution traite le problème de la reconstruction de maillage à partir d'un ensemble d'images de basse résolution prises à des positions et orientations connues de l'espace. Dans ce contexte, les méthodes traditionnelles échouent, et apprendre des a priori directement sur de grandes bases de données de formes 3D nous permet tout de même d'effectuer une reconstruction. Plus précisément, notre méthode apprend des a priori afin de fournir une première approximation de la forme à reconstruire que nous affinons ensuite en ajoutant des contraintes géométriques. Notre méthode fournit directement un maillage et se compare positivement avec les méthodes de l'état de l'art dont la sortie est limitée à un nuage de point bruité.

Notre troisième contribution est VASAD, un jeu de données pour la reconstruction volumique et sémantique, que nous avons créé à partir de modèles BIM bruts disponibles en ligne. C'est le premier jeu de données de cette taille (62.000m²) à fournir simultanément des annotations géométriques et sémantiques. En plus de ce jeu de données, nous proposons 2 méthodes pour reconstruire de manière conjointe la géométrie et la sémantique à partir d'un nuage de point et nous démontrons que notre jeu de données propose des défis susceptibles de fortement influencer les recherches futures.

Mots-clé BIM, apprentissage, reconstruction, sémantique.

Abstract

The advent of Building Information Models (BIM) in the field of construction and city management revolutionizes the way we design, build, operate and maintain our buildings. BIM models not only include the geometric aspect of the buildings but also semantic information which identifies its logical components (walls, slabs, windows, doors, etc.). While this information is fairly reasonable to create during the building design, only 1% of the building stock to be restored is renewed each year. There is therefore an increasing need for automated methods to generate BIM models on existing buildings from sensors such as simple RGB cameras or more advanced LIDAR sensors which directly provide a point cloud. In this context, the goal of this thesis is to develop approaches for BIM reconstruction, including both geometric reconstruction and semantic analysis. These tasks have been explored, but an important research effort is being conducted to make them more robust to the variety of use cases found in practice. 3D reconstruction is usually operated based on direct 3D acquisitions such as LIDARs or using photogrammetry, i.e., using pictures to triangulate key point locations and reconstruct the surface afterward. In the context of buildings, the latter case is usually limited by the presence of textureless areas which makes it hard for the algorithms to find key points and to triangulate them. Moreover, some parts of the buildings might be missing from the input data because of occlusions or omission from the acquisition operator. Regarding semantics in point clouds, important ambiguities exist between semantic classes: the discontinuity between a wall and a door can be hard to distinguish from a point cloud only; a slab, a roof and a ceiling sometimes need additional context to be disentangled. In this thesis, we present three technical contributions to address these issues.

First, for 3D reconstruction of building scenes, we propose the first method to reconstruct piecewise-planar scenes from images using line segments as primitives. While wide textureless areas exist in indoor scenes (e.g., walls), making it particularly difficult to detect key points, lines are often more visible and easier to detect (e.g., change of illumination at the intersection of two walls) and therefore should be used to ensure robustness of image-based reconstruction approaches. We leverage the presence of these line segments and the possibility to detect and triangulate them. This makes the method robust to textureless surfaces, and we show that we can reconstruct scenes on which point-based methods fail.

The second contribution is more theoretical and addresses the problem of mesh reconstruction from multiple images of low resolution with known position and orientation in space. In this context, traditional methods completely fail and directly learning priors on a large scale dataset of 3D shapes allows us to still perform reconstruction. More specifically, our method uses the learned priors to provide an initial rough shape which is further refined by incorporating geometric constraints. Our method directly outputs a mesh and competes with state-of-the-art methods, which can only output a noisy point cloud.

Finally, the third technical contribution is VASAD, a dataset for volumetric and semantic reconstruction, which we created from raw BIM models available online. It is the first large scale dataset (62,000m^2) to offer both geometric and semantic annotations at point and mesh level. With this dataset, we propose two methods to jointly reconstruct both geometry and semantics from a point cloud and we show that the proposed dataset is challenging enough to stimulate research.

Keywords BIM, machine learning, reconstruction, semantic.

Publications

This dissertation draws heavily on earlier work and writing in the following papers:

Refereed conferences

- Pierre-Alain Langlois, Alexandre Boulch, and Renaud Marlet (2021). VASAD: Volumetric and Semantic Reconstruction, submitted.
- Alexandre Boulch, Pierre-Alain Langlois, Gilles Puy, and Renaud Marlet (2021). NeeDrop: Unsupervised Shape Representation from Sparse Point Clouds using Needle Dropping, submitted.
- [LFW+21] Pierre-Alain Langlois, Matthew Fisher, Oliver Wang, Vladimir Kim, Alexandre Boulch, Renaud Marlet, and Bryan Russell (2021). 3D Reconstruction by Parameterized Surface Mapping (ICIP);
- [LBM19] Pierre-Alain Langlois, Alexandre Boulch, and Renaud Marlet (2019). Surface Reconstruction from 3D Line Segments. (3DV);
- [XQL+19] Yang Xiao, Xuchong Qiu, Pierre-Alain Langlois, Mathieu Aubry and Renaud Marlet (2019). Deep Pose Estimation for Arbitrary 3D Objects. (BMVC);

Patent

- [LFR+20] Vladimir Kim, Pierre-Alain Langlois, Matthew Fisher, Bryan Russell and Oliver Wang (2020). Reconstructing three-dimensional scenes using multi-view cycle projection. (US Patent);

I presented my work in several research teams, namely the ONERA team DTIS, IGN team MATIS, LRI team Tau, Telecom ParisTech team 3D, Adobe Research and Amazon Imaging Sciences.

Contents

List of Artworks

List of Figures

List of Tables

List of Terms and Acronyms

GLOSSARY

BIM Building Information Modeling. vi, viii, x, 3–7, 9, 10, 15, 16, 85, 86, 90, 102, 110, 111, 114, 115

GPU Graphics Processing Unit. 9

IFC Industry Foundation Classes: a file format for BIM models. 7, 111

LIDAR Laser imaging, detection, and ranging. iii, iv, 3, 6, 7, 9, 15, 18, 82, 85–87, 90, 92, 94, 108, 111, 114, 115

MVS Multi-view stereo. 62

RANSAC Random Sample Consensus. xi, 14, 16, 17, 20–22, 29, 42, 43, 47, 52, 56

RGB Red Green Blue. 5, 9

RGB-D Red Green Blue Depth. 6, 87

Part I

PROLOGUE

1

Introduction

Chapter 2 ➤

1.1 MOTIVATION

1.1.1 *BIM*

Since the spread of computers in almost every professional field, construction and civil engineering have benefited a lot from this technology. Early in this interaction, the idea grew to save time and money by making digital models for buildings. Though it might seem obvious to use computers for building design, it was soon understood that bringing more information into the digital model could lead to gains in the whole building life-cycle steps [ULW19], such as operations, maintenance, repair or asset management.

The Building Information Models were invented to fill this purpose: not only do they contain the geometry of the building, but many operational details such as building parts, infrastructure, network, metadata and documentation (see Figure 1.1). Modern software has been built to make it convenient to generate such complex files, e.g., ReVit, BIM 360, ARCHICAD, or TEKLA Structures. While efforts are being made to standardize BIM files, the use of this digital twins gets more and more generalized. A visual representation of a building and its corresponding BIM file is provided on Figure 1.2.

1.1.2 *Automatic model generation*

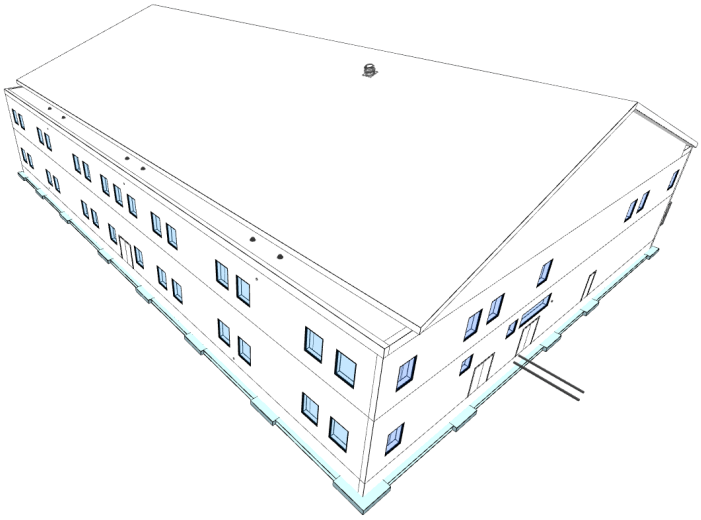
While this software allows the generation of rich data during the building design, it is expensive to manually create BIM files for existing building. As a matter of fact, this involves getting precise measurements, guessing what is behind hidden spots, gathering all the data in the software and of course, avoiding mistakes along this tedious work.

Again, computer science can have an impactful role to play in automating this task [HNZ21]. While parts of the infrastructure require specific sensors to recover (e.g., electric network in walls), recovering the logical functionality of each building component (walls, windows, stairs, etc.) as a semantic label is doable for a human by visualizing the data.

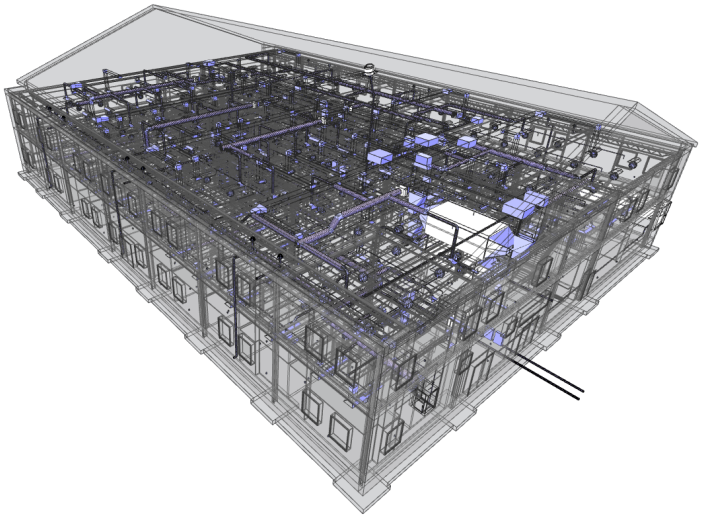
In this context we can ask: in the spirit of BIM models, is it possible with simple sensors such as regular cameras or LIDARs (see 1.2.1) to recover both the geometry and the semantic class of each component in a given building? This thesis is dedicated to answering that question. Our problem is twofold and we discuss here the definition of both aspects.

1.2 APPROACH

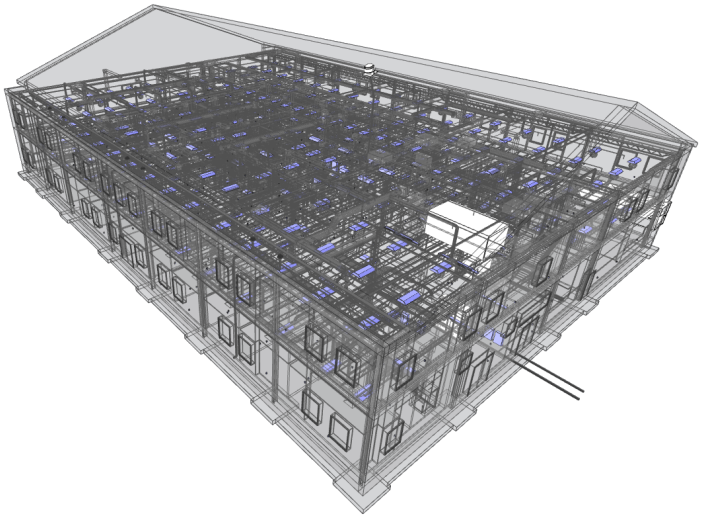
We provide here a general description of the tasks involved in the thesis and general bibliographic elements. A detailed related work is provided for each of our contributions



Structure and walls



Hydraulic network highlighted

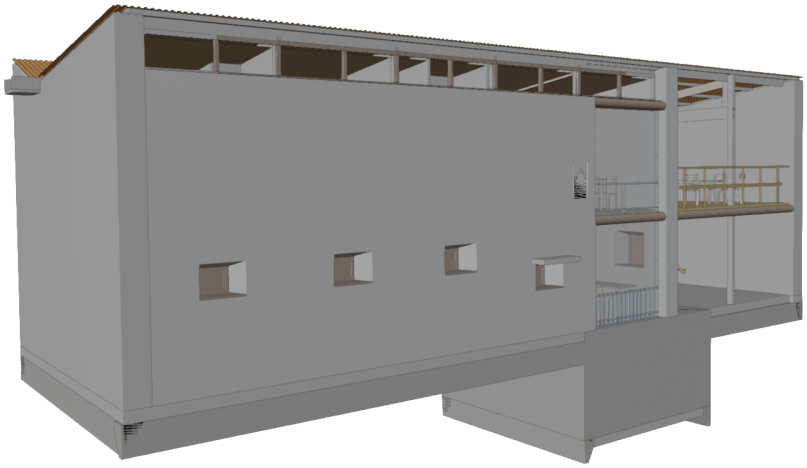


Electric network highlighted

Figure 1.1: A BIM model and its different components.

Actual building [Com12]

Figure 1.2: A building (Villa Sextant) and its digital twin.



Representation of the BIM model

in sections 2.2, 5.2 and 8.2.

1.2.1 *Sensors*

Every signal-processing routine starts with a sensor acquisition. For buildings, it is possible to use several kinds of sensors:

- **RGB cameras**: Regular cameras produce an image as a grid of pixels, each of which contains an RGB byte triplet which encodes a red/green/blue light color information. While this is a very natural way of capturing and representing our environment as we see it; this does not allow a direct access to a geometric information. With this type of

data, recovering the 3D information requires additional processing, which we develop below.

- **RGB-D cameras**: Also called 2.5D sensors, these devices (e.g., Microsoft Kinect, Intel Realsense and Apple's Face ID) make use of structured light to produce a depth information on top of the color information for every pixel. A point cloud can then directly be obtained thanks to the camera calibration parameters. Though they are still slightly more expensive than regular cameras, these sensors become more and more common as they are being integrated on modern smartphones.

- **LIDAR sensors**: Laser-based sensors produce a point cloud where each point represents the intersection between a laser beam and a physical surface. These devices are the most expensive but their price is decreasing fast. They are by far more precise than RGB-D camera and they cover a larger field of view.

Because they measure physical quantities, all of these sensors are subject to noise from various sources. These sources include the manufacturing precision of the optical components, the general condition of the optical components, the homogeneity of the atmosphere between the sensor and the captured object, and specific properties of the surface which may be incompatible with a proper capture (e.g., a mirror for RGB-D cameras and LIDAR sensors).

Dealing with noise is central when working on real data and specific methods perform denoising [MGMH04, TXFY18]. For this reason, algorithms can be designed first on synthetic data for which the level of noise is controlled, before being applied to real data.

1.2.2 *3D Reconstruction*

Reconstructing the geometry from photographs or LIDARs is usually referred to as *3D Reconstruction* in the computer vision literature. This process aims at retrieving the shape and appearance of real objects. The notion of virtual shape is very ill defined as there are many ways to represent a shape in 3D: point clouds, meshes, primitive set, implicit functions. The choice of the correct representation depends on the input, and the use we want to make of the 3D reconstruction.

In the particular case of BIM models, current software mostly represents the shapes as meshes, more specifically closed meshes. This allows us to define an interior and an exterior for any part in the scene. This distinction is paramount since most approaches in the computer vision literature are focused on surface reconstruction (i.e., generating surfaces which do not necessarily partition the space between interior/exterior) rather than volume reconstruction. In this dissertation, when dealing with building reconstruction, we strive to propose methods which reconstruct proper volumes as they are more useful for downstream applications. Additionally, the parts in BIM models are usually represented in an idealized version, i.e., the way the architect drew them. In order to get closer to this representation, it is usual to add constraints such as piecewise planarity and/or encourage right angles in the final reconstruction.

3D Reconstruction has the particularity to be an ill-defined problem: the representation we have of the object we try to reconstruct is often partial and there are therefore many ways to perform reconstruction while respecting the input data.

Images as input. When dealing with images, two major classes of methods are being used:

- **Photogrammetry**: given a set of input images, the idea is to first find corresponding point pairs across different images to calibrate the cameras (i.e., guessing the relative camera position to one another as well as the internal parameters such as the focal length and the optical center). It is then possible to obtain a point cloud by triangulation techniques. In practice, the bundle adjustment methods does both tasks (calibration and triangulation) simultaneously, but it is necessary to densify the matches.

- **Prior-based methods**: Assuming that we can decompose an image in a depth image, a reflectance image and an illumination model [Hor74], the *Shape from shading* algorithms propose to recover the shape of an object assuming just the illumination and reflectance are known [ZTCS99]. These methods make strong assumptions on the scene and work well only under specific conditions. More recently, the advent of deep learning techniques (see 1.2.4) allows to directly learn strong priors from large-scale datasets of objects.

Point cloud as input. When dealing with point clouds, we already have access to a geometric information which may be sufficient for some applications. However, point clouds have major flaws: to well represent a shape, they need to be particularly dense (typically hundreds of millions for a building), and the sampling needs to be as uniform as possible. This is a very heavy and inefficient representation, and due to the capture from fixed point of views, the raw point clouds are usually not uniformly sampled (see Figure 1.3). As previously mentioned, raw point clouds are also subject to noise, which makes the estimation of the underlying surface less precise. Finally, for many applications, the normal information of the surface at every point of is often required. Though this can be estimated by various methods [BM12], the accuracy of the method highly depends on the point cloud aforementioned qualities.

For these reasons, an important research effort has been conducted to perform surface prediction from a point cloud. Classical methods make use of priors such as the smoothness of the underlying surface [KBH06] or negligible noise [BMR+99]. In the case of BIM, surface reconstruction is not enough: we need a volume reconstruction, i.e., a partition of the space between empty and full parts. This is a harder task in general and less methods provide such partition as an output [BdLGM14, NW17].

1.2.3 *Semantic segmentation*

Our objective also includes semantic segmentation, i.e., the process of labeling each building part. In general, this task is still an important research challenge. Simple shapes like planes [SWK07] can efficiently be retrieved. When the segments are well separated by a high curvature, a region growing based algorithm [ZPK+02] can be used. However, for many practical tasks, and in the context of BIM reconstruction in particular, the parts we want to segment are not described by their geometry, but by their functional use. The IFC format often used for BIM projects defines parts such as slabs, roofs, windows, doors to name a few. These parts are usually represented by a closed volume and a label. Most of the time, the parts do not overlap and the separation between two objects is visually clear. However, when looking at the point cloud only (in the case of a LIDAR sensor), it can be very difficult to tell different components apart (see Figure 1.3). Moreover, parts in buildings can be ambiguous: in modern buildings flat roofs can for instance be mistaken for slabs. For these reasons, it makes sense to learn the shape priors directly on the data instead of hand crafting it. Again, recent deep-learning-based techniques show promising

results in this direction (see 1.2.4).

1.2.4 *Deep Learning*

In recent years, the deep-learning-based techniques have spread through all the tasks studied in computer vision and allowed major breakthrough. Even though these techniques have been first developed during the early years of computer vision [Ros58], two main factors account for the recent success of these algorithms:

- **Data**: The present-day steady release of large scale datasets [DDS+09, CFG+15] has allowed training deep learning models at large scale.
- **Computing power**: One of the first major breakthrough was accomplished thanks to a smart implementation of deep neural networks on a modern GPU for image classification [KSH12].

The influence of deep learning on our task is twofold:

- **Deep reconstruction methods**: Classical methods can only make use of the data from the specific scene they are trying to reconstruct, as well as hand-crafted priors. While this allows impressive results [SF16], the algorithm often fail where a human would perfectly understand the geometry of a scene thanks to his/her own experience of his/her surrounding world. Deep learning methods allow to go past this limitation by enabling scene completion [SYZ+17] or single-view reconstruction [GFK+18, PFS+19, MON+19]. Moreover, the encoder/decoder architecture of these methods allow handling various inputs, and as such, [GFK+18, PFS+19, MON+19] also allow reconstruction from a point cloud thanks to the powerful PointNet [QSMG17] encoder.
- **Deep segmentation methods**: As mentioned in 1.2.3, classical methods struggle to model complex classes. The deep-learning-based methods have first brought tremendous improvements on image segmentation [RFB15, LSD15]. These networks have been used to label point clouds by back-projecting the 2D segmentation information [BGLSA18], and specialized networks for directly segmenting 3D data have appeared [QSMG17, QYSG17, ÇAL+16, TQD+19].

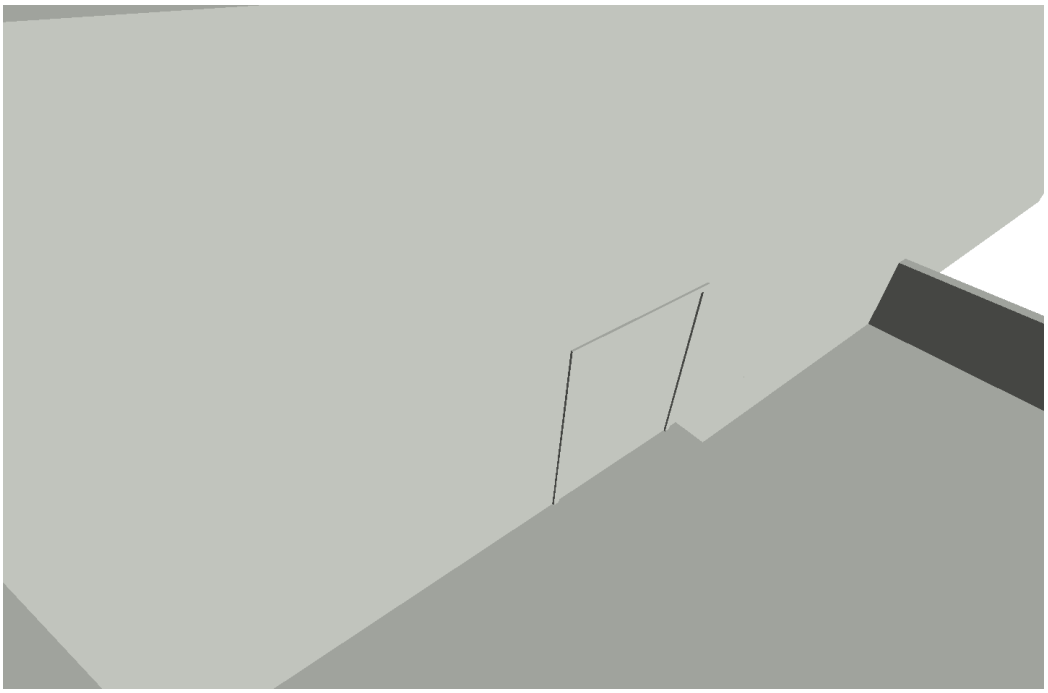
1.3 CONTRIBUTIONS

1.3.1 *Main contributions*

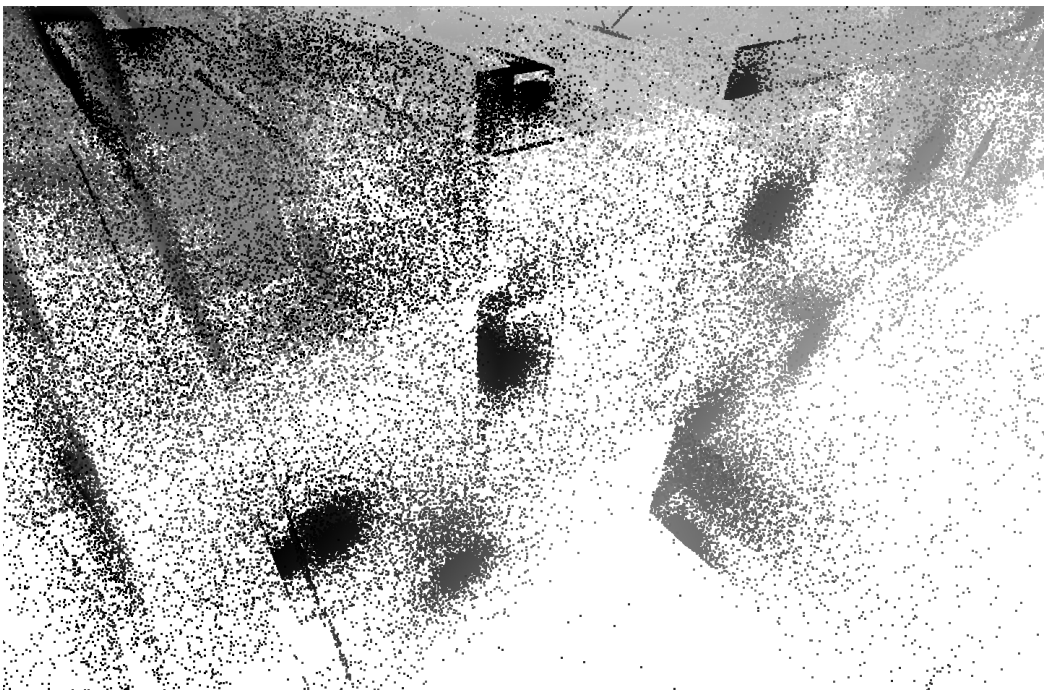
As we noticed in the previous section, many topics are concerned by the subject of BIM reconstruction. In this dissertation, we are interested in four particular aspects:

- 3D reconstruction of **buildings**.
- Reconstruction from **photo**graphs taken thanks to RGB cameras.
- Reconstruction from **point cloud**s obtained from LIDAR scans.
- The use of **machine learning**, especially deep learning to leverage large scale datasets to perform both reconstruction and semantic segmentation.

This dissertation aims at paving the way towards efficient methods to automatically reconstruct and label each logical part of a building. As such, we bring three contributions which deal with the four aforementioned aspects (see Figure 1.4):



Original mesh



Point cloud (darker points are closer to the sensor)

Figure 1.3: Even on synthetic data, it is difficult to distinguish a door on a point cloud.

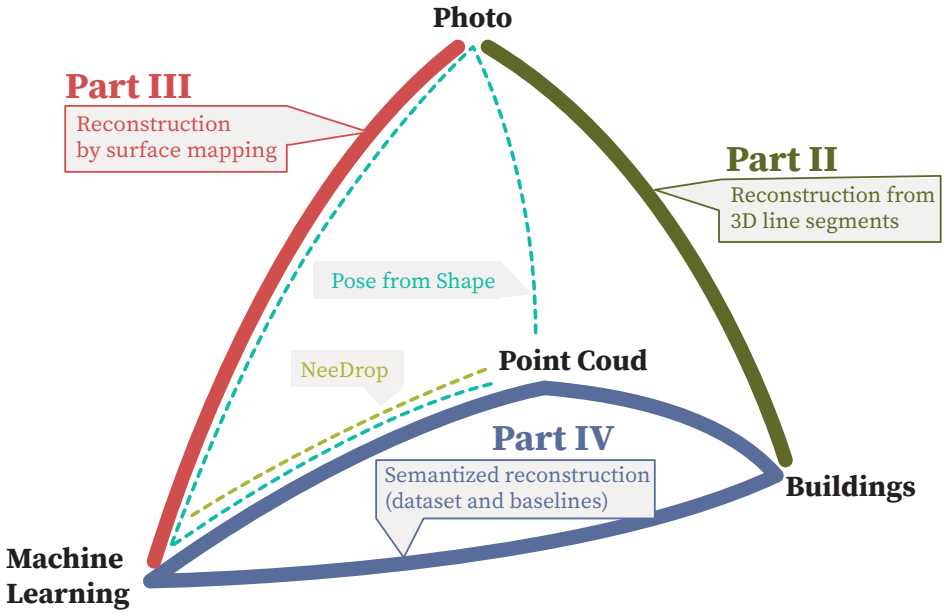


Figure 1.4: Summary of our contributions (framed) and their relationship to the topics cover by the thesis subject (black).

- **Surface Reconstruction from 3D Line Segments**

  In this contribution, we are interested in the pure problem of surface reconstruction in the context of buildings. In particular, we address one of the most important failure cases of the traditional methods, i.e., the reconstruction of textureless areas, which are omnipresent in modern buildings. To that end, we develop an original reconstruction method based on line segments instead of points.

- **3D Reconstruction by Parameterized Surface Mapping**

  In this contribution, we are interested in the possibility to use machine learning to learn priors for reconstruction. Moreover, we are interested in leveraging multiple views of an object to further refine the obtained mesh.

- **VASAD: a Volume and Semantic dataset for Building Reconstruction from Point Clouds**

  With this contribution, we bring into play the semantics along with geometry to make the synthesis of both aspects. We propose VASAD, a dataset built from freely available BIM products, and show that we can directly learn on models for the 3D reconstruction with semantic task, and therefore come closer to a full automation of the BIM generation process.

  We open source the code and data for our contributions on Github [Lan21].

1.3.2 *Other contributions*

Our other contributions in the course of the PhD will not be discussed in this dissertation. They include:

- Pose from shape [XQL+19]: a 3D pose estimation for objects from a single picture.

- NeeDrop: a work including an unsupervised loss to learn to reconstruct implicit surfaces from a dataset of point clouds.

1.4 ORGANIZATION OF THE DISSERTATION

Our surface reconstruction from 3D line segments algorithm is presented in part II while our 3D reconstruction by parameterized surface mapping algorithm is presented in part III. Our dataset VASAD and its baselines are presented in part IV and part V concludes this dissertation.

Part II

SURFACE RECONSTRUCTION FROM 3D LINE SEGMENTS

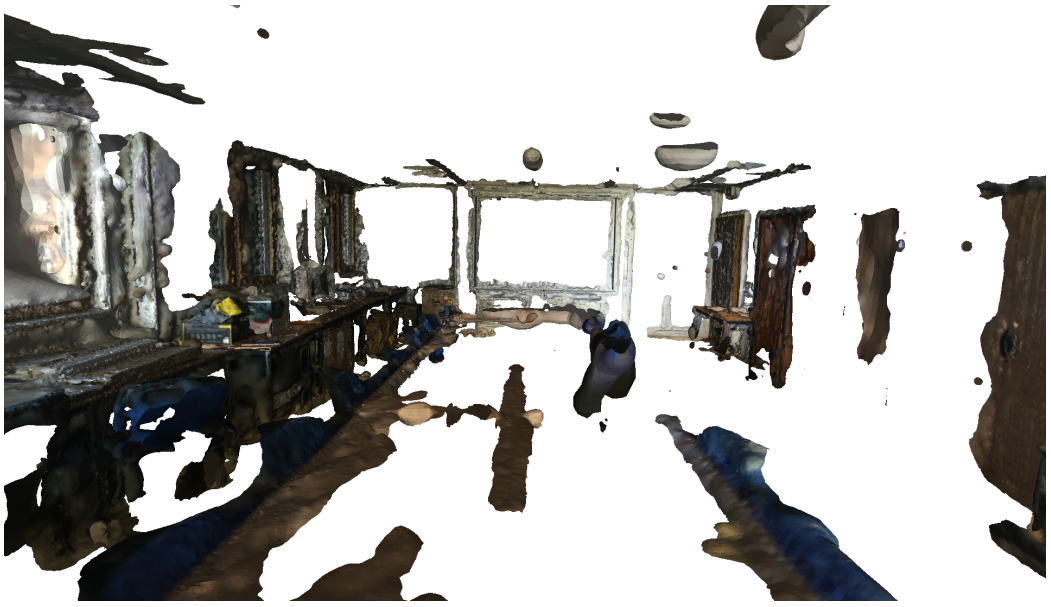
2

Introduction

◂ Chapter 1
Chapter 3 ▸

Synopsis In man-made environments such as indoor scenes, when point-based 3D reconstruction fails due to the lack of texture (see Figure 2.1), lines can still be detected and used to support surfaces. We present a novel method for watertight piecewise-planar surface reconstruction from 3D line segments with visibility information. First, planes are extracted by a novel RANSAC approach for line segments that allows multiple shape support. Then, each 3D cell of a plane arrangement is labeled full or empty based on line attachment to planes, visibility and regularization. Experiments show the robustness to sparse input data, noise and outliers.

3 image samples out of the 32 images used for reconstruction



Colmap [SF16] + Poisson [KBH06] failed reconstruction

Figure 2.1: Failed point-based reconstruction in a textureless environment

Numerous applications make use of 3D models of existing objects. In particular, models of existing buildings (e.g., BIMs) allow virtual visits and work planning, as well as simulations and optimizations, e.g., for thermal performance, acoustics or lighting. The building geometry is often reconstructed from 3D point clouds captured with LIDARs or using cameras and photogrammetry. But with cameras, registration and surface reconstruction often fail on indoor environments because of the lack of texture and strong view points changes: salient points are scarce, point matching is difficult and less reliable, and when calibration nonetheless succeeds, generated points are extremely sparse and reconstructed surfaces suffer from holes and inaccuracies.

Yet, recent results hint it is possible to rely on line segments rather than points. Lines are indeed prevalent in man-made environments, even if textureless. From robust detection [GvGJMR12, SMM16a] and matching [ZYL11, WWH09, HS12] to camera registration [EE11, SMM16b, SMM17, MDR18] and 3D segment reconstruction [HWB13, HMB16], lines can be used when photometric approaches fail for lack of texture. But as opposed to point processing, line-based surface reconstruction has little been studied [WM14, MG16]. This paper presents a novel approach to do so.

A change of paradigm is needed to consider 3D line segments rather than points. Transposing point-based methods to lines is difficult as many point-related assumptions do not hold for line segments. Indeed, points should be numerous enough (often, in thousands), with a uniform enough sampling, with an accurate enough detection and matching, and most of all, they must belong at most to one primitive. On the contrary, only a few tens of lines (rarely hundreds) are typically detected, and their density and sampling uniformity is so low that they cannot directly support a good surface reconstruction. Also, due to noise in local gradients and varying occlusions depending on viewpoints, segment detection is less accurate and often leads to over-segmentation and unstable end-points which are ignored by most 2D line matchers. Only after image registration and 3D segment reconstruction can 2D detections be related to actual fragments of a 3D line segment, moreover possibly differing according to the different viewpoints. Besides, curvy shapes as cylinders may yield unstable occlusion edges (silhouettes), yielding noise or outliers. Finally, some 3D lines identify straight edges that are creases between two planar surfaces, and thus support two shapes, contrary to points, that only support one shape.

Belonging to two primitives rather than one requires reconsidering shape detection. In particular, in greedy iterative methods, removing all data supporting a detected shape could prevent detecting other shapes because all or a significant fraction of features would then be missing. For instance, it would not be possible to detect all the faces of a cube given only its edges (see Figure 3.3). And even if enough 3D data remains for detection, shape sampling would be affected and some shapes would be less likely or unlikely to be detected.

Overview. We propose the first complete reconstruction pipeline that inputs 3D line segments with visibility information and outputs a watertight piecewise-planar surface without self-intersection (cf. Figure 2.2). We first extract primitive planes from the line cloud, distinguishing two kinds of line segments: *textural lines*, supporting a single plane, and *structural lines*, at the edge between two planes. Then, we label each 3D cell of the plane arrangement as full or empty by minimizing an energy based on line type, line segment support, visibility constrains and regularization.

Our main contributions are as follows:

- We define a robust and scalable plane detection method from 3D line segments,

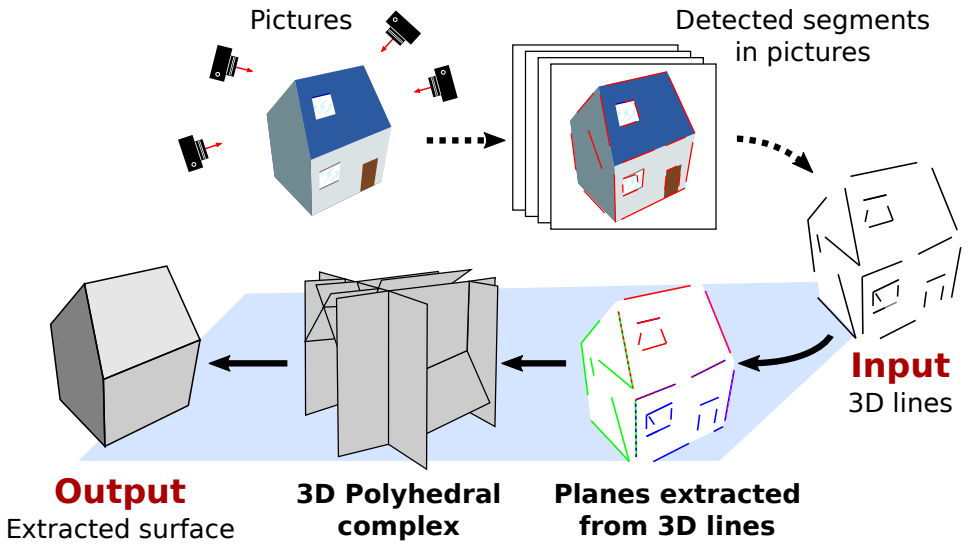


Figure 2.2: Line-based 3D reconstruction pipeline. This paper covers from **Input** to **Output**.

without scene assumptions. This novel non-straightforward RANSAC formulation takes into account a key specificity of lines vs points, namely that they can support up to two primitives (at edges), which breaks the greedy iterative detection traditionally used with points.

- We totally recast the surface reconstruction approach of [CLP10, BdLGM14] into a line-based setting. We meaningfully and efficiently generalize data fidelity and visibility from points to line segments, taking care of lines supporting two planes. We also feature a simpler and lighter treatment of noise.

- We validate our method on existing datasets, and provide new ones to assess line-based reconstruction quality.

2.2 RELATED WORK

Surface reconstruction has been extensively studied from 3D points [BTS+14] and/or images [FH15]. The conventional multi-view stereo reconstruction algorithms essentially focus on finding pointwise correspondences across images to first find the relative position of images to each other, and then to extract dense point cloud by incorporating epipolar constraints and designing photometric errors as detailed in [FH15]. Schoenberger et al. have developed COLMAP [SF16], a method which leverages both geometric and photometric constraints to build a point cloud from images. This algorithm outperforms the state-of-the-art on several public benchmarks. A surface can then be obtained from the dense point cloud thanks to meshing methods which leverage priors on the surface smoothness [KBH06] or the point cloud density [LPK09].

As opposed to the point-based methods, we consider here the input to be 3D line segments (with viewpoints), that can be sparse, missing, noisy and corrupted with outliers. We aim at an idealized piecewise-planar and watertight surface to be consistent with the current representation of buildings in BIM models.

To deal with sparse data, some methods detect planes based on 3D features and dominant orientations [SSS09], possibly with a Manhattan-world assumption [FCSS09], and create piecewise-planar depth maps taking into account visibility and local consistency. Other approaches consider 2D image patches back-projected on 3D planes [MK10, BdLGMK17]. In contrast, our method produces a watertight mesh, does not impose a few specific orientations, and can work with 3D features only, not requiring

images and not working at pixel level.

Another approach to little input data is to extend boundaries and primitives until they intersect [CLF02]. It however does not ensure a watertight reconstruction either. This is only achieved by methods that create a volumetric partition of the 3D space and extract the surface from full and empty cells. The partition can be a voxel grid [SDK09], a 3D Delaunay triangulation [LPK07, WWW11] or a plane arrangement [CLP10, BdLGM14].

Wireframe reconstruction is what most lined-based methods focus on: rather than surfaces, they study how to generate meaningful 3D line segments [JKTS10, HWB13, HMB16, IS17], after line matching is performed [SZ97]. And more general curves than lines are not used beyond structure from motion [NF15].

Surface reconstruction with lines in addition to points has received a modest attention. [BZ99] reconstruct planes from a 3D line and a neighboring detected point. It requires lines surrounded with texture and is outlier-sensitive. It also does not prevent self-intersections nor guarantees watertightness. [BENV06] segments images into likely planar polygons based on 3D corner junctions and use best supporting lines to reconstruct polygons in 3D. For 2.5D reconstruction, extracted 3D lines [SZ97] are used with a dense height map to build a line arrangement on the ground plane and create geometric primitives and building masks [ZBKB08]. In [SSS09], pairs of 3D lines generated from vanishing directions provide plane hypotheses, validated by 3D points. The surface is a set of planar patches created from plane assignment to pixels. [STO15] adds points uniformly sampled on the 3D lines to the Delaunay triangulation, introducing extra parameters, and although visibility is treated without sampling, the method is unlikely to work on scenes with only sparse lines. [HMB14] also shows a meshing improvement using 3D line segments.

Surface reconstruction from line segments only, when points fail due to the lack of texture, has little been studied.

[ZRK12] presents a single-view surface reconstruction based on 2D line segments. Lines are paired from segment extensions along their direction, and planes orientations are sought by RANSAC, hypothesizing mutually orthogonal corresponding 3D lines. Articulating lines are found at plane intersections to construct a multi-plane structure. Our *structural lines* are called 'articulation' or 'articulating' lines in [ZRK12]. They are discovered late, to set plane offsets, whereas we differentiate them early at plane detection. For robotic mapping, [WM14] considers all combinations of two non-collinear coplanar line segments as plane hypotheses. Line segments are then assigned to possibly multiple planes in a face complex built from plane intersections. The reconstructed surface is made of faces depending on an occlusion score. Compared to our approach, this method does not scale well to many lines, is sensitive to outliers, relies on a number of conservative heuristics that can be detrimental to surface recall, involves no regularization, and does not reconstruct a watertight mesh. As for [MG16], it first reprojects 3D lines into images that see them, studies the intersection of segments in 2D rather than planes in 3D, and infers plane hypotheses. The surface is made from image faces back-projected onto a possible 3D plane. Although less sensitive to outliers, this method involves heuristics and no proper regularization, and it reconstructs a non-watertight mesh with floating polygons and possible self-intersections.

Extracting 3D planes from line segments has little been treated; the literature focuses on point clouds, chiefly ignoring line clouds. The most popular scheme for points, which is robust to sparsity contrary to region growing as in [CLP10, BdLGM14], is RANSAC [CKY09, SWK07]. But as explained below, it cannot straightforwardly be applied to line segments because it relies on different distribution hypotheses and because of the possible association of a single segment to several primitives, also invalidating line discretization

into points. Still, [WM14] takes line segments as input, but plane detection is somewhat exhaustive, hence with scalability issues, and sensitive to outliers. Using laser data, [CCO15] exploits 3D lines to detect planes, but it uses strong properties of LIDAR acquisition, namely line parallelism and large and dense data.

An open question is if multi-model methods [ZKM05, TF08, IB12, MF14], which assume non-overlapping segmented data, can be adapted not only to large inputs but also to multiple shape support [KKZ09, BTL16], as absolutely required for line segments.

Surface reconstruction from a plane arrangement is a common topic, with variants enforcing plane regularity [LWC+11, MMBM15] or level of detail [VLA15], or offering reconstruction simplicity [NW17]. It is largely orthogonal to our work. Here we build on [BdLGM14], with line-specific data and visibility terms.

3

Method

◂ Chapter 2
Chapter 4 ▸

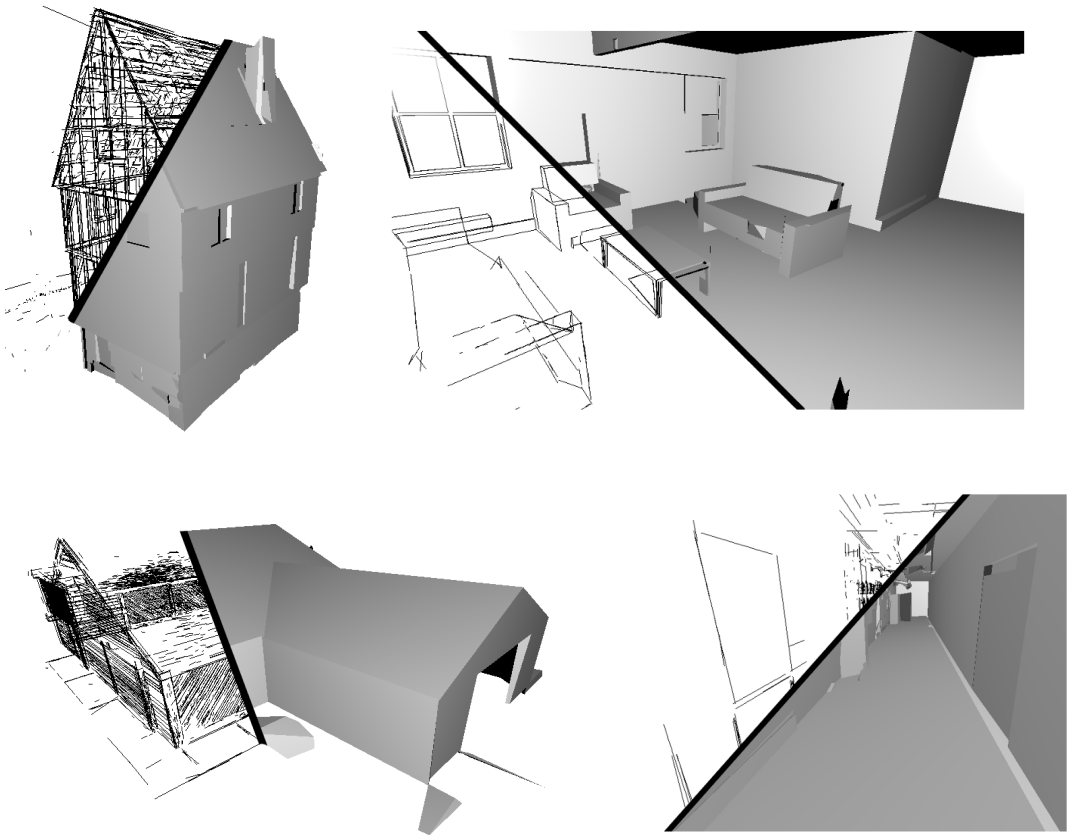


Figure 3.1: Input lines and reconstructed surfaces on 4 datasets (from left to right): TimberFrame, HouseInterior, Barn, Terrains.

3.1 LINE EXTRACTION

In this work, we assume we have as input 3D line segments as well as their viewpoints. Concretely, for our experiments, we extract 3D line segments thanks to the Line3d++ algorithm [Hof16, HMB16]. First, when the calibration is not provided, we compute it thanks to the classical feature matching and filtering method with bundle adjustment available in COLMAP [SF16]. Then, lines are detected thanks to the a contrario method LSD [GvGJMR12]. Line3d++ essentially consists of 4 steps, plus 1 optional step. It first generates potential line matches across the images based on epipolar constraints. The matches are then filtered based on the heuristic that correct matches usually support each other, i.e., a good potential 3D segment should be close to other 3D segments. After this process, each remaining 2D detection which has at least one match generates a 3D

hypothesis thanks to the match that has the highest confidence, as computed during the previous step. Finally, the 2D detections are clustered based on the spatial proximity of their 3D hypothesis, and a 3D line segment is determined for each output cluster. As an optional refinement step, a bundle adjustment can be applied to ensure that the final 3D line segments fit their 2D residuals.

The output 3D line segments along with their 2D residuals and the camera poses constitute the input to our method.

3.2 Plane detection from 3D line segments

The first step of our approach is to detect planes that are supported by line segments in the input line cloud $\mathcal{L}$. We use the RANSAC framework [FB81] as it scales well to large scenes and deals well with a high proportion of outliers, which are unavoidable in photogrammetric data. The general idea is to randomly select two candidate lines in the input line set and to find the lines which are close to the plane generated by the candidate lines. Repeating this process and keeping the plane with the largest number of inlier lines statistically allows to recover the most salient plane of the line set (see Figure 3.2). Our whole algorithm is summarized in Figure 3.5 and we provide here a detailed explanation.

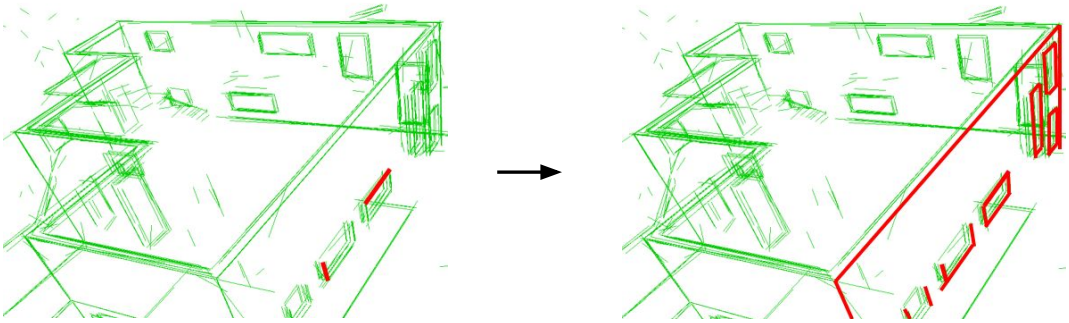


Figure 3.2: One iteration of our RANSAC process: two random line segments are selected (left) and the lines fitting the corresponding candidate plane are selected (right).

As argued above and shown experimentally (cf. Sect. 4), a key requirement is to allow a line to belong to two planes. Lines supporting one plane are considered *textural*; lines supporting two planes are deemed *structural*. Yet some actual texture lines may support additional "virtual" planes, as when a line is drawn around an object, e.g., at the borders of a frieze around the walls of a room, which belongs both to the vertical walls and to an non-physical horizontal plane.

3.2.1 *Candidate plane construction*

We generate candidates by sampling the minimum number of observations required to create a model, i.e., two non-collinear line segments to define a plane. Two 3D segments l_a, l_b can be coplanar in two ways: they can be parallel, or their supporting infinite lines can intersect. With noisy real data, the latter can be relaxed using a maximum small distance ϵ between the lines. We discard parallelism because, when reconstructing man-made environments such as buildings, it may generate many bad planes. Indeed, two random vertical segments (e.g., detected on windows) are parallel but statistically unlikely to support an actual, physical plane (e.g., segments on different facades). We thus threshold the angle $\angle(l_a, l_b)$, which also excludes the degenerate case of collinear segments.

3.2.2 *Greedy detection and multi-support issues*

We sample planes as line pairs and perform an iterative extraction of the most significant planes, i.e., with the largest number of supporting segments after a given number of sampling trials. However, contrary to usual RANSAC, we cannot remove supporting segments at once as they may actually belong to two planes; it would lead to detecting the main planes only, missing planes with a smaller support. Conversely, we cannot consider all segments as available at each iteration: it would statistically lead to multiple detections of the same large planes and again miss planes with small support.

A natural way to allow a datum to be part of several detection in greedy RANSAC is to remove inliers for model sampling but not for data assignment to models [ZRK12]. But for sparse data (which is the case with line segments), it fails to detect models with little data support, e.g., preventing detecting all the faces of a cube from its sole edges (see Figure 3.3).

Another way to allow the same datum to seed several models is to bound their number, i.e., 2 for lines supporting planes. But it does not work either as it often associates a line twice to more or less the same plane.

Our solution, as described bellow, is to bound the number of supported planes per line segment, but with an additional condition to prevent shared segments to belong to similar planes.

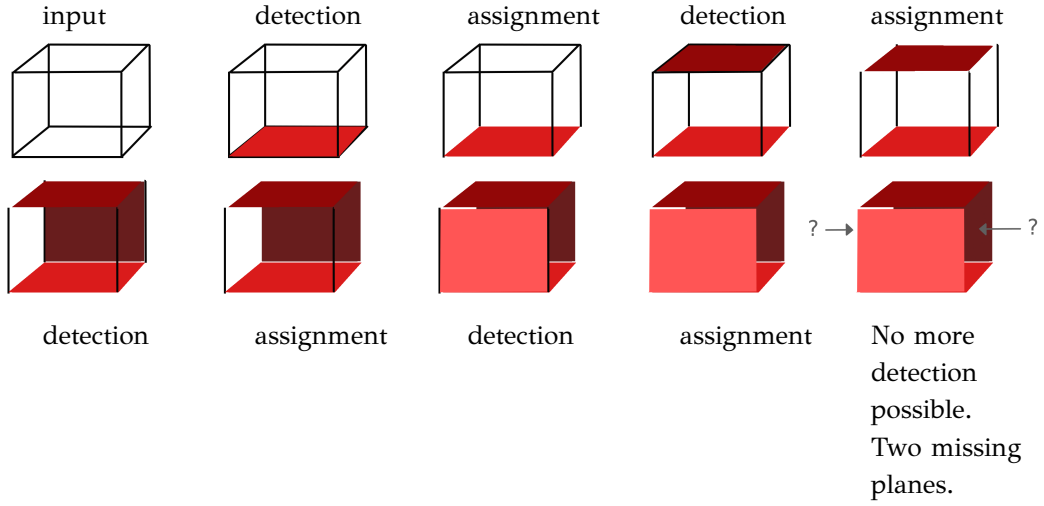


Figure 3.3: Because line segments are sparse, removing the inliers immediately after a plane detection leads to missing planes in the simple case of a cube whose detected lines are the edges.

3.2.3 *Candidate plane generation*

We note $\Lambda(P)$ the set of line segments supporting a plane P, $\Pi(l)$ the set of planes supported by a line segment $l \in \mathcal{L}$, with $|\,\Pi(l)\,| \leq 2$, and $\mathcal{L}_i$ the set of segments supporting i plane(s) for i in $0, 1, 2$.

We construct these sets iteratively by generating candidates planes P and assigning them segments $l \in \mathcal{L}$, some of which may have already been assigned to another plane $\Pi(l)$. Only line segments in $\mathcal{L}_2$ are discarded from the pool of available segments to support a plane, as they already support two planes. Initially, $\mathcal{L}_0 = \mathcal{L}$, and $\mathcal{L}_1 = \mathcal{L}_2 = \emptyset$.

As line segments are not put aside as soon as they are assigned to a plane, they can be drawn again to generate new candidate models. However, generating several times the same plane (with the same supporting line segments) would not only reduce efficiency, but

also make some models little likely to be drawn, as models with a large support would be sampled much more often. To prevent it, after drawing a first line segment $l_a \in \mathcal{L}_0 \cup \mathcal{L}_1$, there are two cases. If $l_a \in \mathcal{L}_0$, i.e., if l_a has not been assigned to any plane yet, then the second segment l_b can be drawn unconditionally in $\mathcal{L}_0 \cup \mathcal{L}_1$ as it will always yield a new model. If $l_a \in \mathcal{L}_1$, i.e., if l_a has already been assigned to some plane P', with $\Pi(l_a) = \{P'\}$, then lines in $\Lambda(P')$, i.e., supporting P', are excluded when drawing the second segment l_b. This ensures l_a, l_b cannot participate to the same already existing model. As the number of extracted planes is typically less than a few hundred, this drawing can be optimized by incrementally keeping track of the sets $\bar{\Lambda}(P) = \mathcal{L} \setminus (\mathcal{L}_2 \cup \Lambda(P))$, that have *not* already been assigned to a detected plane P.

Note that we do not prevent a line pair to be redrawn when it previously failed to generate an accepted model (for lack of planarity, parallelism or poor support). It is not an issue as it does not lead to unbalanced chances to detect a plane. Yet, when the number of input line segments is not too large, we can perform a systematic drawing of all line pairs, possibly exploiting the above filtering. In this case, all possible models are considered and at most once.

3.2.4 *Inlier selection*

After picking a candidate plane P, we populate the support $\Lambda(P)$. For this, we go through each segment $l \in \mathcal{L}_0 \cup \mathcal{L}_1$ and assign it to $\Lambda(P)$ if close enough to P, i.e., if $d(l, P) \leq \epsilon$. Several distances can be used, such as the average or the maximum distance to the plane.

If l already supports some other plane P', i.e., if $\Pi(l) = \{P'\}$, then also assigning l to P would make it a structural segment. As such, we impose that it lies close to the line at the intersection of both planes, i.e., $d(l, P \cap P') \leq \epsilon$ (see Figure 3.4). This condition is stronger than imposing both $d(l, P) \leq \epsilon$ and $d(l, P') \leq \epsilon$ as the angle between P and P' could be small and l could then be close to both P and P' although far from their intersection. This condition is actually crucial. Without it, we would tend to associate l to two planes P and P' which are very similar, and fail to detect crease lines.

3.2.5 *Plane selection*

Last, we sample N_{iter} models and keep the plane with the largest number of inliers. (See Figure 3.5 for the abstract version of the algorithm.)

This plane detection differs from [ZRK12], that samples and populates planes from 2D line pairs instead of 3D lines, making inlier search quadratic, not linear, and requiring heuristically to only consider pairs defined by intersecting segment extensions, which is highly unstable due to noise in endpoints and which induces plane splitting at occlusions. We have none of these downsides. Besides, structural lines in [ZRK12] are found with heuristics after RANSAC, considering plane pairs and candidate lines, which only makes sense as they have few (<10) planes. We get them directly, without heuristics, in greater number, and for many more planes.

3.2.6 *Plane refitting*

After each plane P_{best} is selected, it is actually refitted to its inliers Λ_{best} before being stored into Π, based on the (signed) distance of the segment endpoints, weighted by the segment length. As it changes the plane equation, we check whether the slice centered on the

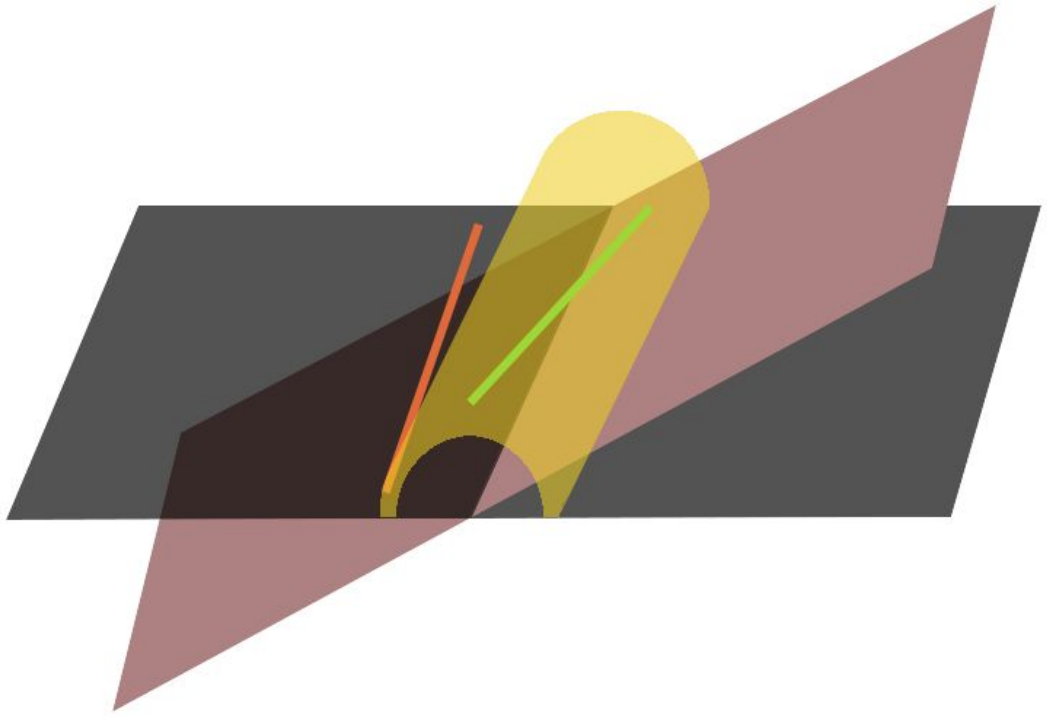


Figure 3.4: Inlier selection. We assume the black plane has already been detected and we are selecting inliers for the pink plane. Both the red and green line segments are close to the planes but only the green segment is selected because it is close enough to the plane intersection (i.e., within the yellow cylinder).

refitted plane P' with thickness ϵ now contains extra segments. If so, they are added as inliers and refitting is repeated.

3.2.7 *Plane fusion*

Modeling a building may require different levels of details, including small plane differences such as wall offsets for door jambs, baseboards or switches. But setting a small ϵ to do so may easily break a wall or a ceiling into several fragments because it is not perfectly planar due to construction inaccuracies or load deflections. Each country actually has standards (official or not) defining construction tolerances, e.g., 1 cm error every 2 m for walls.

To prevent this arbitrary fragmentation while preserving details, we add a post-processing plane fusion step with a tolerance higher than ϵ, i.e., with a maximal distance threshold $\epsilon_{\text{fus}} > \epsilon$ to the plane refitted on the union of inliers. This allows merging at ϵ_{fus} accuracy several plane fragments detected at ϵ. However, to make sure it applies only to cases described above, we impose a maximum angle θ_{fus} when merging two planes and minimum proportion p_{fus} of common inliers. Concretely, we consider all pairs of planes in Π whose angle is less than θ_{fus}, sort them, pick the pair with the smallest angle, and try merging it. If it succeeds, the two planes are removed, the new refitted plane is added, and the priority queue based on angles is updated before iterating. If it fails, the pair of planes is discarded and the next pair is considered. This is similar to a heuristics used in Polyfit [NW17].

3.2.8 *Plane limitation*

To make sure not too many planes are given to the surface reconstruction step, because of possible limitations (cf. Sect. 4.7), the algorithm may be stopped after at most N_{max} (best)

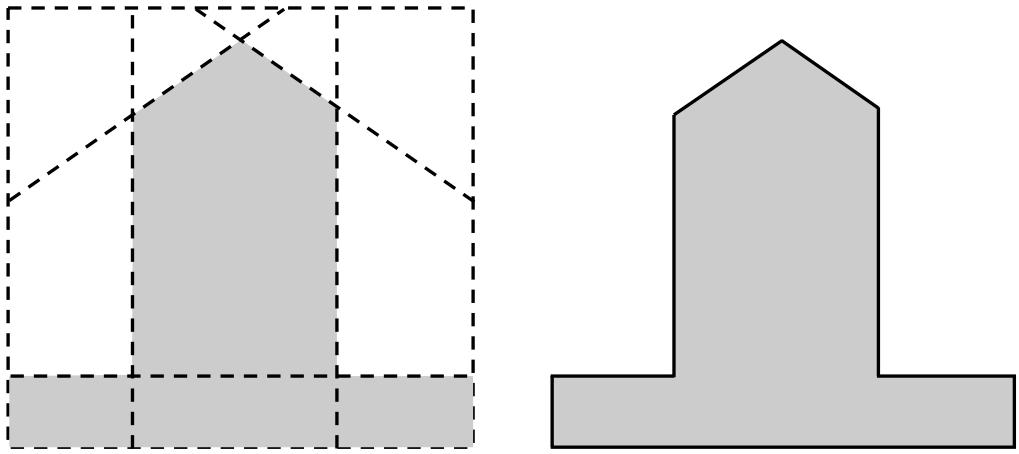
INPUT: set $\mathcal{L}$ of 3D line segments
$\mathcal{L}_0 \leftarrow \mathcal{L}, \quad \mathcal{L}_1 \leftarrow \emptyset, \quad \mathcal{L}_2 \leftarrow \emptyset, \quad \Pi \leftarrow \emptyset$
while $|\,\mathcal{L}_0 \cup \mathcal{L}_1\,| \geq 2$ **and** $|\,\Pi\,| < N_{\text{max}}$ **do**
  $\Lambda_{\text{best}} \leftarrow \emptyset$ *// Current best set of coplanar lines*
  repeat N_{iter} **times**
    //// Sample a candidate plane by sampling 2 lines
    Pick $l_a \in \mathcal{L}_0 \cup \mathcal{L}_1$ *// 1st sample line*
    //// For 2nd sample, exclude lines of the plane of l_a if any
    Pick $l_b \in \mathcal{L}_0 \cup \mathcal{L}_1\, \Lambda(\Pi(l_a))$ *// 2nd sample line*
    //// Make candidate plane and check consistency
    $P \leftarrow \text{plane}(l_a, l_b)$
    next if P degenerate **or** $d(l_a, P) > \epsilon$ **or** $d(l_b, P) > \epsilon$
    //// Gather line support for plane P
    $\Lambda \leftarrow \{l \in \mathcal{L}_0 \,|\, d(l, P) \leq \epsilon\} \cup \{l \in \mathcal{L}_1 \,|\, d(l, P \cap \Pi(l)) \leq \epsilon\}$
    //// Remember best candidate
    if $|\,\Lambda\,| > |\,\Lambda_{\text{best}}\,|$ **then** $\Lambda_{\text{best}} \leftarrow \Lambda$, $P_{\text{best}} \leftarrow P$ **end if**
  end repeat
  //// Update data structures
  $\Pi \leftarrow \Pi \cup \{P_{\text{best}}\}$ *// Set of detected planes*
  $\Lambda(P_{\text{best}}) \leftarrow \Lambda_{\text{best}}$ *// Support of best plane*
  $\forall\, l \in \Lambda_{\text{best}}, \Pi(l) \leftarrow \Pi(l) \cup \{P_{\text{best}}\}$ *// Planes supported by l*
  $\mathcal{L}_2 \leftarrow \mathcal{L}_2 \cup (\Lambda_{\text{best}} \cap \mathcal{L}_1)$
  $\mathcal{L}_1 \leftarrow (\mathcal{L}_1\ (\Lambda_{\text{best}} \cap \mathcal{L}_1)) \cup (\Lambda_{\text{best}} \cap \mathcal{L}_0)$
  $\mathcal{L}_0 \leftarrow \mathcal{L}_0\ (\Lambda_{\text{best}} \cap \mathcal{L}_0)$
end while
OUTPUT: set of planes Π, with related support $\Lambda(P)_{P \in \Pi}$

Figure 3.5: RANSAC-based plane detection from 3D lines segments, differentiating structural lines from textural lines.

greedy detections.

3.3 SURFACE RECONSTRUCTION

The second step of our approach is surface reconstruction based on detected planes and observations of 3D line segments. Rather than selecting plane-based faces with hard constraints for the the surface to be manifold and watertight [NW17], we follow [CLP10, BdLGM14] and consider a scene bounding box, partition it into 3D cells constructed from the planes, and assign each cell with a status 'full' or 'empty' depending on segment visibility, with a regularization prior coping with sparse and missing data. The reconstructed surface is then the interface between full and empty cells (see Figure 3.6). By construction, it is watertight and free from self-intersections.



2D Plane arrangement with labels (white: 0, i.e., empty; grey: 1, i.e., full)

Extracted surface

Figure 3.6: Extracting the reconstructed surface from the plane arrangement

Our contribution is a total reformulation of [CLP10, BdLGM14] in terms of lines, making the difference between textural and structural lines, and with a lighter treatment of noise in data.

The volume partition is given by a cell complex $\mathcal{C}$ made from an arrangement of planes detected in the line cloud. For each cell $c \in \mathcal{C}$, we represent occupancy by a discrete variable $x_c \in \{0,1\}$: 0 for empty and 1 for full. A surface is uniquely defined by a cell assignment $\mathbf{x} : \mathcal{C} \mapsto \{0,1\}$, where $\mathbf{x}(c) = x_c$. The optimal cell assignment $\mathbf{x}$ is defined as the minimum of an energy $E(\mathbf{x})$ which is the sum of three terms: a primitive term $E_{\text{prim}}(\mathbf{x})$ penalizing line segments not lying on the reconstructed surface, a visibility term $E_{\text{vis}}(\mathbf{x})$ penalizing surface reconstructions on the path between observations and their viewpoints, and a regularization term $E_{\text{regul}}(\mathbf{x})$ penalizing complex surfaces.

$$E(\mathbf{x}) = E_{\text{prim}}(\mathbf{x}) + E_{\text{vis}}(\mathbf{x}) + E_{\text{regul}}(\mathbf{x}) \tag{3.1}$$

3.3.1 *Dealing with noise*

To deal with noise in input data, [BdLGM14] introduces slack in the choice of cells penalized for not being at the reconstructed surface and lets regularization make the right choices. The resulting formulation and resolution is heavy. Instead, we assume that plane

extraction (Sect. 3.2) did a good-enough job: any segment supporting a plane (resp. two planes) is considered as a noisy inlier and is projected on the plane (resp. the intersection of the two planes). A segment not supporting any plane is treated as an outlier for data fidelity (no penalty for not being on the reconstructed surface) but not for visibility (penalty for not being seen from viewpoints if hidden by reconstructed surface).

3.3.2 *Primitive term*

$E_{\text{prim}}(\mathbf{x})$ penalizes line segments that support planes but do not lie on the reconstructed surface. But it does not penalize the presence of matter in front of segments w.r.t. viewpoints, letting the visibility term do it. Segments that support no plane are ignored as if outliers.

For a segment l supporting one plane P, and for each viewpoint v seeing at least a part of l, we consider the set C of all cells c immediately behind l w.r.t. v, possibly only along a fraction l_c of l due to occlusions (cf. Fig. 3.7(a)). Each $c \in C$ is penalized if not full, with a cost $1-x_c$.

For a segment l supporting two planes P_1, P_2, a cell behind l w.r.t. viewpoint v need not be full. Any configuration is valid as long as the space around l is not empty (cf. Fig. 3.7(b)): salient edges, reentering edges or planes (if the seemingly structural line happens to only be textural). To penalize only when all three cells c around a visible fraction of l are empty (ignoring the cell in front), we consider a cost of $\max(0, 1-\sum_c x_c)$, which is equal to 1 in this case, and 0 in other configurations.

Both textural and structural cases can be covered with a single formula, where we weigh the cost by the length of the visible fraction of l and normalize it by a scale of interest σ:

$$E_{\text{prim}}(\mathbf{x}) = \sum_{l \in \mathcal{L}_1 \cup \mathcal{L}_2} \sum_{v \in \mathcal{V}(l)} \sum_{C \in \mathcal{C}(l,v)} \frac{|\, l_C \,|}{\sigma} \max(0, 1 - \sum_{c \in C} x_c) \tag{3.2}$$

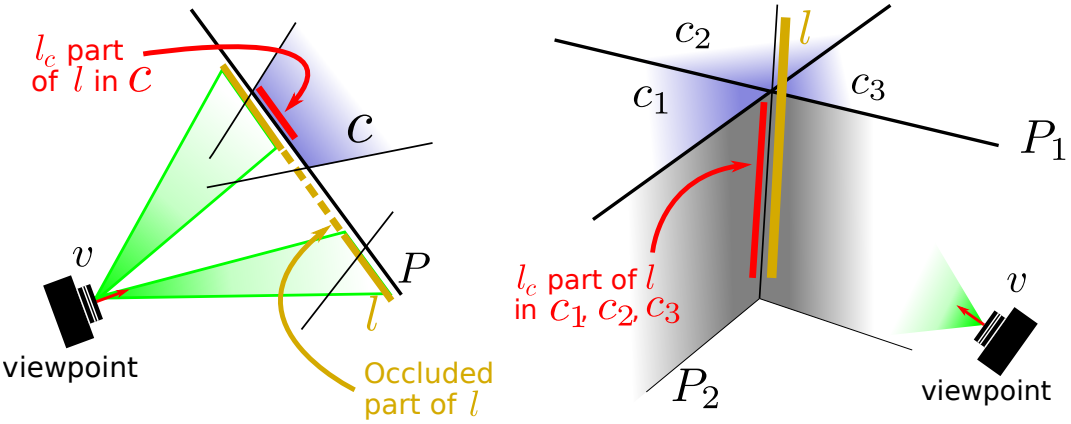
where $\mathcal{L}_1 \cup \mathcal{L}_2$ is the set of segments l supporting at least one plane, $\mathcal{V}(l)$ is the set of viewpoints v seeing l, $\mathcal{C}(l,v)$ is the set of cells c adjacent to l but not in the triangles of sight from v to non occluded fragments of l (locally 1 or 3 cells as to whether l belongs to 1 plane or 2 planes), l_C is the set of fragments of l in each cell $c \in C$, and $|\, l_C \,|$ is the sum of the lengths of segment fragments in l_C.

3.3.3 *Visibility term*

$E_{\text{vis}}(\mathbf{x})$ penalizes reconstructed surface boundaries between viewpoints and segments, as [CLP10, BdLGM14]. It measures the number of times a 3D segment l is to be considered an outlier as it should not be visible from a given viewpoint v, weighted by the length of the visible parts $l_{v,f}$ of l on the offending faces f (possibly fragmented due to occlusions). Contrary to $E_{\text{prim}}(\mathbf{x})$, all segments are considered in $E_{\text{vis}}(\mathbf{x})$, not just segments supporting a plane:

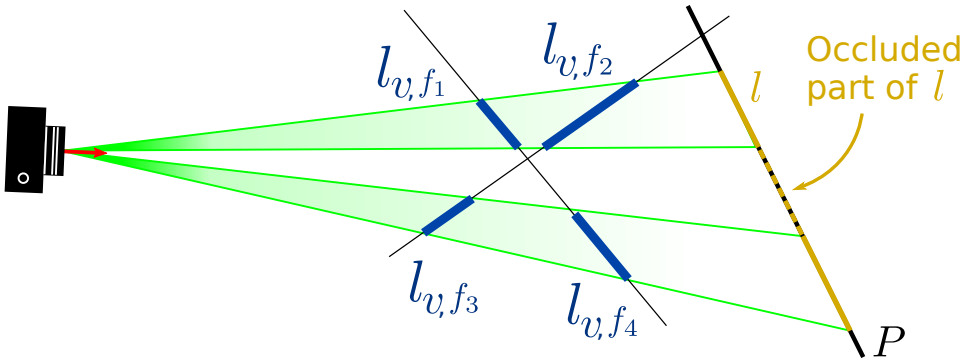
$$E_{\text{vis}}(\mathbf{x}) = \lambda_{\text{vis}} \sum_{l \in \mathcal{L}} \sum_{v \in \mathcal{V}(l)} \sum_{f \in \mathcal{F}(l,v)} \frac{|\, l_{v,f} \,|}{\sigma} \, | \, x_{c_f^{+v}} - x_{c_f^{-v}} \, | \tag{3.3}$$

where $\mathcal{F}(l,v)$ is the set of faces f of the complex intersected by the visibility triangle (l,v), at some unoccluded segment fractions $l_{v,f}$ totaling a length of $|\, l_{v,f} \,|$, and c_f^{+v}, c_f^{-v} are the cells on each side of f (c_f^{+v} being nearest to v).



(a) Primitive term, $l \in P$

(b) Primitive term, $l \in P_1 \cap P_2$



(c) Visibility term

Figure 3.7: Primitive and visibility terms.

3.3.4 *Regularization term*

$E_{\text{regul}}(\mathbf{x})$ penalizes surface complexity as the sum of the length of reconstructed edges (see Figure 3.8) and the number of corners (see Figure 3.9), with relative weights $\lambda_{\text{edge}}, \lambda_{\text{corner}}$, as defined in [BdLGM14]. Area penalization makes little sense here due to the low density of observations in some regions. We look for the most simple surface, which we assume is the surface that corresponds to a balanced minimization of these 2 measures (short edge length, few corners) while being consistent with the observation (line segment attachment to plane and visibility).

$E_{\text{regul}}(\mathbf{x})$ is higher-order in the the existence of a corner depends on 8 cell values. (We assume any 4 planes have no intersection.) As in [BdLGM14], we use a mixed-integer programming formulation of the regularization terms.

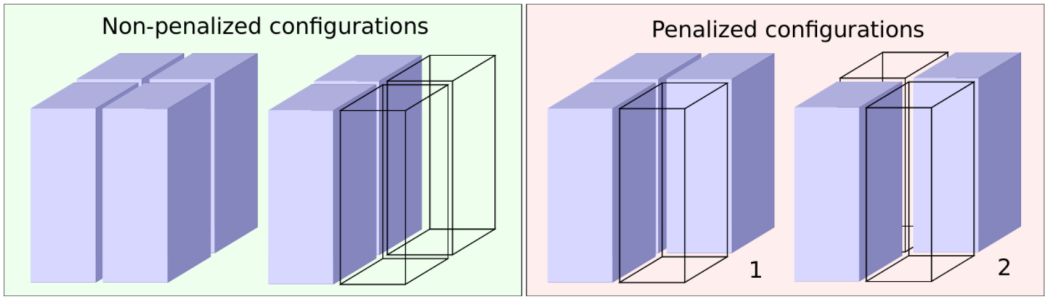


Figure 3.8: Non-penalized and penalized configurations of the 4 cells adjacent to each edge. Courtesy of [BdLGM14]

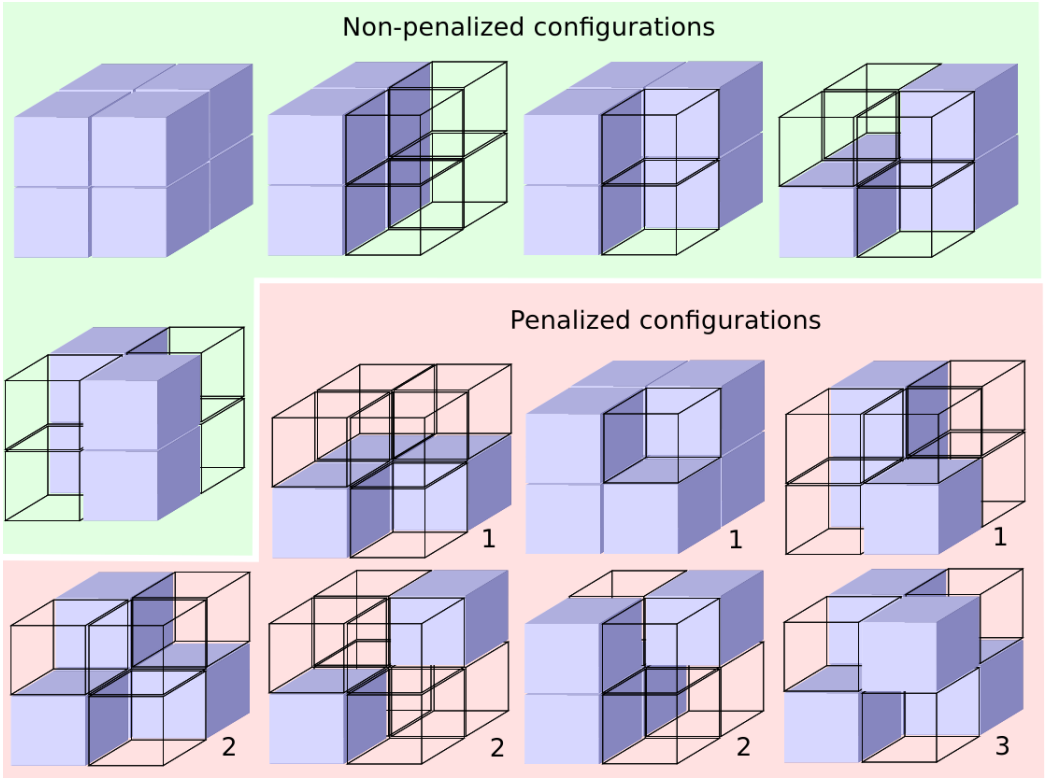


Figure 3.9: Non-penalized and penalized configurations of the 8 cells adjacent to each corner. Courtesy of [BdLGM14]

3.3.5 *Solving*

Due to the specific treatment of structural lines, the primitive energy term is not linear: it involves maximum values (cf. Eq. (3.2). However, the optimization problem

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \sum_{l \in \mathcal{L} \setminus \mathcal{L}_0} \sum_{v \in \mathcal{V}(l)} \sum_{s \in \hat{l}_v} \frac{|s|}{\sigma} \max(0, 1 - \sum_{\substack{c \in \mathcal{C} \\ c \cap (v \lhd s) = s}} x_c) \\ \text{subject to} \quad & 0 \leq x_c \leq 1, \ c \in \mathcal{C}. \end{aligned} \tag{3.4}$$

can be rewritten as a standard linear program by introducing for each max term a new slack variable $z_s \in \mathbb{R}$:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{z}}{\text{minimize}} \quad & \sum_{l \in \mathcal{L} \setminus \mathcal{L}_0} \sum_{v \in \mathcal{V}(l)} \sum_{s \in \hat{l}_v} \frac{|s|}{\sigma} z_s \\ \text{subject to} \quad & 0 \leq x_c \leq 1, \ c \in \mathcal{C}. \\ & z_s \geq 1 - \sum_{\substack{c \in \mathcal{C} \\ c \cap (v \lhd s) = s}} x_c \\ & z_s \geq 0 \end{aligned} \tag{3.5}$$

where $\mathbf{z}$ is the vector of all variables z_s, $v \lhd s$ is the visibility triangle formed by the line segment s and the point of view v and $\hat{l}_v$ is the set of subsegments of l as seen from viewpoint v.

The regularization term also is not linear. Minimizing the edges length and the number of corner indeed involves higher-order constraints: the presence of an edge (resp. corner) depends on the value of 4 (resp. 8) adjacent cells (full or empty). We reformulate these constraints, that are hard to solve, using linear terms only as proposed by [BdLGM14], i.e., using the absolute value of linear combinations of cell values, which can also be turned into an equivalent linear program with extra slack variables.

The resulting energy minimization problem is formulated as mixed-integer programming, with integral values (0 or 1) for the occupancy of cells x_c and continuous values for slack variables. As solving it is NP-hard, we also do as in [BdLGM14]: we relax the problem for optimization, i.e., the problem is solved for $x_c \in [0, 1]$ for all $c \in \mathcal{C}$. This corresponds to a linear program that can be solved efficiently using off-the-shelf solvers. The obtained fractional values for variables x_c are rounded independently of each other.

3.3.6 *Properties of reconstructed surface*

By construction, the surface we produce is watertight, even if the input data is very sparse, and not self-intersecting. Our process treats outliers (with RANSAC at plane detection stage, and regularization during reconstruction) and noise (with a model tolerance at plane detection stage and via projections when reconstructing). It has also several positive properties:

• *Insensitivity to line over-segmentation:* if a 3D line segment l is split, $E(\mathbf{x})$ does not change and thus the same surface is reconstructed. This provides robustness to over-segmentation, which is a common weakness of line segment detectors. (It may however change inlier-ness.)

• *Little sensitivity at endpoints*: given a line segment l, slightly changing its endpoint only makes a marginal change to $E(\mathbf{x})$. (Yet it may change inlier-ness too.)

• *Insensitivity to dummy planes:* given a 3D cell assignment $\mathbf{x}$, if an extra plane is randomly inserted in the arrangement, the value of $E_{\text{vis}}(\mathbf{x})$ does not change as it only depends on surface transitions encountered on visibility path.

4

Results

◂ Chapter 3
Chapter 5 ▸

4.1 DATASETS

We experimented both with real and synthetic data. The real datasets consist of images of a 'MeetingRoom' from [SMM17], of a 'Barn' from Tanks and Temples [KPZK17], of a 'DeliveryArea', a 'Bridge' and of a corridor named 'Terrains' from ETH3D [SSG+17]. All scenes are poorly textured (walls of uniform colors). The synthetic datasets include a 'TimberFrame' house [JKTS10] as well as two new synthetic datasets, that have been publicly released: 'HouseInterior' is a living room, with both large planar areas (walls, floor and ceiling) and smaller details (chair and table legs); 'Andalusian' is the outside of a modern house; it is piecewise-planar and uniformly white.

Table 4.1: Dataset statistics

Dataset	#img	$\lvert\mathcal{L}\rvert$	$\lvert\Pi\rvert$	$\lvert\Pi_{\text{fus}}\rvert$	$\lvert\mathcal{L}_0\rvert$	$\lvert\mathcal{L}_1\rvert$	$\lvert\mathcal{L}_2\rvert$	#res	Figure
Andalusian	249	1234	160	148	242	597	395	14503	4.1
DeliveryArea	948	1586	160	160	30	771	785	29222	4.2
Barn	410	7936	160	141	41	2157	5738	83989	4.3
TimberFrame	241	7268	140	131	264	4507	2497	79024	4.4
Bridge	110	7437	150	102	338	4168	2931	48315	4.5
MeetingRoom	32	831	135	130	25	383	423	9028	4.6
Terrains	42	3223	120	105	9	356	2858	18189	4.7
HouseInterior	159	1995	120	106	1	286	1708	18304	4.8

Number of images #img, number of 3D line segments $\lvert\mathcal{L}\rvert$, number of 3D planes before fusion $\lvert\Pi\rvert$, number of 3D planes after fusion $\lvert\Pi_{\text{fus}}\rvert$, number of segments supporting no plane $\lvert\mathcal{L}_0\rvert$, one plane $\lvert\mathcal{L}_1\rvert$ or two planes $\lvert\mathcal{L}_2\rvert$, and total number of sub-segments #res.

MeetingRoom was calibrated with LineSfM [Sal17, SMM17] and we recalibrated the other real datasets using COLMAP [SF16], with distortion correction as it impacts line detection. The synthetic datasets came with their exact calibration.

We then ran Line3D++ [Hof16], as defined in [HMB16], to detect and reconstruct 3D line segments.

Finally, we ran our plane detection and surface reconstruction, using a complete plane arrangement as baseline (see Sect. 4.7). Tab. 4.2 lists default parameters for all datasets. We often had to tweak σ_p of Line3D++ to get decent input lines, and sometimes our $\lambda_{\text{edge}} = \lambda_{\text{corner}}$. Tab. 4.1 reports detection statistics.

4.2 OBSERVATIONS ON THE INPUT DATA

4.2.1 *3D line segments detectors*

There has been some recent approaches to adapt structure-from-motion (SfM) methods from points to line segments [ZK14, SMM17]. However, few methods focus on the actual production of 3D line segments [JKTS10, HMB16], possibly reconstructing more lines than just what the SfM algorithms would produces, by leveraging on the calibration to match or significantly augment matched lines. To our knowledge, only Line3D++ [Hof16, HMB16] provides code for this denser 3D line segment reconstruction.

In our experiments, we thus use Line3D++ to detect 3D lines segments from a set of images, and to provide as well sub-segments associated the each viewpoints.

4.2.2 *Modest quality of input data from Line3D++*

Although quite efficient, Line3D++ produces a somewhat noisy output on which we have little control:

- It can miss some important lines (see Artwork 1 (a)), which may lead to pertinent planes not being detected.
- It also generate many outliers (see Artwork 1 (b)), which pressurize the visibility and regularization terms.
- A number of actual, physical 3D lines (including shadows) are given multiple reconstructions (see Artwork 1 (c)).

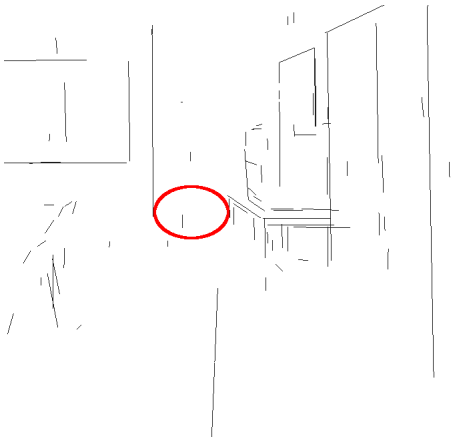
These flaws are also visible on the line segments extracted on our datasets as seen on Figures 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8.

We expect any improvement on these aspects, for a 3D line segment detector to be used as a preliminary stage to our method (which is out of the scope of this work), to have a particularly positive impact on our results.

4.3 QUALITATIVE EVALUATION OF RECONSTRUCTIONS

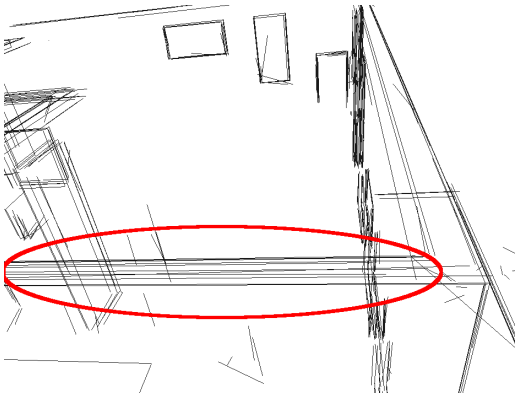
4.3.1 *Comparing to point-based reconstruction*

To show the relevance of lines for scenes with little or no texture, in contrast to point-based methods (which are doubtlessly superior on textured scenes), we compare our method to a point-based piecewise-planar reconstruction [CLP10] on HouseInterior (cf. Fig. 4.8). Even when densely sampling point on the ground-truth surface as seen from the viewpoints, [CLP10] yields a reconstruction with missing details (e.g., the lounge table) due to missing primitives in hidden area (e.g., under the table). Moreover, [CLP10] uses a regularization that minimizes the reconstructed area, which is relevant for points uniformly sampled on the surface but strongly penalizes unsampled regions (e.g., invisible planes of lounge table). In contrast, our method leads to a better plane discovery and a reconstruction robust to non-uniform sampling. (We also tried reconstructing from points sampled on the 3D lines, but the result is terrible; many planes are missed as points belong at most to one plane. As lines mostly lie on edges, the area cost also dominates the data term and

(a) Missing detections of 3D lines in the input: visible edge between floor and wall in a view of the MeetingRoom dataset (left), and extracted lines with Line3D++ where that important edge is missing to recover the floor (right).


(b) Spurious detections of 3D lines in the input: view from the MeetingRoom dataset (left), and reconstructed line cloud using Line3D++ where vertical outlier lines float above the table (right).

(c) Spurious duplication of 3D lines in the input: view from the Andalusian dataset (left), and the extracted lines with Line3D++ where some actual 3D lines in the scene are detected many times at slightly different locations (right).

Artwork 1: Line3d++ failing cases

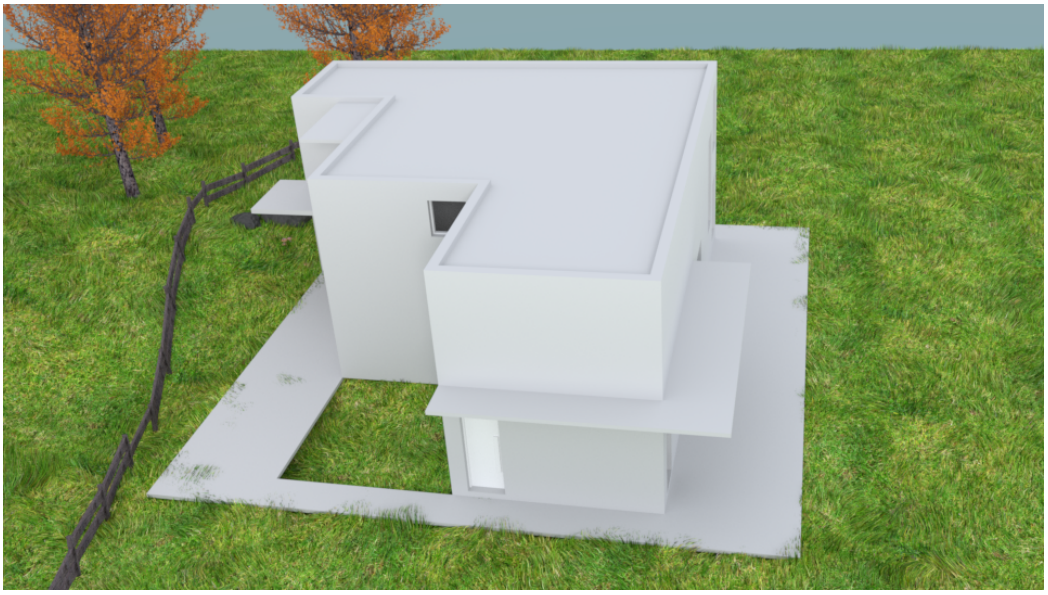
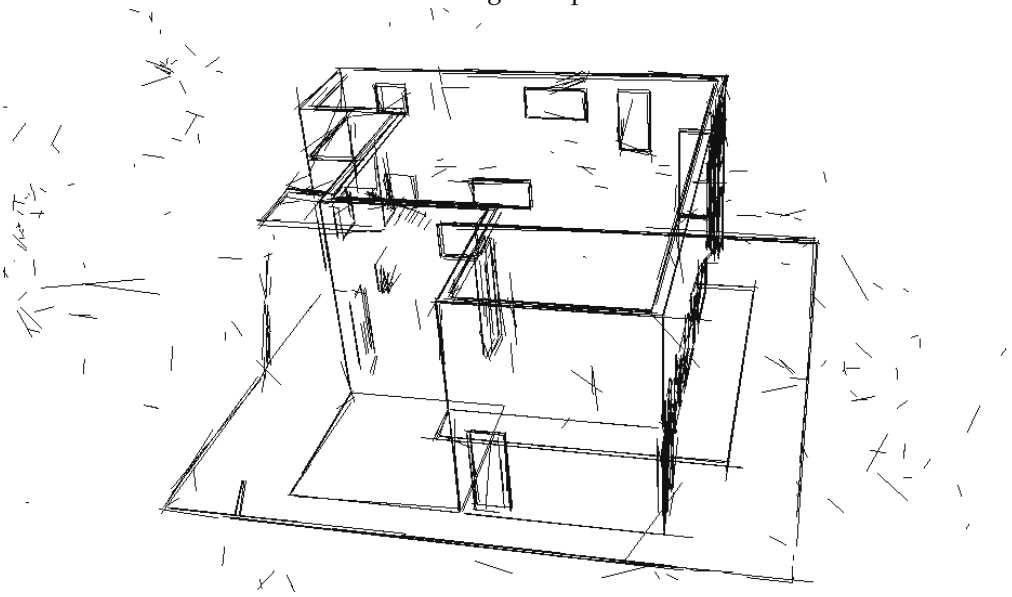
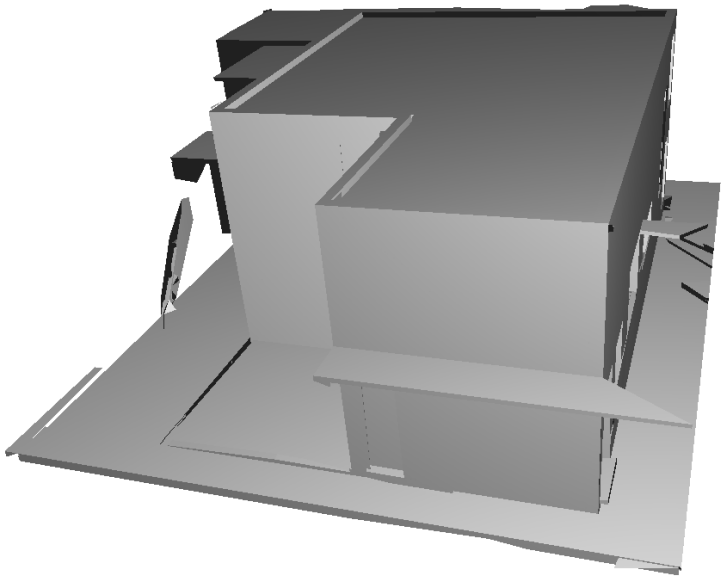


Image sample



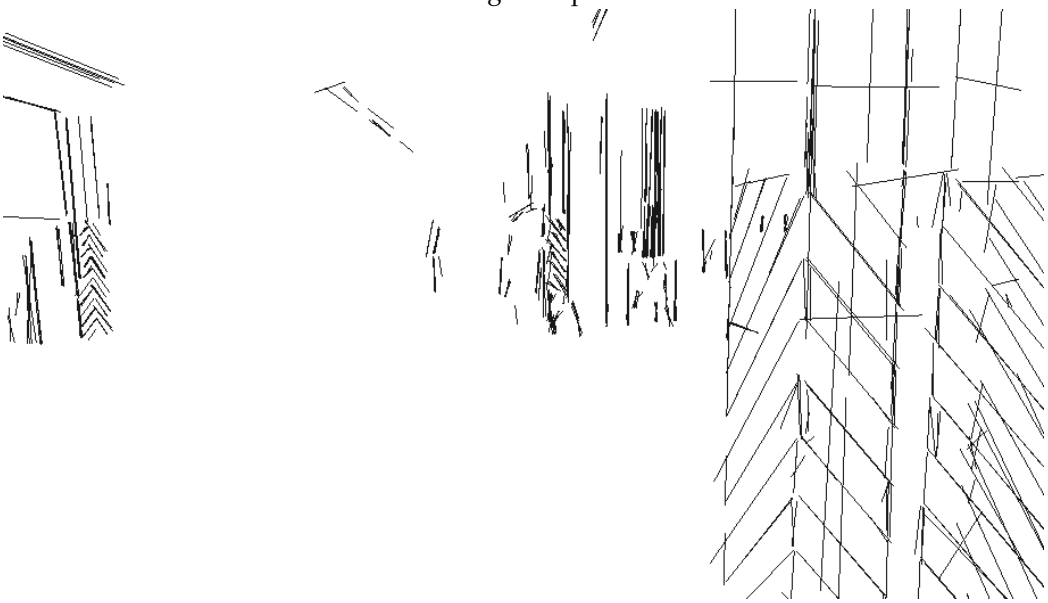
3D line segments



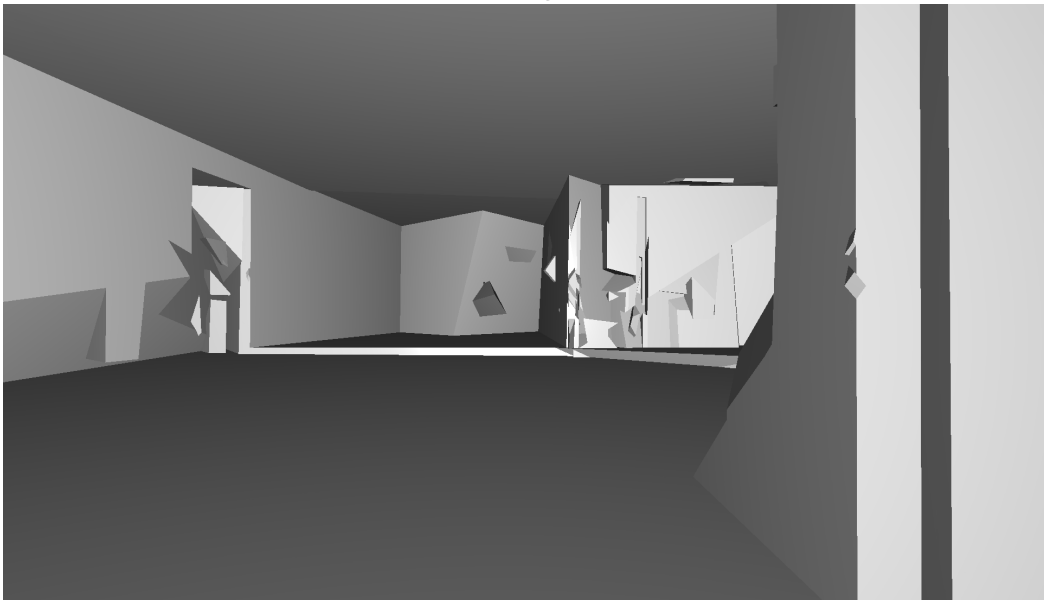
Our reconstruction

Figure 4.1: Results on Andalusian

Image sample



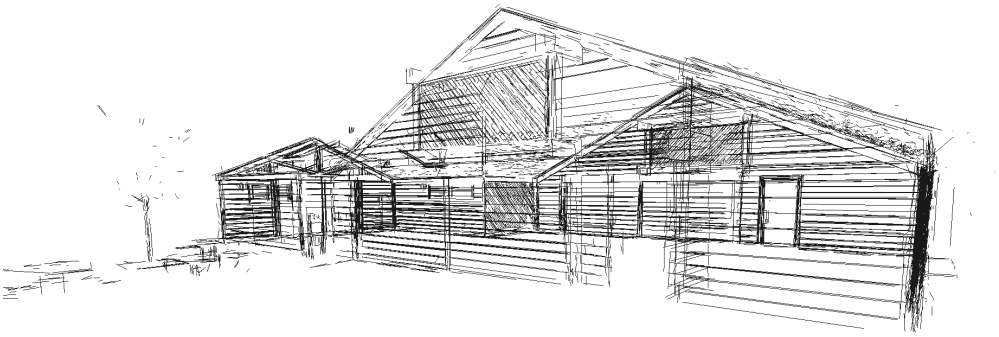
3D line segments



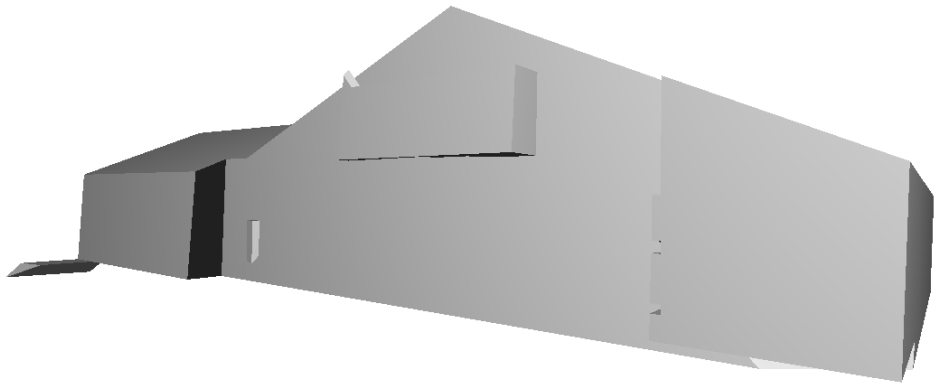
Our reconstruction

Figure 4.2: Results on Delivery Area

Image sample



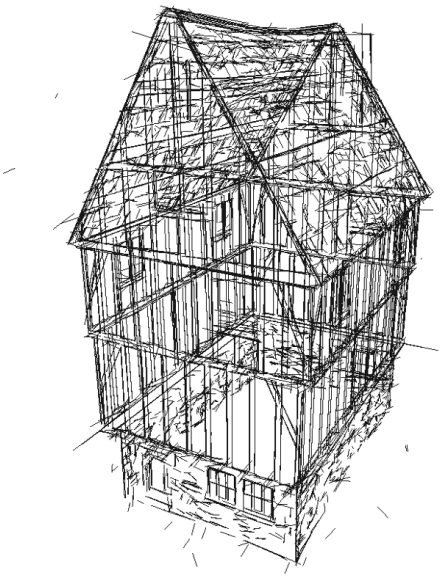
3D line segments



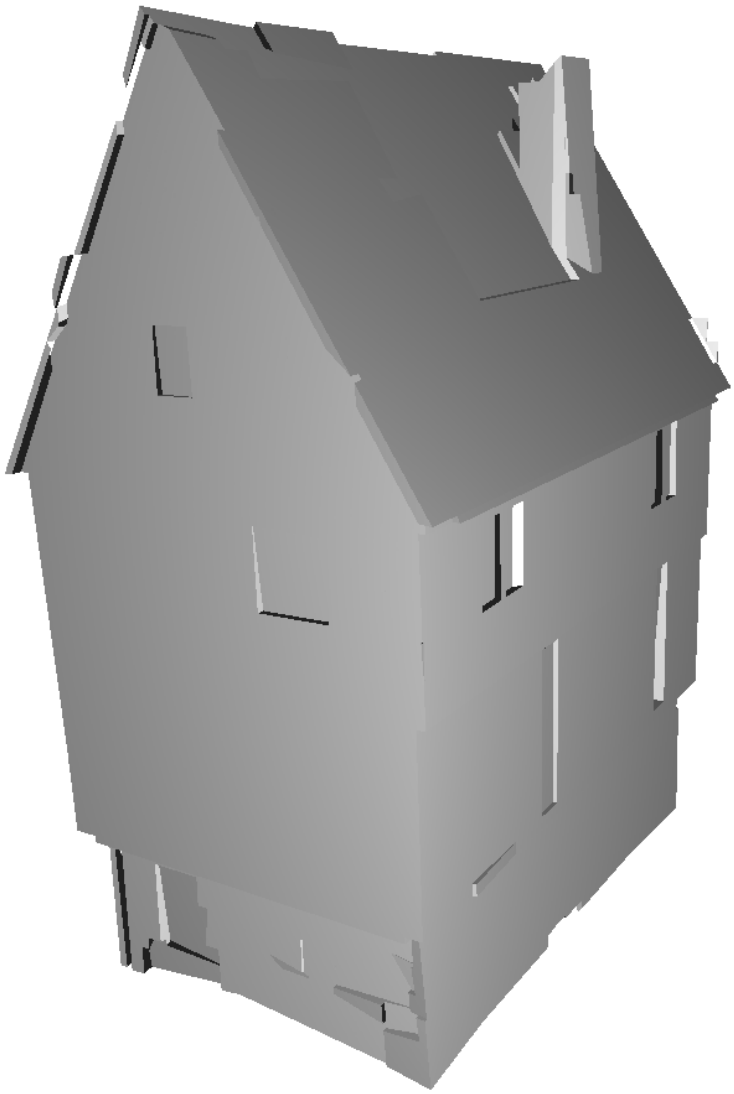
Our reconstruction

Figure 4.3: Results on Barn

Image sample



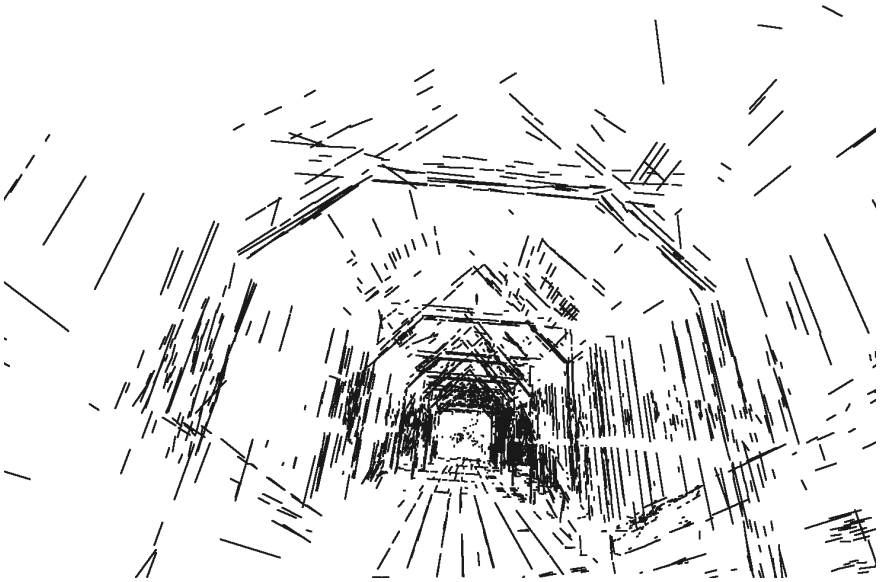
3D line segments



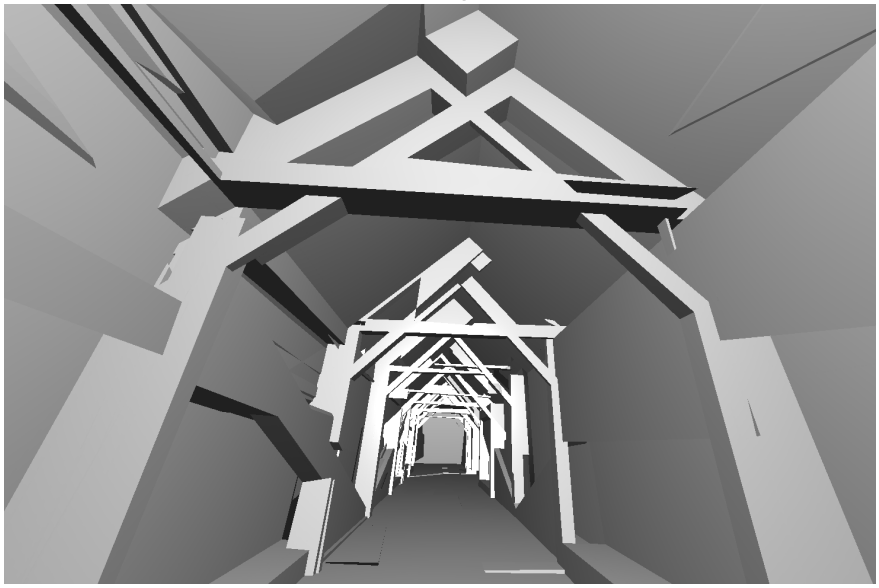
Our reconstruction

Figure 4.4: Results on Timberframe

Image sample



3D line segments



Our reconstruction

Figure 4.5: Results on Bridge

creates holes in large planar regions.) More comparisons, also with Colmap [SF16] and Polyfit [NW17], are on Figure 4.6 and Figure 4.7.

4.3.2 *Comparing to other line-based reconstruction methods*

As said above, there are very few reconstruction methods based on lines. [MG16] mostly reconstructs a soup of planes, sometimes with adjacencies, but without any topological guarantee. [WM14] provides a slightly more behaved mesh, but reconstructions still look messy and overly simple, although usable enough for robotic planning. No code nor data are available for comparing with either of these methods.

Our reconstruction with sparse data are illustrated on the datasets Andalusian (Figure 4.1), Delivery Area (Figure 4.2) and MeetingRoom (Figure 4.6). We also experimented on more textured datasets such as TimberFrame (Figure 4.4) and Barn (Figure 4.3) or less textured datasets such as HouseInterior (see Figure 4.8), and datasets with thin structural objects such as beams with Bridge (Figure 4.5).

Compared to usual point clouds, our 3D line clouds are extremely sparse. Despite the noise on inliers and the number of outliers due to Line3D++, our method is able to reconstruct a good approximation of the scenes, which illustrates the robustness of our approach. Still Barn shows that it is hard to reconstruct a sieve-like shape (balcony) due to the visibility lines traversing it.

4.4 QUANTITATIVE EVALUATION OF RECONSTRUCTIONS

We performed quantitative evaluations of the quality of our reconstructions using our datasets with ground truth. In this section, we describe the metrics we used in our experiments, as well as our setting and principles when varying a parameter or input data to study the sensitivity of our method (in following sections).

4.4.1 *Metrics to assess the quality of surface reconstruction*

We used 4 metrics derived from the Metro distance.

To compare a reconstructed mesh M_{recons} with a ground-truth mesh M_{gt}, we first sample 2 millions of points on the surface of each mesh. We then compute the distances of each point of M_{recons} to their nearest neighbour in M_{gt}; we note this set $D_{\text{recons}\to\text{gt}}$. Conversely, we also compute the set of distances $D_{\text{gt}\to\text{recons}}$.

The 4 metrics we use are the following:

- The max Metro distance measures the worst reconstruction error:

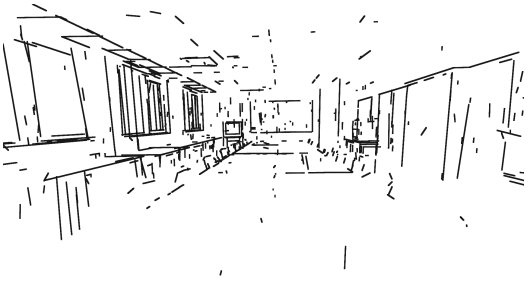
$$\max(D_{\text{recons}\to\text{gt}} \bigcup D_{\text{gt}\to\text{recons}})$$

- The mean Metro distance measures the average reconstruction error:

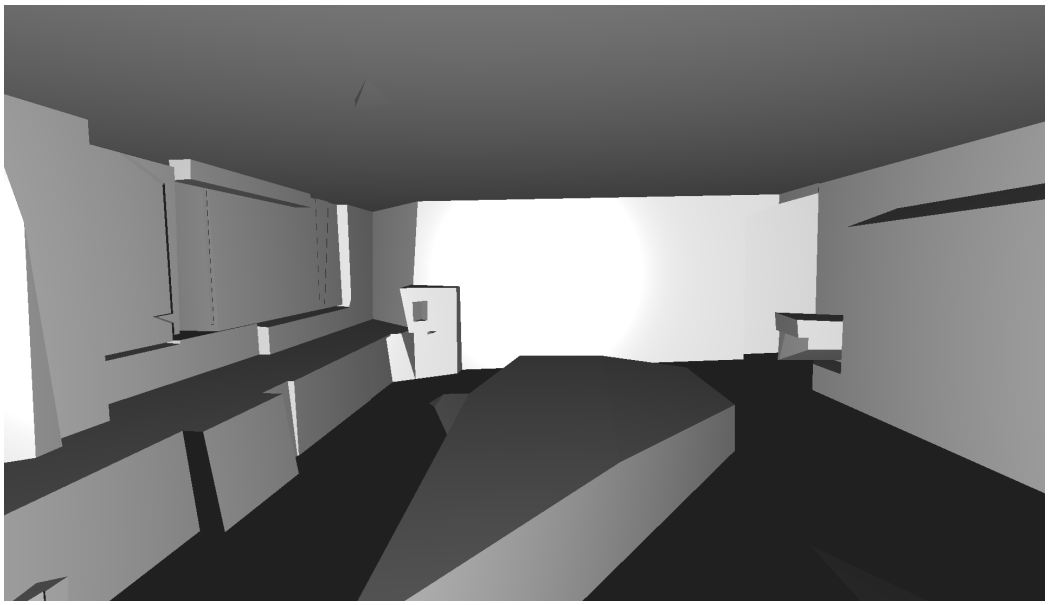
$$\text{mean}(D_{\text{recons}\to\text{gt}} \bigcup D_{\text{gt}\to\text{recons}})$$

- The "95%-completeness" is the 95% percentile of $D_{\text{gt}\to\text{recons}}$. It measures the distance under which most of the ground truth surface has been reconstructed.
- The "95%-precision" is the 95% percentile of $D_{\text{recons}\to\text{gt}}$. It measures the distance under which most of what has been reconstructed is close enough to the ground truth.

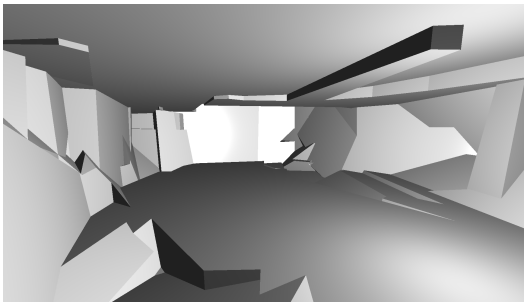
(1)



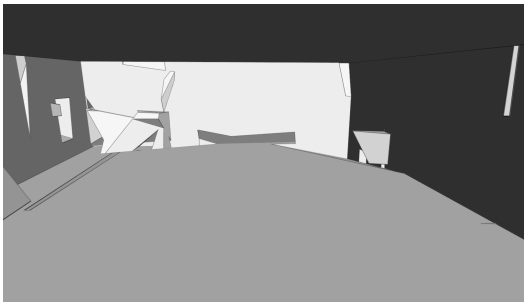
(2)



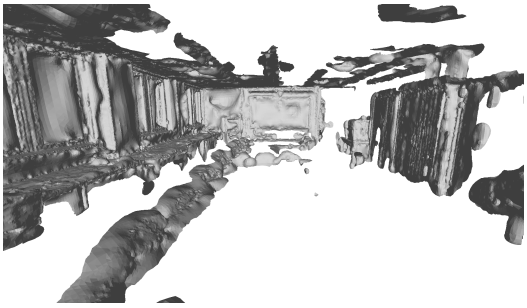
(3)



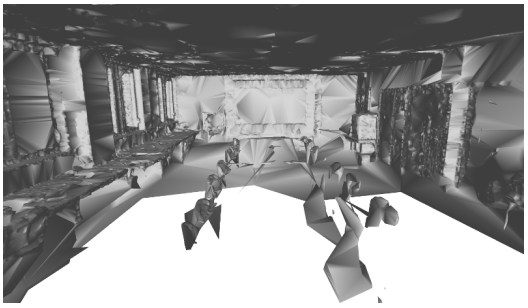
(4)



(5)



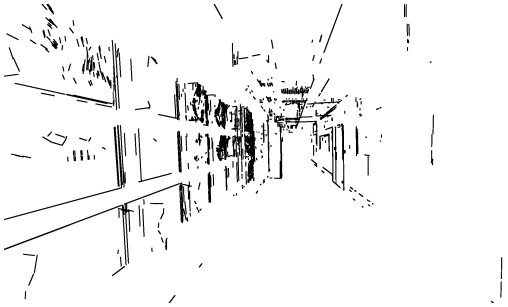
(6)



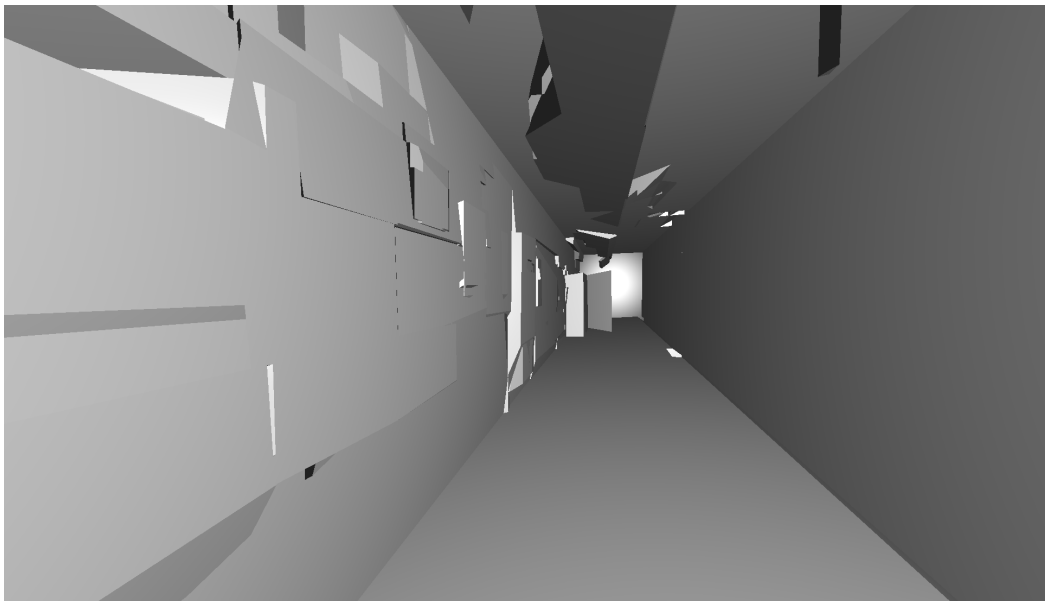
(7)

Figure 4.6: MeetingRoom: (1) image sample, (2) segments from Line3D++ [Hof16, HMB16], (3) our reconstruction, point-based reconstructions with Colmap [SF16] (4) then Chauve et al. [CLP10], (5) Polyfit [NW17], (6) Poisson [KBH06], (7) Delaunay [LPK09].

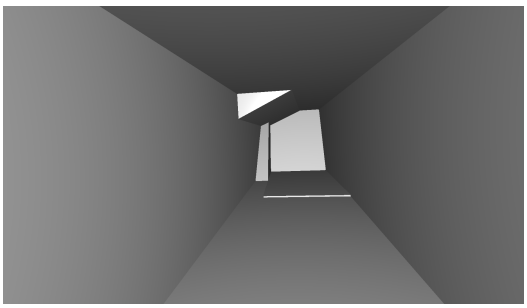
(1)



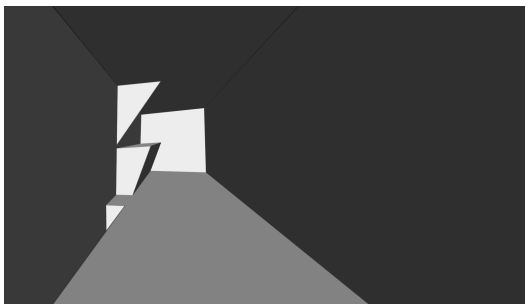
(2)



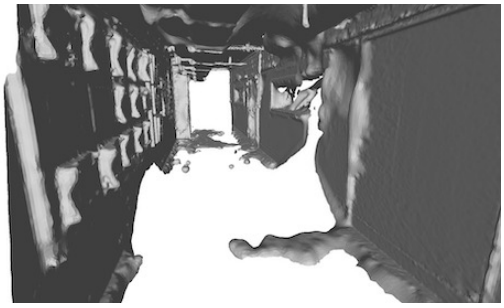
(3)



(4)



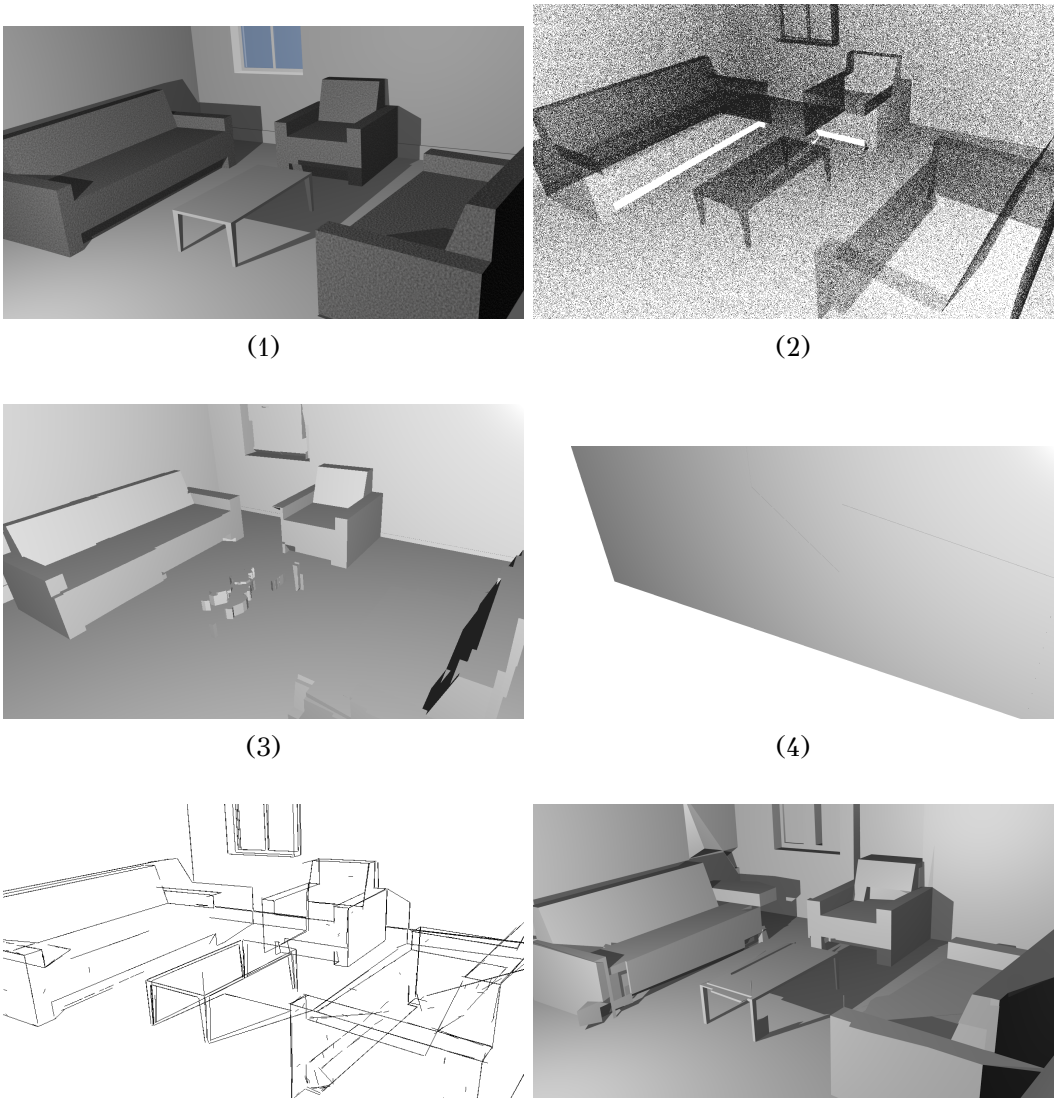
(5)



(6)

(7)

Figure 4.7: Terrains: (1) image sample, (2) segments from Line3D++ [Hof16, HMB16], (3) our reconstruction, point-based reconstructions with Colmap [SF16] (4) then Chauve et al. [CLP10], (5) Polyfit [NW17], (6) Poisson [KBH06], (7) Delaunay [LPK09].



(1) (2) (3) (4) (5) (6)

Figure 4.8: HouseInterior: (1) an image of the dataset, (2) points densely sampled on surface, (3) reconstruction with [CLP10], (4) failed reconstruction with [CLP10] from points sampled on lines, (5) 3D lines detected with Line3D++ [Hof16], with noise and outliers, (6) our reconstruction, which is nonetheless superior.

We use these metrics to assess the quality of reconstructed surfaces as well as to define the value of our parameters (see following sections).

Histograms of distances for HouseInterior are plotted on Fig. 4.9. Regarding precision, most of the points sampled on the reconstruction (91.4%) lie at less than 5 cm to the ground truth, showing that our RANSAC planes fit well the underlying surface and that our energy properly balances data fidelity and regularization. The error profile for completeness is similar, and 95% of the points on the ground truth are less than 8 cm to the reconstruction. It shows our regularization term do not over-smooth too much the surface by erasing details that would penalize completeness.

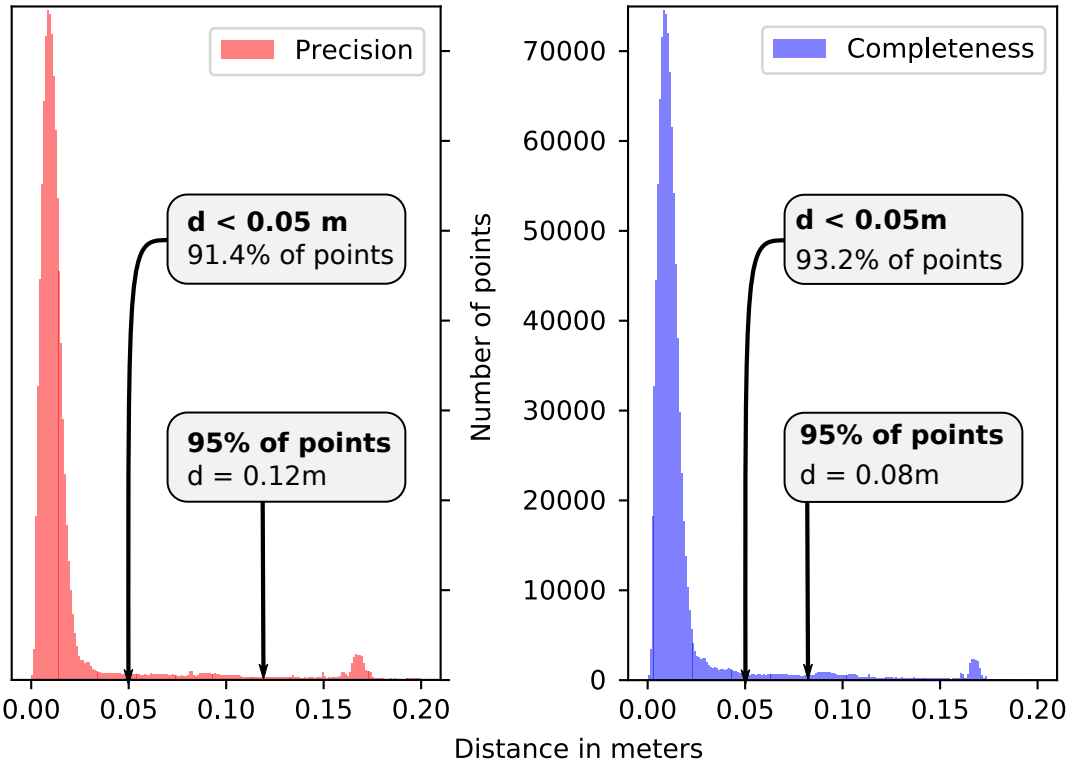


Figure 4.9: HouseInterior: histograms of distance errors w.r.t. ground truth (m).

4.4.2 *Varying parameters or input data*

In the following sections, starting from the default parameter setting in Table 4.2, we vary the value of a chosen parameter (e.g., λ_{vis}) or the quantity of input data (e.g., the number of detected 3D line segments), and we display a graph representing the impact on a quantitative assessment of the reconstruction.

For these experiments, we evaluate on the synthetic dataset HouseInterior, for which we have a ground truth. Although the results are specific to this dataset, we observed that the conclusions are relatively general. In particular, these variation studies on HouseInterior lead our choice of the best default parameter values (see Section 4.4.3), but we did not observe a strong need to alter this parameter setting when running on other datasets, although small changes could sometimes provide slightly better results.

Please note that although we may vary a parameter continuously, the labeling of the cells in the plane arrangement as full or empty is discrete and thus can lead to strong changes on the metrics when the altered cells are large or when the measure is based on a

maximum distance. As a consequence, curves showing the impact of parameter variations can display significant discontinuities.

Also, due to the RANSAC stage, our algorithm is not deterministic. For each set of parameters, we actually ran our method 5 times, and we report in the graphs both the average value of the quantity we monitor over these 5 runs as well as its standard deviation.

Impact of Line3D++ parameters. The 3D line segments were obtained using the default parameter settings of Line3D++ [Hof16]. We tried playing around with the parameters but did not get significantly better segments. In particular, we studied the influence of σ_p, a major regularization parameter of Line3D++, on the HouseInterior dataset. It is represented on Figure 4.10. The default value of Line3D++ for σ_p is 2.5, which is in a low plateau area of the graph.

Impact of the number of input images. We made a variant of the dataset MeetingRoom with 100 images and studied qualitatively the impact of providing a variable number of images as input (keeping a calibration based on all 100 images). Results are show on Figure 4.11 and compared to a point-based approach, namely Colmap [SF16] + Poisson [KBH06] reconstruction. When going down from 100 images to 50 images only, the quality of our reconstruction is progressively reduced, but the general shape of the room as well as a number of details are preserved. In contrast, the point-based reconstruction with 100 images is filled with holes and degrades rapidly when the number of images decreases.

Impact of the number of 3D line segments. We randomly sampled 3D line segments produced by Line3D++. The influence of the number of lines segments as input to our method is illustrated on Figure 4.12. As can be expected, the more lines, the better (in general), as it leads to a larger diversity of possible reconstructions, with more details. The mean Metro distance plateaus to a small value around 700 lines. Quite naturally, the max Metro distance remains sensitive until a few thousands lines are provided. 95%-precision also plateau a bit after a thousand lines, while 95%-completeness also retains a limited sensitivity. The computation time of the visibility term however increases, although mostly linearly in the number of lines.

4.4.3 *Parameter setting and sensitivity study*

We strove to reduce as much as possible the number of parameters of our method. Still, it has a few parameters, that have to be set. In this section, we study how to assign a value to these parameters, either using a formal argument (for the number of RANSAC iterations N_{iter}) or using an empirical justification (for the other parameters).

The parameter default setting is recalled in Table 4.2. Besides, the normalizing factor σ is set to 1 m.

ϵ	ϵ_{fus}	θ_{fus}	p_{fus}	N_{iter}	N_{max}	λ_{vis}	λ_{edge}	λ_{corner}
2 cm	$3\,\epsilon$	10°	20%	50k	160	0.1	0.01	0.01

Table 4.2: Parameters (all datasets have metric dimensions).

These parameters can be slightly adapted depending on the model. For instance, increasing λ_{vis} will dig more into the volumes, but will also make the model more sensitive to outliers. Likewise, increasing λ_{corner} and λ_{edge} will lead to more regularization, which is useful when data are missing, but can also lead to a loss of details.

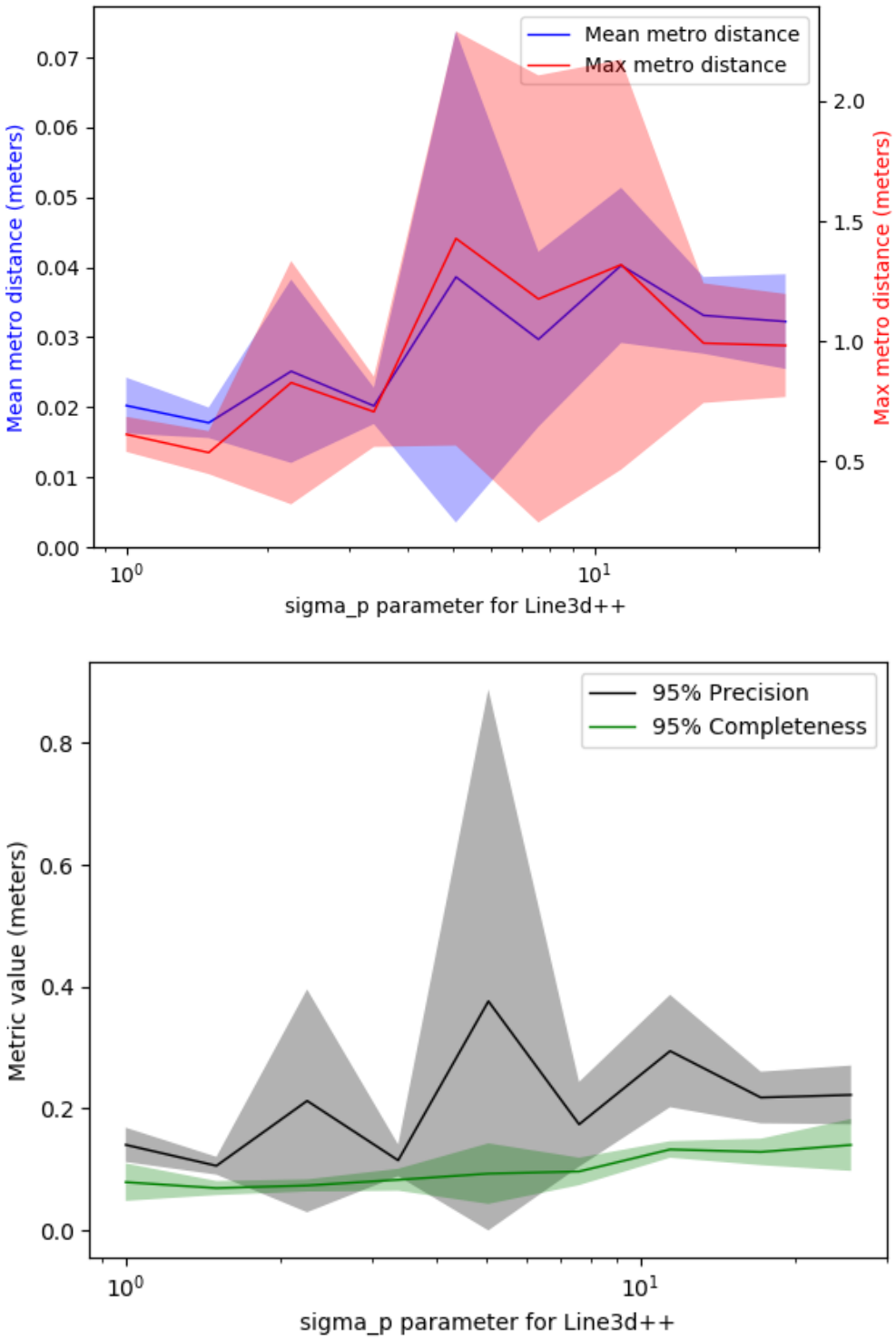


Figure 4.10: Impact of σ_p on the max / mean Metro distance (top), and the 95% precision / completeness (bottom).

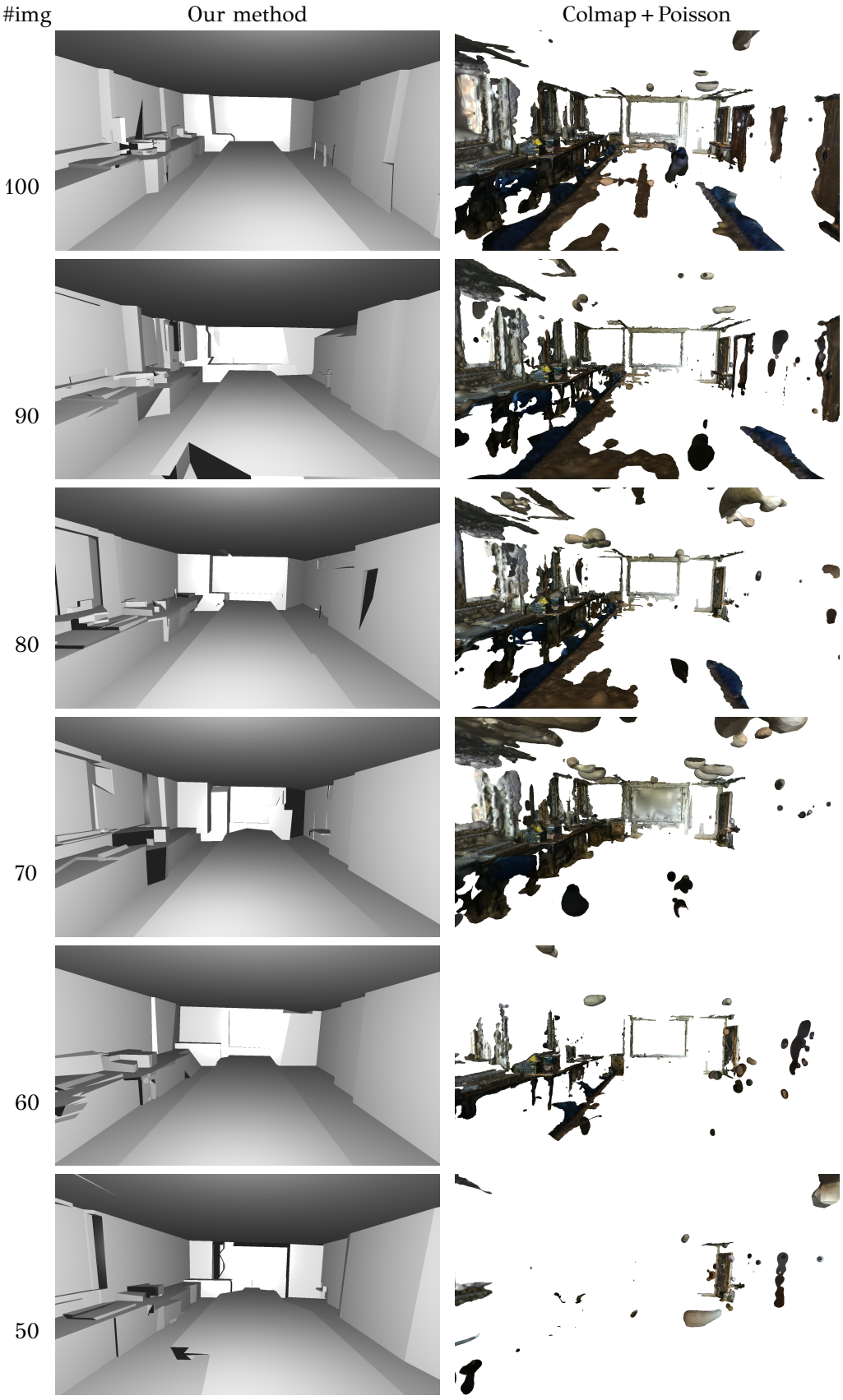


Figure 4.11: Variable number of input images: comparison of our method vs Colmap [SF16] + Poisson [KBH06] reconstruction on a variant of MeetingRoom with 100 images.

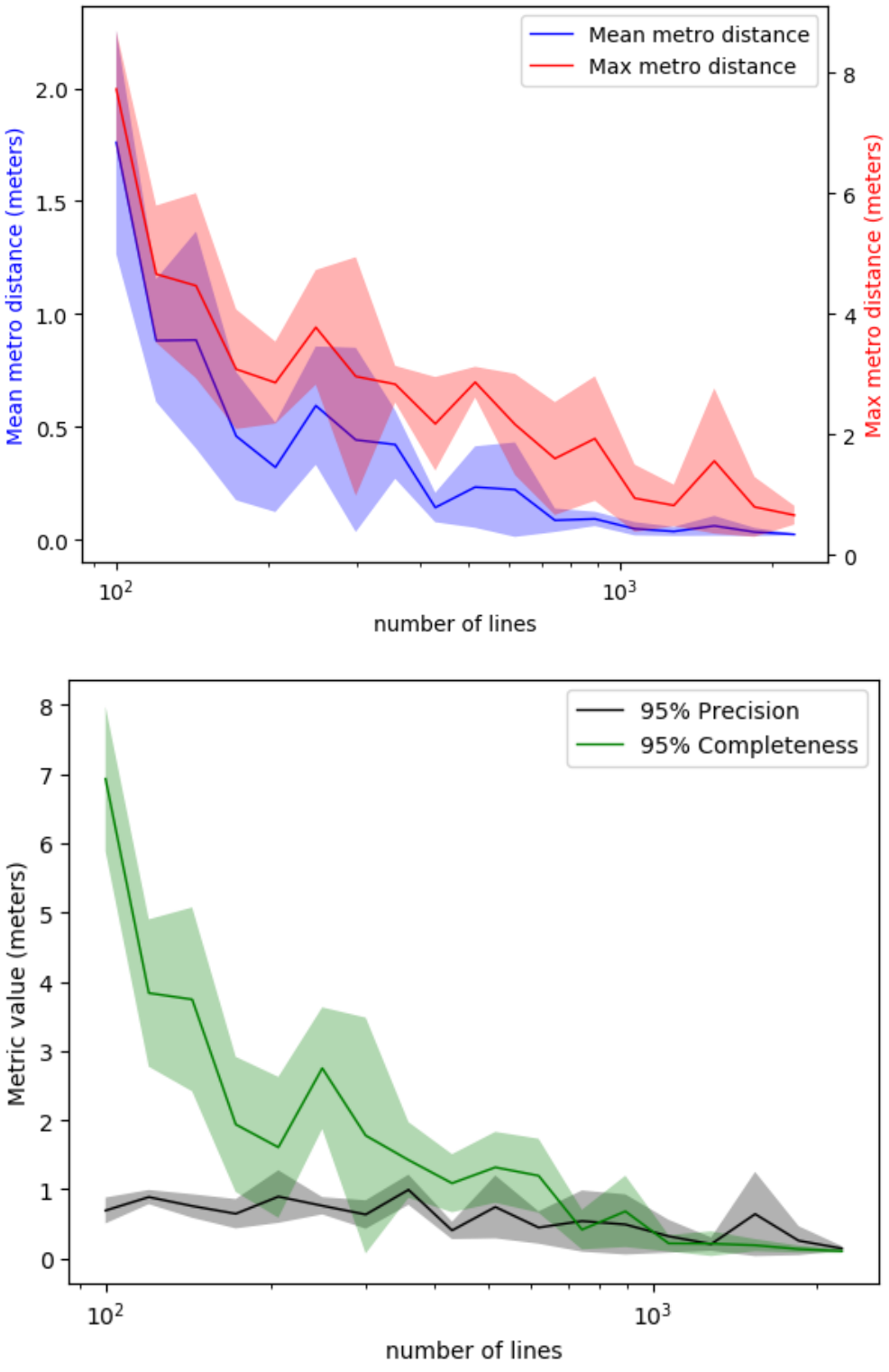


Figure 4.12: Impact of the number of input lines on the max/mean Metro distance (top), and the 95% precision/completeness (bottom).

In the following, we study the sensitivity of our method under different parameter settings, to discover ranges of parameter values leading to a good and relatively stable behavior. Table 4.2 resulted from this sensitivity study.

Number of RANSAC iterations N_{ITER}. Our choice for setting the value of the number of RANSAC iterations N_{iter} can be justified as follows.

The number of 3D line segments detected in the scenes of our datasets varies between 1,000 and 10,000, and the number of detected planes is in practice limited to about 150 (see Table 4.1). After the largest planes (with the most inliers) have been detected, the inlier rate of the current best plane can be quite low. Yet we must make sure that planes with a small line support are eventually detected, and first of all, sampled.

To define N_{iter}, we want to make sure at $\beta = 99\%$ chance that we sample the best plane assuming it is supported by at least $\alpha = 1\%$ of the line segments in the current value of $\mathcal{L}_0 \cup \mathcal{L}_1$. If we call X the event "not finding the best fitting plane after N_{iter} iterations among data which contain an inlier rate of α", its probability is:

$$P(X) = (1 - \alpha^k)^{N_{\text{iter}}}$$

where $k = 2$ is the number of line segment samples needed to generate a plane hypothesis. The criterion $P(X) \leq 1 - \beta$ in turn yields:

$$N_{\text{iter}} \leq \frac{\log(1 - \beta)}{\log(1 - \alpha^k)}$$

The right term evaluates to 46,049 with the given values for α and β. Given that the number of candidate line segments actually decreases at each iteration, as structural lines are detected, this is a worst case analysis.

In practice, in our experiments, we set $N_{\text{iter}} = 50000$. On a 12-core CPU, the whole RANSAC process (finding all planes) takes about 10 minutes for an upper bound of 10,000 lines with 50,000 iterations, which represents a minor fraction of the total reconstruction time.

Maximum number of detected planes N_{MAX}. We introduced a possible limit on the number of planes discovered by our RANSAC variant. The influence of that maximum number of detected planes N_{max} is shown on Figure 4.16.

As can naturally be expected, the more planes, the better. The mean Metro distance plateaus to a small value after 90 planes. So does the maximum Metro distance although with a small value but slightly larger variance. Both the 95%-precision and the 95%-completeness also plateau to a small value after 90 planes.

The only drawback of adding more planes, in the case of a surface reconstruction based on a full-extent plane arrangement, is the increasing computation time, as adding a plane has a cubic time complexity.

Weight of the visibility term $_{\text{VIS}}$. The impact of varying the value of parameter λ_{vis} is represented on Figure 4.13. We observe a plateau starting a bit before 10^{-1} (our default parameter), which starts deteriorating after 10^1 and even more after 10^2. As for the regularization parameters below, this observation qualitatively also applies to the other datasets.

Weight of the edge term $_{\text{EDGE}}$. The impact of varying the value of parameter λ_{edge} is represented on Figure 4.14. The error remains small when λ_{edge} is under 10^{-1}, in particular around value 10^{-2} (our default parameter).

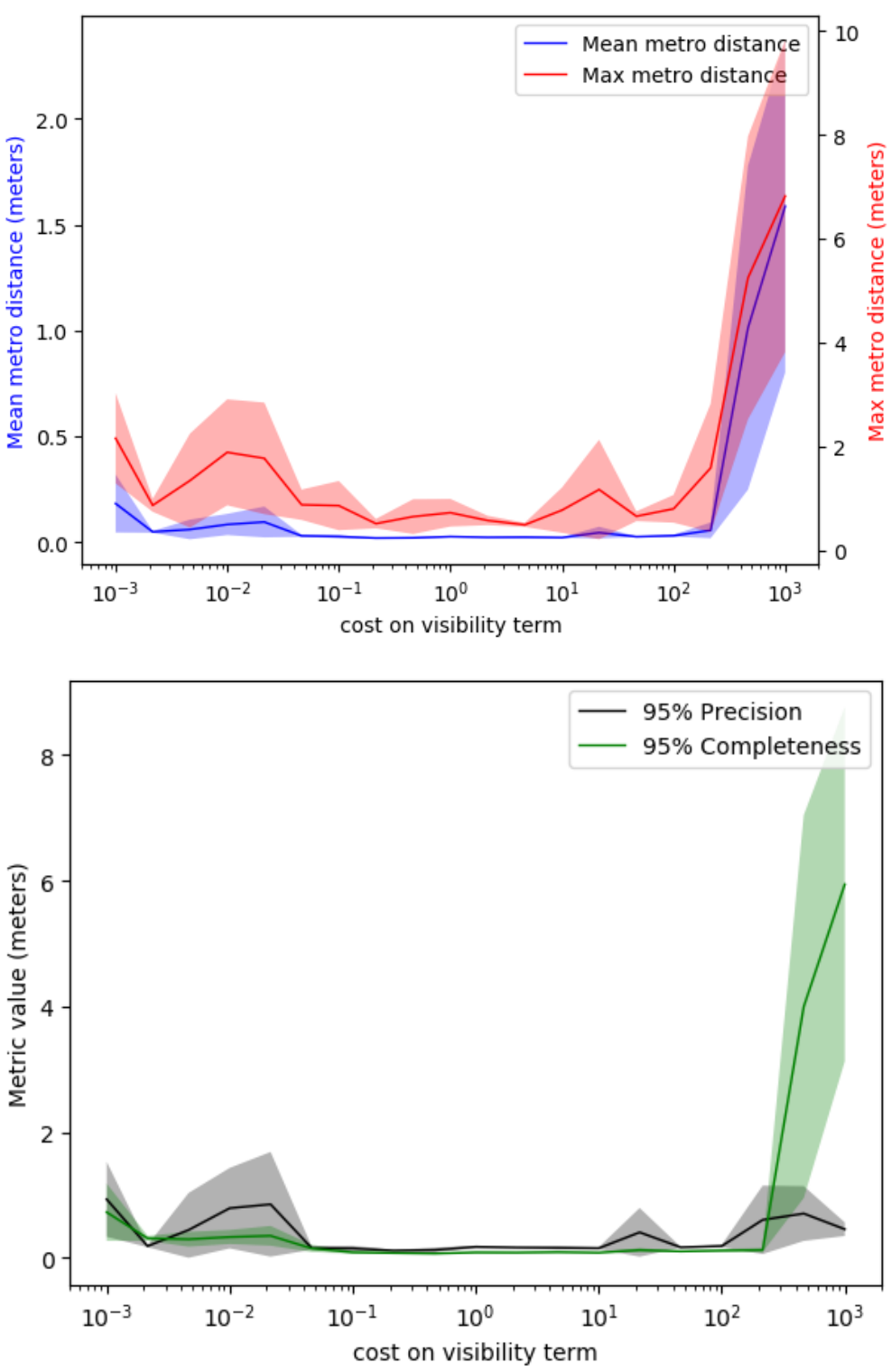


Figure 4.13: Impact of λ_{vis} on the max / mean Metro distance (top), and the 95% precision / completeness (bottom).

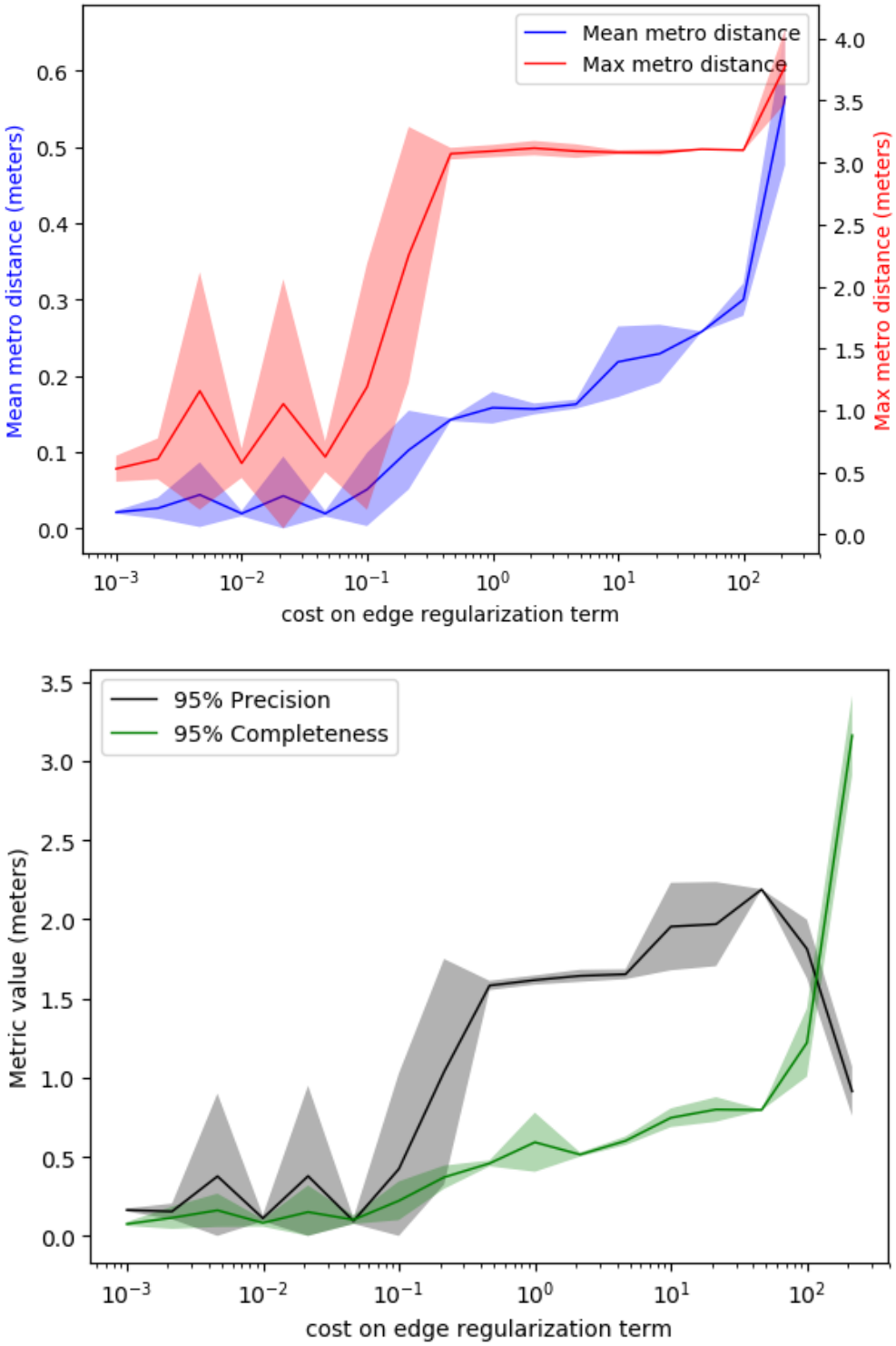


Figure 4.14: Impact of λ_{edge} on the max / mean Metro distance (top), and the 95% precision / completeness (bottom).

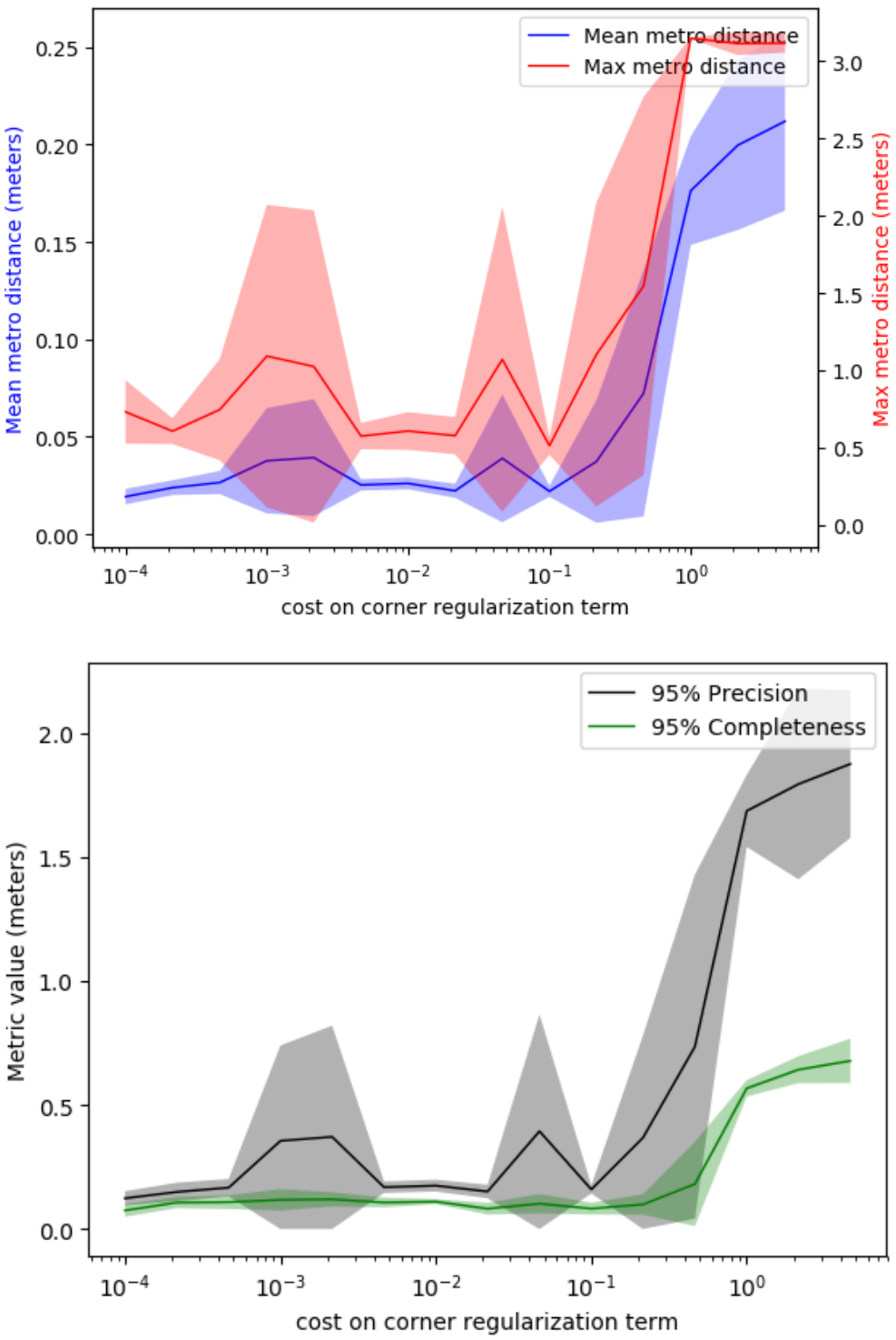


Figure 4.15: Impact of λ_{corner} on the max / mean Metro distance (top), and the 95% precision / completeness (bottom).

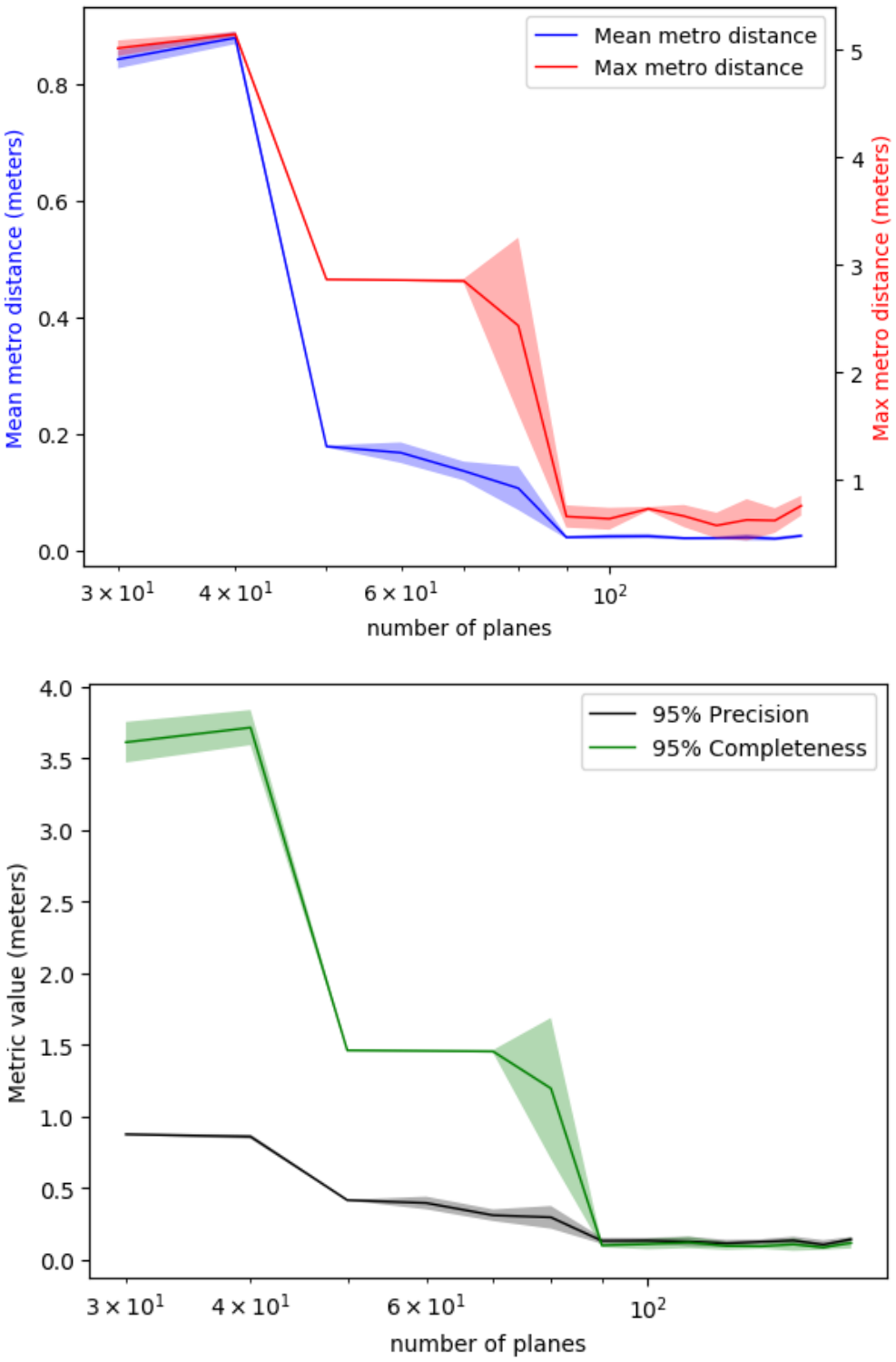


Figure 4.16: Impact of the number of planes N_{max} on the max/mean Metro distance (top) and on the 95% precision/completeness (bottom).

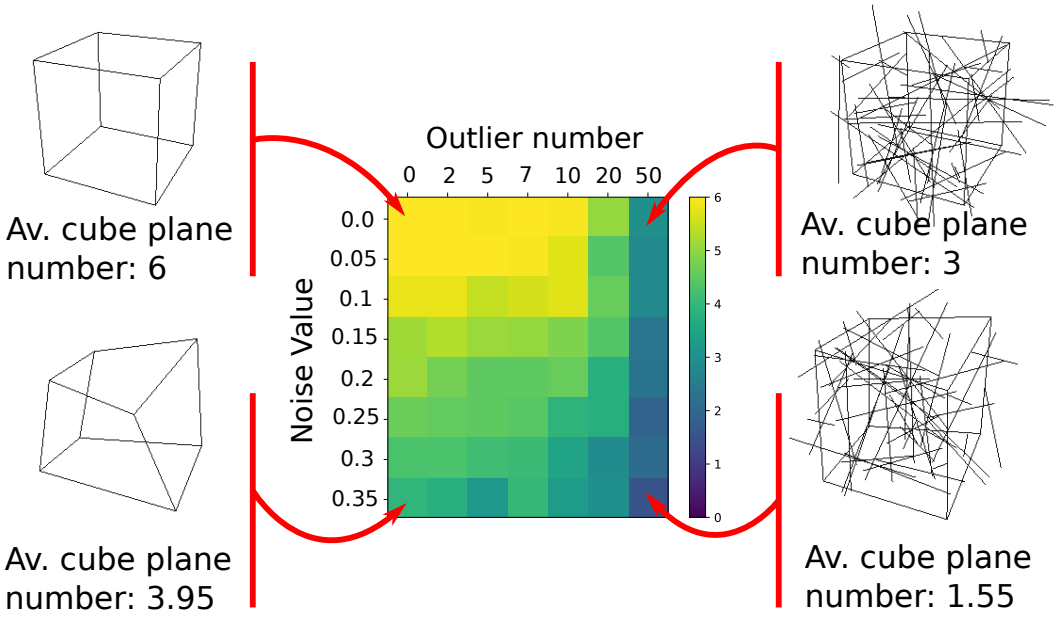


Figure 4.17: Robustness of RANSAC on lines for a cube defined by its edges. Value in grid is the average number of cube planes found, depending on the perturbation.

Weight of the corner term $_{\text{corner}}$. The impact of varying λ_{corner} is represented on Figure 4.15. As for λ_{edge}, the error remains small when λ_{corner} is under 10^{-1}, in particular on the plateau around value 10^{-2} (our default parameter).

4.4.4 *Computation times*

Although computing the visibility term is linear in the number of sub-segments, it is the most time-consuming part as it depends mostly on the number of cells in the plane arrangement, which is up to cubic in the number of planes. Time required for performing a whole reconstruction varies from 30 minutes (MeetingRoom) to 3 hours 30 minutes (TimberFrame). Creating the linear program from scene data takes more time than solving it.

4.4.5 *Robustness of plane detection*

To explore the robustness of our RANSAC formulation, we experimented with a toy example made of the 12 edges of a cube. We seek to extract the 6 planes associated to the 6 faces of the cube. We consider two types of perturbations: noise and outliers.

The cube has an edge length of 2. We add noise to each segment endpoint, drawn from a uniform distribution with standard deviation ranging from 0 to 0.35. Outliers, from 0 to 50, are segments generated by uniformly picking pairs of points in a 2-radius ball. Finally for each couple (noise, #outliers), we report the number of planes that include the 4 edges of an actual face of the cube, using parameters $\epsilon = 0.06$ and $N_{\text{iter}} = 100$, and averaging over 20 iterations.

Results are presented on Fig. 4.17. As expected, with a low level of perturbation, all planes are perfectly extracted. As the level of perturbation increases, for both noise and outliers, the rate of missed detections increases. Yet, even with a high level of noise, corresponding to a highly distorted cube (very non planar faces), we get a mean of 3.95 planes.

which fails miserably. Last, we compared with a regularization using only corners or

edges, which yields lower quality reconstructions.

4.5 ABLATION STUDY

To show that all line specificities in our method are useful if not crucial, we performed an extensive ablation study.

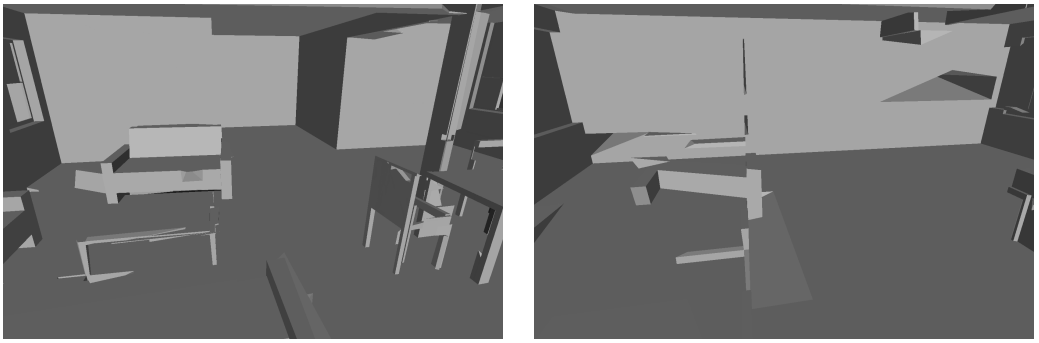


Figure 4.18: Reconstruction with our method where a line may support up to two planes (left, our RANSAC), or at most one plane (right, standard RANSAC).

IF ONE LINE SUPPORTS AT MOST ONE PLANE, as in an ordinary RANSAC framework, and if all lines supporting a detected plane are thus put aside before performing the following plane detection, as is the case in a greedy detection scheme, then fewer lines are available to detect succeeding planes, and less planes are detected. Experimentally, on HouseInterior, this detects only 45 planes instead of 120 with our method. The impact on reconstruction is illustrated on Figure 4.18: only the main planes are more or less reconstructed appropriately.

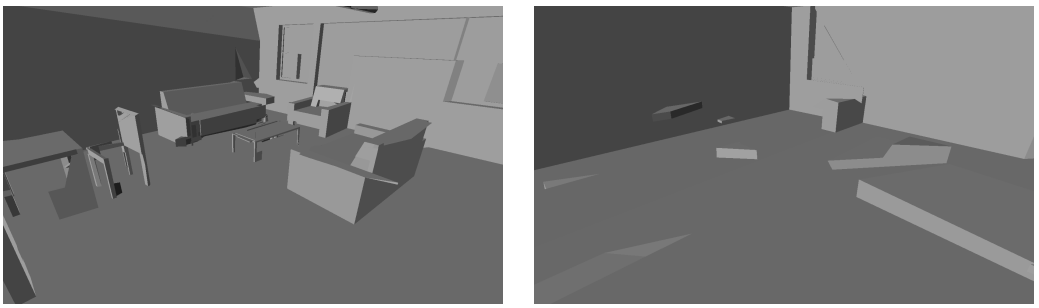


Figure 4.19: Reconstruction with our method where the gathering of inliers for a plane model relies on the distance to the plane intersection for lines already supporting a plane (left, our approach), and to the plane only (right, standard approach).

IF INLIERS ARE DECIDED ONLY FROM THE DISTANCE TO THE MODEL, rather than from the distance to the intersection of the plane model with the plane that already supports the inlier in case the line is structural, i.e., if we make no difference between lines in $\mathcal{L}_0$ and lines in $\mathcal{L}_1$ when gathering inliers for a given model (see Section 3.2), then the algorithm tends to associate a line twice with more or less the same plane. Experimentally, on HouseInterior, this detects only 92 planes, vs 120 with our algorithm. See Figure 4.19 for an illustration of the impact on the reconstruction.

IF STRUCTURAL LINES ARE TREATED AS TEXTURAL LINES, i.e., if a structural line in $\mathcal{L}_2$ is simply treated as two textural lines in $\mathcal{L}_1$ (one for each supported plane), then reconstruction totally fails (see Figure 4.20). This illustrates the importance of distinguishing structural lines form textural lines, not only for plane detection but also for surface reconstruction.

IF REGULARIZATION PENALIZES ONLY THE LENGTH OF EDGES, rather than only the number of corners, the reconstruction is not as good, as can be seen qualitatively on Figure 4.21. This

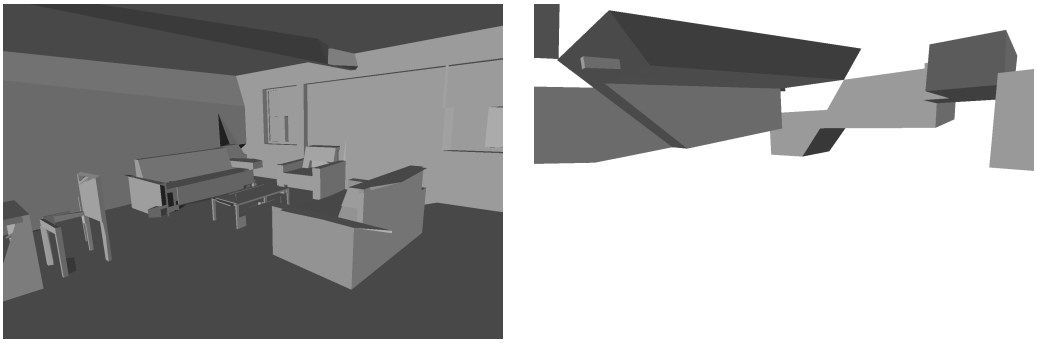


Figure 4.20: Reconstruction with our method where all structural lines are treated specifically (left), and considered as two textural lines, i.e., one for each supported plane (right).

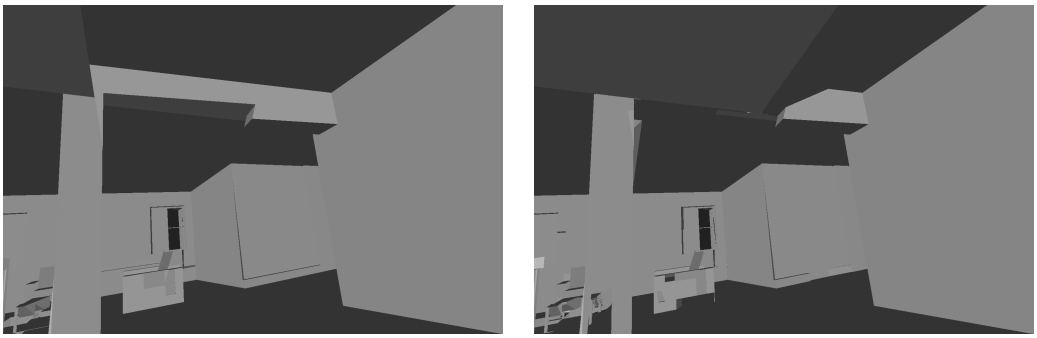


Figure 4.21: Reconstruction with a regularization on corners only (left), vs on edges only (right).

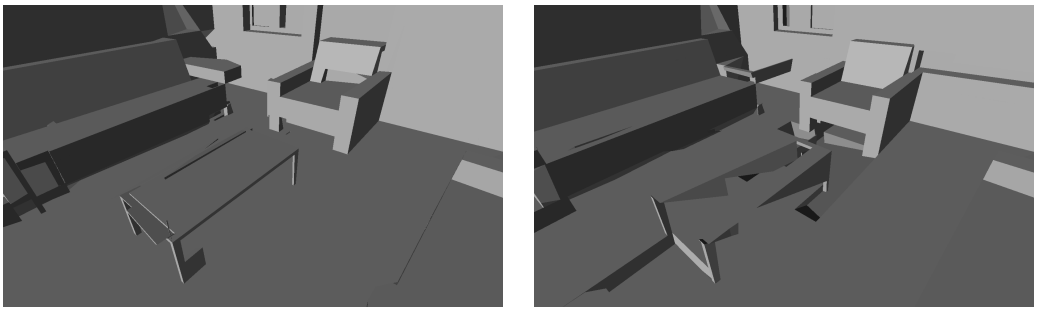


Figure 4.22: Reconstruction with a regularization on corners and edges (left), vs on corners only (right).

fact has already been observed by [BdLGM14]: providing a good balance between λ_{edge} and λ_{corner}, the regularization on both corners and edges is slightly better.

If regularization penalizes only the number of corners, rather than both corners and edges, the reconstruction also is not as good (ground floor), although it can be locally better (armchair), as can be seen qualitatively on Figure 4.22. This fact has also already been observed by [BdLGM14].

If regularization also penalizes the surface area, as a measure of surface simplicity like in [CLP10, BdLGM14], the metrics do not improve but rather tend to deteriorate, as can be seen on Figure 4.23. The fact is this term makes little sense on the kind of building scenes we are considering because we want to be able to reconstruct large surfaces with little data, that mostly lies on their boundaries only.

4.6 Limitations and perspectives

The quality of 2D and 3D line segments at input (from Line3D++) is the main bottleneck of our method. Improving them would be very helpful.

Mainly, it would be specially relevant to merge points and lines treatments into a single framework to offer a smooth transition from textured regions to textureless areas.

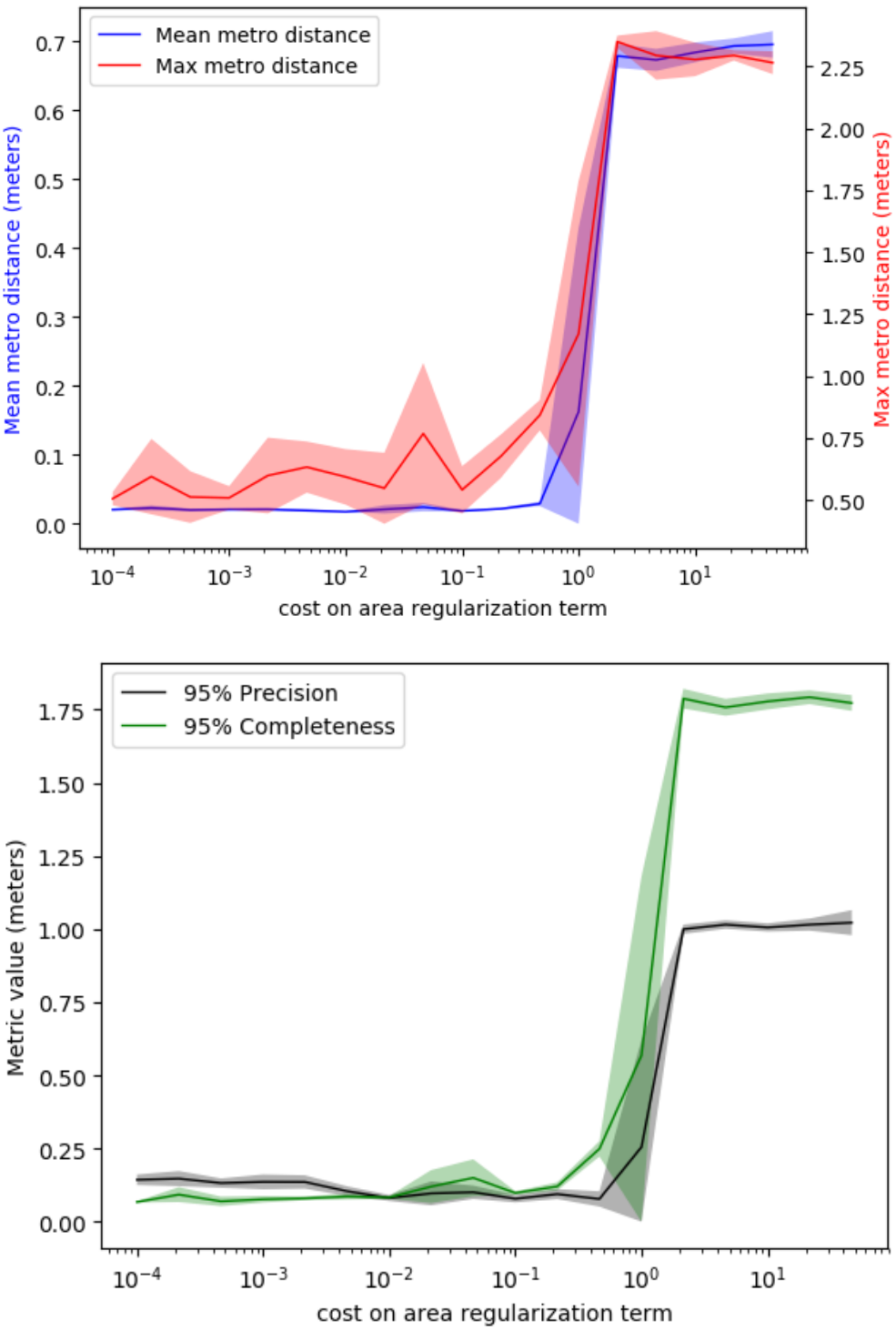


Figure 4.23: Impact of λ_{area} on the max / mean Metro distance (top), and the 95% precision / completeness (bottom).

Also, in our experiments, we used the full-extent plane arrangement, i.e., with planes extending all the way to the scene bounding box. This is not intrinsic to our method; it merely provides a baseline. Because of a cubic complexity in the number of planes, the acceptable number of planes is limited to a few hundreds, which is in practice often enough for a single room or the exterior of a building, but not enough for a complete BIM model. (It is also easy to keep the best few hundred planes after RANSAC detection to make sure the pipeline succeeds.) Yet, preliminary experiments with a coarse-to-fine approach show promising results for scaling to large scenes. In the cell complex, limiting the plane extent with a heuristic on a coarse voxel-based partition [CLP10] or the 3D adaptation [BL20] of the 2D kinetic polygonal plane partitioning [BL18] would also reduce the complexity.

Moreover, defining a notion of extent for line-detected planes, similar to α-shapes in the case of points [EKS83] but adapted to lines [vKvLV11, vKvLV13], could also be used to introduce so-called 'ghost planes', corresponding to unobserved, hidden planes at occluding edges of observed surfaces [CLP10, BdLGM14].

Last, global regularization weights favor highly sampled surfaces. Adapting them to be more sensitive to weakly supported surfaces as in [JP11] could improve the results. They also need to be adjusted for each scene to perform the best reconstruction; thus, automatizing the parameters search would greatly improve the convenience of use of the pipeline.

4.7 CONCLUSION

In this work, we proposed a surface reconstruction pipeline from images using 3D line segments. The whole approach takes into account the line set properties: small size, sparsity, density variations. We first presented a RANSAC approach designed for plane extraction in such line set. It is robust to over segmentation of lines. Then we adapted a surface reconstruction method using a plane arrangement to fit line specificities. In particular, we designed the data term for multiple viewpoints and insensibility to dummy plane insertion. Finally, we experimented on both synthetic and real datasets show the practical interest and robustness of our method.

Future work also includes finding a mean to better normalize the regularization parameters, although we believe there will still be some dependency on the type of scene, in particular depending on whether the scene is inside or outside. More general perspectives include developing a general framework accepting both 3D points and line segments and being able to smoothly adapt from point-based paradigm on textured scenes [CLP10] with proper regularization [BdLGM14], to line-based reconstruction on textureless areas.

One of the main perspective is to create a unified approach to deal with heterogeneous line/point clouds. It would involve developing a new criteria for plane extraction and reformulating the energy for surface extraction to be well balanced between lines and points.

Transition

In this part, we have used a direct method to perform 3D reconstruction of buildings. We tackled specific issues met in building environments and paved the way towards an efficient algorithm. However, as discussed, limitations remain. In particular, some parameters must be adapted for each scene, and there is no semantic information in the output. Both problems require strong and complex priors to be solved. With the growing amount of data publicly available online and the generalization of machine learning techniques throughout all computer vision tasks, a promising way to build strong priors is to directly learn them from the data. In the next part, we conduct experiments on a dataset of objects, in order to study how one can learn reconstruction priors and incorporate geometric constraints to enhance the overall resulting mesh.

Part III

3D RECONSTRUCTION BY PARAMETERIZED SURFACE MAPPING

5

Introduction

◂ Chapter 4
Chapter 6 ▸

Synopsis We introduce an approach for computing a 3D mesh from one or more views of an object by establishing dense correspondences between pixels in one or different views and 3D locations on a learnable parameterized surface. Our insight is to optimize a novel geometric cycle-consistency loss between the learnable parameterized surface and input views over an embedded representation of the target shape, allowing the reconstruction of a 3D shape that closely aligns with its depiction in one or more views. In addition, we introduce a multi-view shape encoder that can jointly be trained with the AtlasNet surface parameterization. We demonstrate the efficacy of our approach on the ShapeNet-COCO dataset and show that our approach outperforms XNOCS [SRV+19] when averaging over three shape categories.

5.1 Learning 3D reconstruction

Figure 5.1: Input: a set of 5 calibrated low-resolution images with arbitrary backgrounds. Output: a 3D mesh

Reconstructing a shape from one or more views is a long-standing problem in computer vision [SCD+06]. When we, as humans, are relatively far away from an object where binocular parallax is negligible, we are nonetheless able to infer a representation of its geometry, including its hidden faces, and as we move closer, we are able to incorporate information from the new viewpoints and from both eyes. An ideal system should be able to reconstruct a shape, including inferring hidden faces, from a single view and update the reconstruction as more image evidence is gathered. To achieve this goal, a system must integrate information from a learned prior for the shape (single-view reconstruction) with geometric constraints when more than one view is available (multi-view reconstruction). This goal is important for scanning, augmented reality, and robotic navigation and manipulation applications.

For shape reconstruction, we argue that the choice of shape representation is important. In this work, we seek to output a surface-parameterized mesh reconstruction of a shape from one or more input views. As we will show, the parameterized surface representation provides a natural medium for establishing dense correspondences across views and allows leveraging recent work that learns surface-parameterized shape priors [GFK+18].

Moreover, the representation lends itself to rasterization, which makes it possible to reason about contours. In contrast, other shape representations, such as points, voxels, and implicit surfaces, do not easily offer such benefits.

Classical multi-view 3D reconstructions approaches establish dense correspondences between pixels or image regions depicting the same surface locations, followed by triangulation to yield a 3D shape [SCD+06]. However, these approaches fail to find correspondences in flat, non-textured regions, and they cannot extrapolate to reconstruct nonvisible/occluded surfaces. In addition, they typically yield point clouds, whereas meshes are more useful in most real-world downstream applications.

Incorporating data-driven shape priors into the 3D reconstruction process has been used to address the above limitations [GFK+18, PFS+19]. However, an ideal shape prior should be able to adapt in order to finely align with its depiction in the input views [LWR+19]. Furthermore, the prior should not require manual annotation of the surface parameterization, i.e., landmarks of shape parts, and should be learnable from training data.

To address the above issues, we propose a method that learns to establish dense correspondences between image pixels and a learned parameterized surface, which is trained on a corpus of shapes from the same class, without known surface parameterizations. The approach has two steps. First, as opposed to the method of Kulkarni *et al*. [KGT19] that uses a fixed template, we generate an initial shape template with a multi-view shape encoder, which extends AtlasNet [GFK+18] to multiple viewpoints. Second, this template, specific to the shape we want to reconstruct, is refined using a cycle-consistency loss which allows the shape to adapt and finely align with its depiction in the input views.

We evaluate our method compared to a recent state-of-the-art approach for 3D reconstruction from multiple views [SRV+19]. We show that our method outperforms the baselines when averaging over three shape categories. Moreover, as opposed to XNOCS [SRV+19] and traditional multi-view stereo methods [SCD+06], our shape parameterization naturally yields a mesh that can be used out-of-the-box for most graphics applications.

5.2 Related work

Single-view reconstruction. Single-view scene reconstruction is a classic problem in computer vision. Most successful method learn 3D orientation priors over images or image patches that are trained for example by supervised learning [SSN09, EPF14].

Recently, methods have tried to relax the explicit 3D supervision requirement by learning class-specific reconstruction priors over, for example, a deformable mesh [KTEM18]. This approach generates impressive textured reconstructions from a single image, but requires extra supervision in the form of keypoints in correspondence during training, whereas we learn these as well.

More recently, Canonical Surface Mapping (CSM) [KGT19] predicts UV mapping from a single image onto a canonical model, trained entirely using self-supervision (via the images themselves) by introducing a geometric cycle-consistency term. Our work is inspired by this, but differs in two key ways. First, we do not assume a fixed canonical model, rather we optimize over a learned surface parameterization to better match the observed data. Second, CSM works only for single views, whereas we develop a multi-view cycle-consistency term to better integrate multi-view information when available. This is critical when reconstructing both the mapping and the underlying model jointly, due to the ill-posed nature of the problem.

Data-driven multi-view reconstruction. When given multiple views, it is possible to infer scene geometry by finding correspondences subject to geometric, e.g., epipolar constraints. Classical approaches such as multi-view stereo (MVS) [SCD+06] are powerful and well established tools with many uses. However, MVS fails in cases where correspondences are ambiguous, e.g., occluded surfaces, texture-less regions, and view-dependent effects. To overcome these limitations, these methods often employ weak smoothness priors. Recently, data-driven methods have been introduced that attempt to extend MVS to leverage much stronger priors to deal with ambiguities [KHM17, HMK+18]. Our work differs from these works in that we reconstruct a mesh rather than a point cloud, which typically requires stronger data priors, as seen in the single-view setting.

Other approaches for learning mesh reconstruction rely on manual alignment between the cameras and shape generator and optimize a photometric loss, which is not robust to illumination changes [LWR+19]. Alternately, Pixel2Mesh++ [WZLF19] presents a feed-forward neural network that maps from input views and a base sphere mesh to an output adapted mesh. However, this approach differs fundamentally in that it does not establish a dense surface mapping from pixels to the reconstructed mesh, which is required for fine scale alignment.

More recently, a number of methods for multi-view novel view synthesis have proposed eliminating the explicit mesh reconstruction altogether and instead learn features, for example, in a voxel grid [STH+19, LSS+19] or a set of orthogonal planes spaced in 3D [ZTF+18]. These approaches demonstrate strong view synthesis results, but cannot be used in cases where mesh reconstructions are still required.

6

Method

◂ Chapter 5
Chapter 7 ▸

6.1 LEARNING A MULTI-VIEW PARAMETERIZED SURFACE MAPPING

We consider a shape S and a collection $\mathcal{V}_S$ of views of S, where a view $v=(I,\pi)\in\mathcal{V}_S$ consists of an image I with associated camera intrinsics and extrinsics π. We seek to output a surface mesh $\mathcal{M}$ of the shape in object coordinates. We assume that a template $\mathcal{T}$ parameterizes the output surface mesh and that the mapping $\mathcal{V}_S\rightarrow\mathcal{T}\rightarrow\mathcal{M}$ produces the output. Our goal is to learn this parameterized surface mapping from image pixel locations to the output surface. For this, we present a two-step approach, depicted in Figure 6.1. The first step (initialization) aims at estimating an initial template, represented by a latent vector $\mathbf{z_S}$ and the weights of an AtlasNet decoder ϕ [GFK+18]. The second step (optimization) improves the shape representation using a cycle-consistency loss.

6.1.1 *Initialization*

Multi-view shape encoder. To compute the initial latent shape feature $\mathbf{z_S}$, we propose a multi-view shape encoder $\mathcal{E}$ jointly trained with the shape parameterization ϕ, cf., Fig. 6.2.

For each view $v=(I,\pi)\in\mathcal{V}_S$, the image I is encoded using a convolutional neural network $\mathcal{C}$. Now the resulting features are expressed in a coordinate system linked to the view. Therefore, we apply a neural network $\mathcal{R}$ that transforms the encoded representation with respect to the view camera parameters π in a representation expressed in the canonical pose. The multi-view encoder $\mathcal{E}$ aggregates by average pooling the features extracted from each view. The resulting representation ensures equivariance with respect to the camera parameters and can handle a variable number of input views. Denoting $|\mathcal{V}_S|$ the size of the view collection $\mathcal{V}_S$, the output of our multi-view shape encoder $\mathcal{E}$ is the average of encoded transformed views:

$$\mathbf{z_S}=\mathcal{E}(\mathcal{V}_S)=\frac{1}{|\mathcal{V}_S|}\sum_{(I,\pi)\in\mathcal{V}_S}\mathcal{R}(\mathcal{C}(I),\pi). \tag{6.1}$$

We assume here a spherical parameterization of a shape collection $\mathcal{S}$: the template $\mathcal{T}$ we consider is a sphere, which covers the range of genus-zero shapes. We learn our parameterization using an AtlasNet decoder ϕ. To do so, we need to establish correspondences between each shape $S\in\mathcal{S}$ in the collection. We achieve this goal by establishing per-shape correspondences $\mathcal{Q}_S$ via the template $\mathcal{T}$. While there are a variety of spherical parameterizations of a shape [PH03], we use a gnomonic projection due to its simplicity (see Figure 6.3).

We jointly optimize the parameters of $\mathcal{E}$ and ϕ using the shape prior loss $\mathcal{L}_{\text{shape}}$ as the sum of L_1 losses over shapes $S\in\mathcal{S}$ and established 3D point correspondences $(\mathbf{q},\mathbf{q}')\in\mathcal{Q}_S$ between the template $\mathcal{T}$ and the shape S:

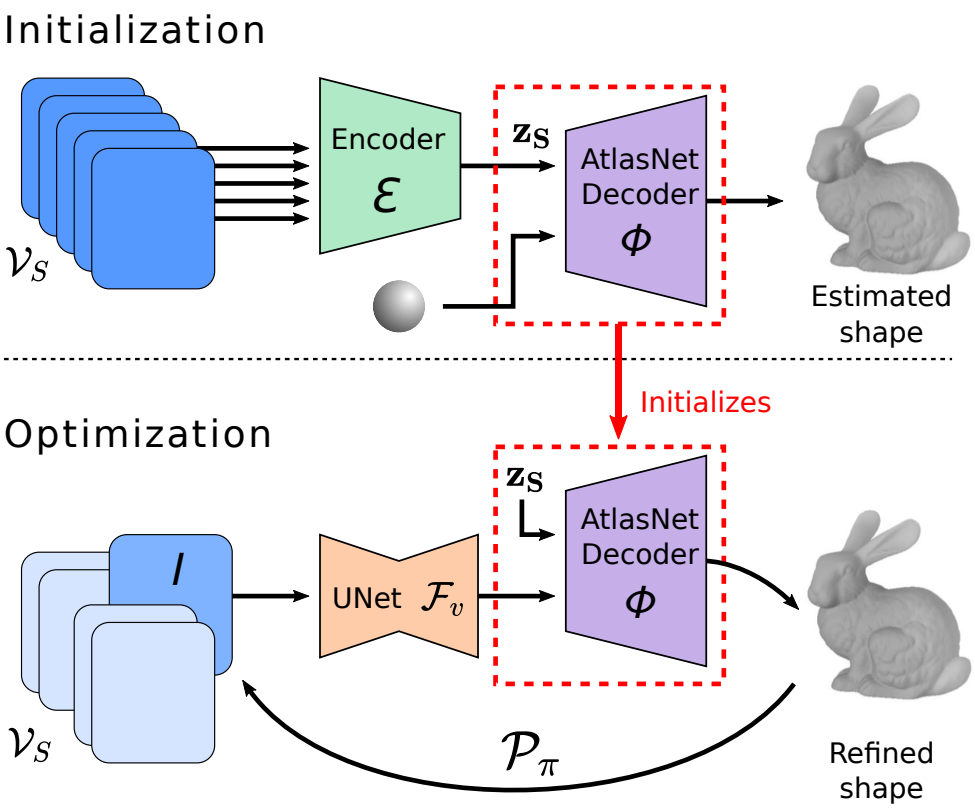


Figure 6.1: **Networks for inferring a depicted shape.** Our pipeline consists of two steps. First, a trained encoder generates an initial latent shape representation from the calibrated images. Next, the decoder is iteratively improved through geometric and mask constraints to further improve reconstruction.

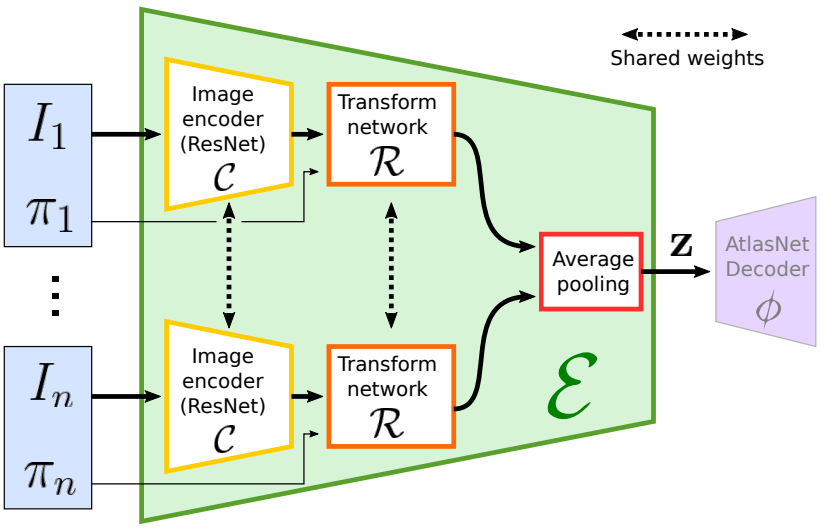


Figure 6.2: **Multi-view shape encoder.** Our multi-view shape encoder $\mathcal{E}$ aggregates transformed features given any number of input views and corresponding camera parameters. A decoder ϕ then maps the encoded representation to the output mesh.

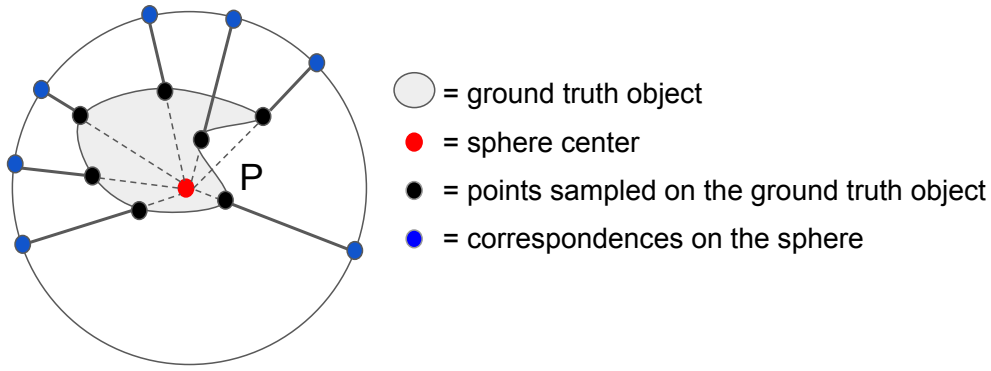


Figure 6.3: The gnomonic mapping: each point sampled on the ground truth shape is projected on the sphere using a central projection.

$$\mathcal{L}_{\text{shape}}(\phi, \mathcal{S}) = \sum_{S \in \mathcal{S}} \sum_{(\mathbf{q}, \mathbf{q}') \in \mathcal{Q}_S} \left\| \phi_{\mathcal{E}(\mathcal{V}_S)}(\mathbf{q}) - \mathbf{q}' \right\|_1 \tag{6.2}$$

Training procedure. Given a dataset of 3D shapes, we render multiple views and jointly train the parameters for the multi-view shape encoder $\mathcal{E}$ and the parameterization ϕ. We follow the network architectures and two-stage training procedure described in Groueix *et al.* [GFK+18] to train a single-view model comprising the parameterization ϕ and single-view encoder $\mathcal{C}$ using loss $\mathcal{L}_{\text{shape}}$. We then jointly train the multi-view shape encoder and parameterization by initializing the parameters for ϕ and $\mathcal{C}$ from the single-view training step.

In our implementation, $\mathcal{C}$ is a ResNet-18 [HZRS16], $\mathcal{R}$ is a multi-layer perceptron with two layers and the AtlasNet decoder ϕ is a multi-layer perceptron with four layers.

6.1.2 *Optimization*

Given the estimated shape produced at the initialization, it is possible to further improve the reconstruction.

Similar to CSM [KGT19], we employ cycle-consistency in our reconstruction loss to learn the parametric surface mapping: after mapping a pixel location $\mathbf{p}$ in an image to the surface mesh via the template, it should project back to the original location $\mathbf{p}$. We further encourage that the projected point lies within a segmentation mask $\mathcal{X}_v$ of the depicted shape, in each view v (see Figure 6.5). The segmentation mask can be obtained with an off-the-shelf instance segmentation algorithm [HGDG17].

The cycle-consistency loss is graphically described on Figure 6.4. Let $v = (I, \pi)$ be a view in collection $\mathcal{V}_S$ and $\mathcal{F}_v : \mathbb{R}^2 \to \mathcal{T}$ be a learnable mapping from a 2D pixel location $\mathbf{p}$ in image I to a point on template $\mathcal{T}$. We note $\mathcal{P}_\pi : \mathbb{R}^3 \to \mathbb{R}^2$ the function projecting a 3D point on the image plane and define the cycle projection function $\mathcal{K} : \mathbb{R}^2 \to \mathbb{R}^2$ as $\mathcal{K}(\mathbf{p}) = \mathcal{P}_\pi \circ \phi_{\mathbf{z_S}} \circ \mathcal{F}_v(\mathbf{p})$.

Our reconstruction loss is the sum of squared-reprojection errors and squared-Chamfer distances to the segmentation mask over all pixels, weighted by $\lambda = 0.25$:

$$\mathcal{L}_{\text{rec}}\left(\mathcal{F}_v, \phi_{\mathbf{z_S}}\right) = \sum_{\mathbf{p} \in \mathcal{X}_v} \left(\|\mathcal{K}(\mathbf{p}) - \mathbf{p}\|_2^2 + \lambda \min_{\mathbf{p}' \in \mathcal{X}_v} \|\mathcal{K}(\mathbf{p}) - \mathbf{p}'\|_2^2 \right) \tag{6.3}$$

Shape refinement procedure. We initialize the latent vector $\mathbf{z_S}$ and ϕ with the weights obtained at the initialization step. In our implementation, the mapping $\mathcal{F}_v$ is a 4-level UNet [RFB15] with a ResNet backbone. $\mathcal{F}_v$ has 3 output dimensions which are normalized in order to represent a UVW mapping on the template sphere $\mathcal{T}$. To properly initialize

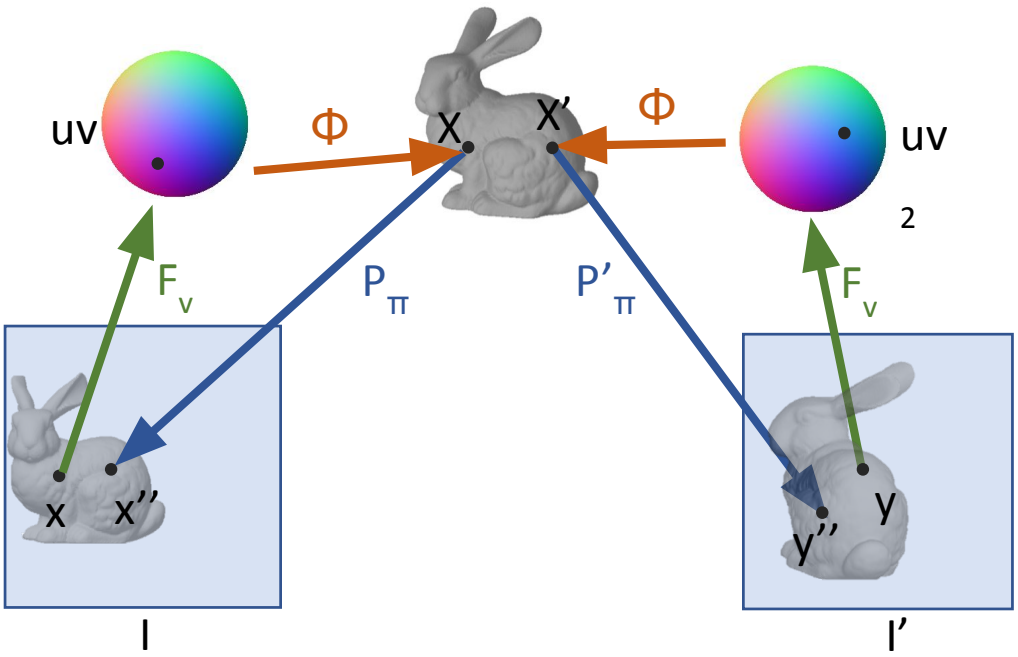


Figure 6.4: The cycle loss through an AtlasNet Decoder.

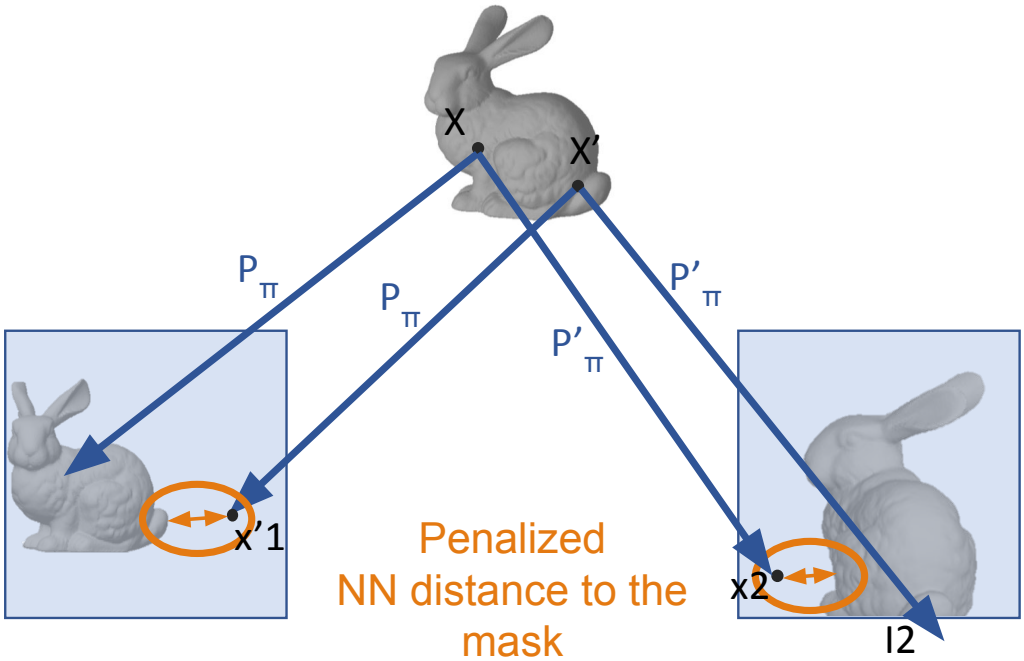


Figure 6.5: The mask loss. We use a 2D Chamfer distance to make sure that the reconstructed mesh masks fits the detected mask. Only the projections which fall outside of the mask and inside the frame are penalized.

the cycles, the UNet weights are set by training to reconstruct the UVW rendering of the initial shape. Once the pipeline is initialized, we optimize both the UNet and the decoder using $\mathcal{L}_{\text{rec}}$.

6.1.3 *Implementation details*

We optimize our full loss $\mathcal{L}_{\text{rec}}$ in stages, starting with the shape prior loss $\mathcal{L}_{\text{shape}}$. Given a dataset of 3D shapes, we render multiple views and jointly train the parameters for the multi-view shape encoder $\mathcal{E}$ and the parameterization ϕ. We follow the network architectures and two-stage training procedure described in Groueix *et al.* [GFK+18] to train a single-view model comprising the parameterization ϕ and single-view encoder $\mathcal{C}$ using loss $\mathcal{L}_{\text{shape}}$. We then jointly train the multi-view shape encoder and parameterization by initializing the parameters for ϕ and $\mathcal{C}$ from the single-view training step. We assume that the relative poses (extrinsic parameters) and camera calibration (intrinsic parameters) for the input views are known. These can be either given a priori (e.g., assuming pre-calibrated cameras and IMU-based SLAM from a smartphone), or computed from the data by traditional structure-from-motion methods [SF16]. We optimize the shape prior loss $\mathcal{L}_{\text{shape}}$ between the ground truth and predicted shapes using the Adam optimization with a learning rate of 1e-3.

Given the trained parameterization ϕ and multi-view shape encoder $\mathcal{E}$, we optimize the reconstruction loss $\mathcal{L}_{\text{rec}}$ over all views $\mathcal{V}$. First, we set the latent vector to be the output of the multi-view shape encoder applied to the input image collection $\mathbf{z}_S \leftarrow \mathcal{E}(\mathcal{V})$. Second, we initialize the $\mathcal{F}$ by making a UVW rendering of the initial mesh (each pixel contains the UVW coordinate of the corresponding point on the mesh), and by optimizing the L2 loss between this rendering and the $\mathcal{F}$ output over the $\mathcal{F}$ parameterization for 30 iterations. This allows the initial pixel-to-UVW mapping to be consistent with the initial mesh. Next, we optimize the reconstruction loss $\mathcal{L}_{\text{rec}}$ over the mapping $\mathcal{F}$ and parameterization ϕ for 220 more iterations (the latent vector $\mathbf{z}_S$ stays fixed during this optimization).

Pixel sampling method. At inference time, we randomly sample 2500 pixels per mask per iteration to prevent memory overflow.

Network architectures. In the multi-view shape, the convolutional neural network encoder $\mathcal{C}$ is a ResNet-18 [HZRS16] (yellow blocks in Figure 6.2). Latent spatial transformation network (LSTN, orange blocks in Figure 6.2) is a multi-layer perceptron with two layers. Finally, the AtlasNet decoder is a multi-layer perception with four layers.

The mapping $\mathcal{F}$, which maps the image space to the template space, is implemented with a 4-level UNet [RFB15].

Optimization parameters. We optimize the loss via Adam over mini-batches comprised of sampled pixels from each view image. We initialize the UNet with random weights and we first optimize solely over the UNet's weights for 300 iterations with learning rate 1e-4 in order to properly initialize it. Then, the UNet weights as well as the decoder's weights are jointly optimized for 2200 more epochs with a learning rate of 1e-5.

6.2 DESIGN CHOICES

6.2.1 *Template choice*

The AtlasNet [GFK+18] framework allows the use of different templates to deform. The default choice is an atlas: a set of (typically 25) planar primitives. Unfortunately, this is incompatible with our UNet-based architecture for the pixel-to-UVW mapping. Indeed, mapping an input pixel to a 3D point on the atlas involves choosing on which of the 25 primitives we want to map the pixel on. This is a hard (discrete) choice, and thus not naturally back-propagable. One could imagine making a soft choice (i.e., predicting a distribution of probability of the point presence on each primitive), but this involve important changes in the architecture, the loss and introduces uncertainty about the convergence of the optimization.

We first made experiments with a single plane as a primitive template. While this allowed to efficiently develop the pipeline, limitations soon appeared. As reported in the original AtlasNet paper, the raw performance of the single plane primitive is very limited. Moreover, since our procedure continuously optimizes the pixel to template position, the pixel closely mapped to the plane boundaries have a clipped gradient in order not to move past this border. This introduces local minima in the optimization and caused failure cases.

It was consequently natural to use a borderless primitive such as a sphere. Mapping a pixel to this template involves no choice and no gradient clipping.

6.2.2 *Using the UVW parameterization*

When using the sphere as a template, we can use the following classical UV-mapping:

$$u = 0.5 + \frac{arctan2(d_x, d_z)}{2\Pi} \tag{6.4}$$

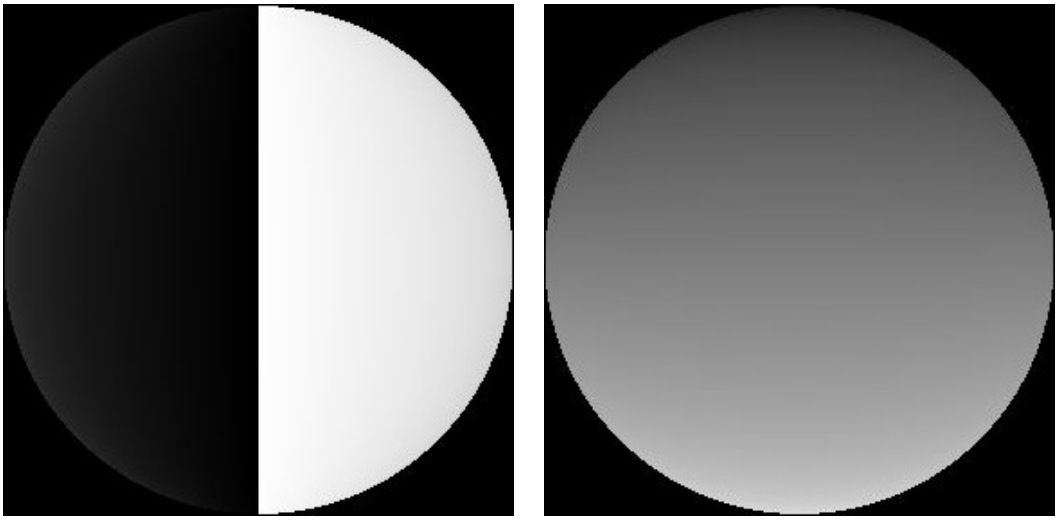
$$v = 0.5 - \frac{arcsin(d_y)}{Pi} \tag{6.5}$$

where (d_x, d_y, d_z) represents any point on the unit sphere. This parameterization is similar to the latitude/longitude system on earth and as such, it presents a discontinuity line (akin to the international date line) on the U coordinate as represented on Figure 6.6.

As our algorithm maps pixels to a position on the sphere, this is an issue, because the points on the sphere can't cross the discontinuity line during the optimization. We therefore use 3 output channels instead of 2 on our UNet and use a simple normalization to obtain a point on the sphere. While this parameterization does not involve any discontinuity, the risk is for the norm of the output channel to either shrink or explode. In practice, we did not notice such phenomenon.

6.2.3 *Optimized parameters*

The authors of [LWR+19] only used the latent vector as an optimization parameter. This assumes that the latent space build at initialization time (see subsection 6.1.1) is expressive enough. In order to check this assumption, we took a trained decoder, generated a latent model thanks to the first input image of a model, and optimized the decoder output against the ground truth with the Chamfer distance. We tried optimizing over the latent vector



U coordinate V coordinate

Figure 6.6: Rendering of the UV coordinates (Black: 0, White: 1) for a unit sphere

on the first hand, and over the decoder's weights on the other hand (see Figure 6.7). It turned out that recovering small details and achieving a very high F-score was not possible when optimizing only over the latent vector, hence our choice to optimize over the decoder's weights.

6.2.4 *Positional encoding*

The AtlasNet decoder is a multi-layer perceptron modeling a deformation of a surface. Recently, the positional encoding method has been introduced to help this kind of network to model finer details [MST+20]. This method consists in providing as input to the network a frequency decomposition of the coordinates. In our case, while this improves the loss during training, it also decorelates the points on the template surface. As we can see on Figure 6.8, even if the point cloud looks correct, the wireframe of the mesh shows a lot of foldings and self-intersections. In our case we therefore concluded that it was better to preserve the ability of neighbour points to keep being correlated during the update of the decoder weights.

6.2.5 *Regularization*

Foldings and self-intersections are bad in general: first, they usually cause additional troubles in downstream applications; second, our optimization process is based on the assumption that we can continuously deform the template to the target shape. When reaching a deformation state that involves self-intersection, this generates discontinuities in the pixel to UV mapping which make the optimization even harder. In order to avoid these issues, we have tried introducing regularization losses for the Chamfer loss:

- Edge regularization: L2 penalization of the distance between neighbour vertices.
- Normal consistency: cosine distance between the normals of each pair of points which share an edge.
- Laplacian regularization: we add the Laplacian of the transformation to encourage neighbour points on the sphere to also be neighbours on the target surface.

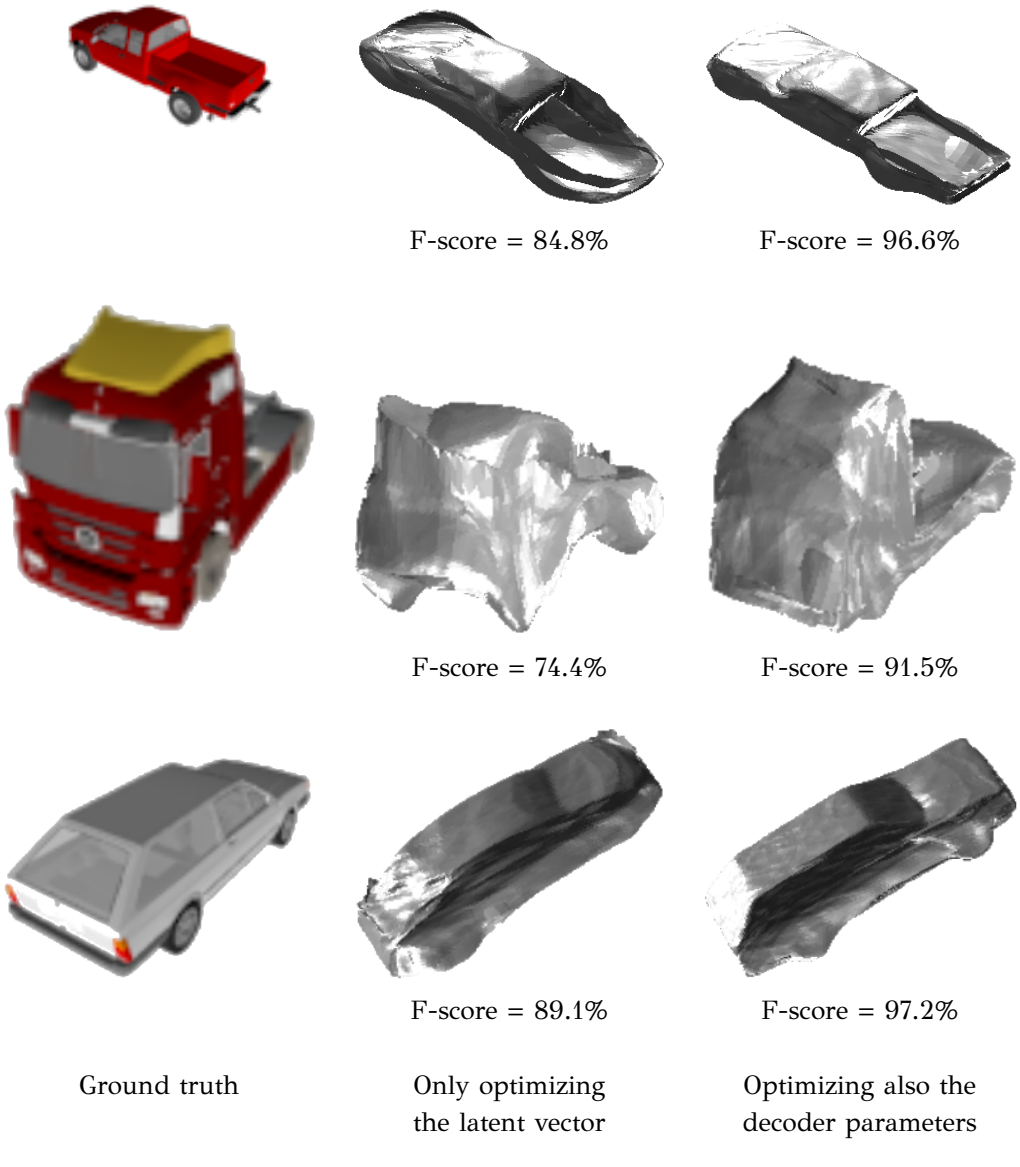
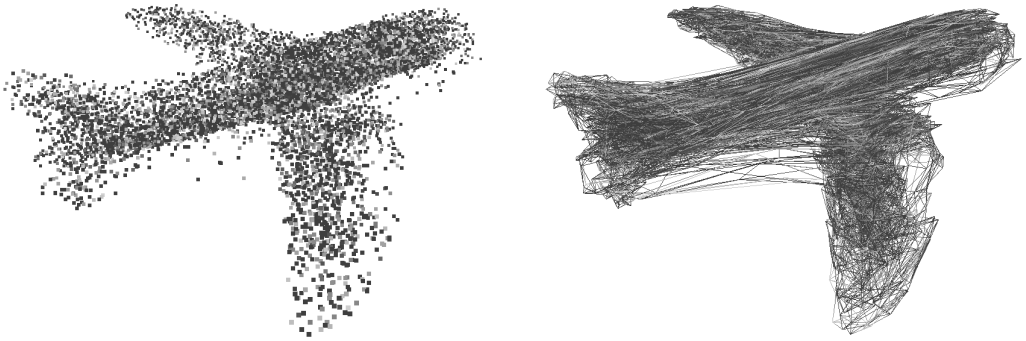


Figure 6.7: Impact of the choice of the parameters used for optimization



Input point cloud Wireframe of the mesh

Figure 6.8: Failed initialization of a plane shape with positional encoding.

This approach has several drawbacks: first, it is hard to find the weights for the regularization terms; second, these regularization terms tend to prevent the model from learning details in the priors. In the end, we obtained better qualitative and quantitative results by introducing the gnomonic distance as a learning loss for the optimization step (see 6.1.1).

7

Results

◂ Chapter 6
Chapter 8 ▸

We report qualitative and quantitative performance compared to the baselines. We consider the task of reconstructing a depicted shape given a set of views with known camera and object segmentation mask for each view.

7.1 DATASET

For our controlled setup, we evaluated on the publicly available ShapeNetCOCO dataset [SRV+19], which has rendered views for three shape categories ("airplane", "car", "chair") from ShapeNet [CFG+15]. We used the published training and validation splits containing 11,469 and 2,868 shape instances, respectively. We evaluated on 100 random shape instances belonging to the validation split for each shape category (300 instances in total). The dataset includes for most shape instances 20 rendered views with widely varying camera distances from the target shape, with the rendered views composited on single still images from the COCO dataset [LMB+14]. Note that the rendered views in the ShapeNetCOCO dataset are more challenging than rendered views from Choy *et al.* [CXG+16] due to the composited natural image background and larger distances from the camera's viewpoint to the shape. We also found that the camera calibration was not accurate for the Choy *et al.* [CXG+16] rendered views, which limits the accuracy of the reconstructions. Note that the ShapeNetCOCO dataset has world origin at the shape instances' centroid. Finally, we use the ground truth segmentation masks provided with the dataset.

7.2 EVALUATION CRITERIA

We evaluate the quality of the output reconstructions by reporting the F1-score (F1) and symmetric Chamfer distance (CD). Following Wang *et al.* [WZLF19], we apply their method for uniformly sampling points on the surface of the ground-truth shapes, yielding 10k points per shape. Following Wen *et al.* [SRV+19], we sample 10k points from the output shapes. Given a threshold τ which is 1% of the ground-truth mesh bounding box diagonal, we compute the fraction of output points that are within τ of a ground-truth point (precision) and vice versa (recall). We follow Knapitsch *et al.* [KPZK17] and report the F1-score, which is the harmonic mean of precision and recall. We follow Sridhar *et al.* [SRV+19] and report squared-symmetric Chamfer distance (multiplied by 100). Note that these metrics are point-based, not meshed-based (which our network outputs), in order to be comparable with XNOCS [SRV+19].

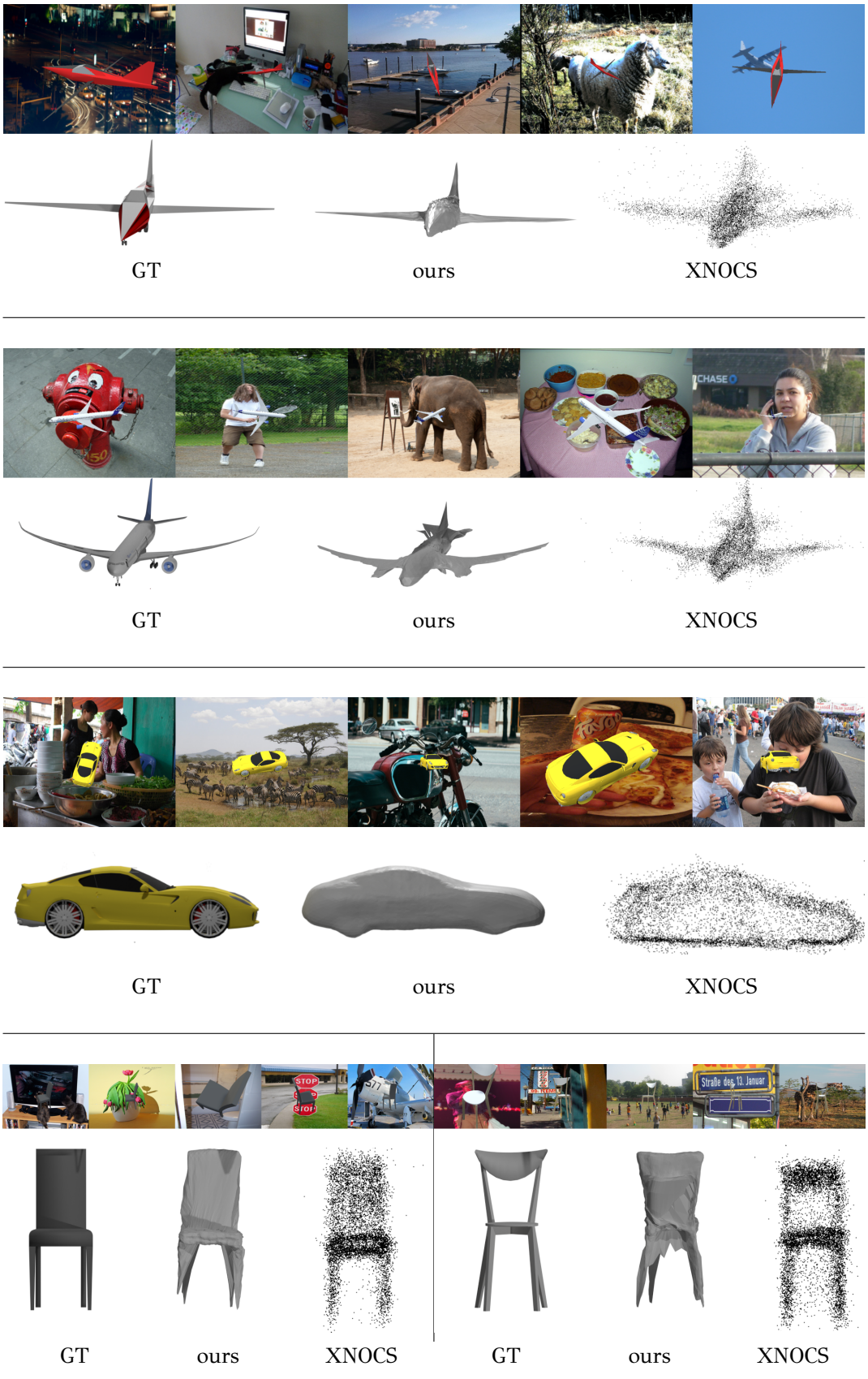


Figure 7.1: **Qualitative results.** For each sub-figure: Input (top), Ground truth (bottom left), our reconstruction mesh (bottom middle) and XNOCS output points (bottom right).

7.3 EXPERIMENTAL RESULTS

7.3.1 *Optimization process*

We represent on Figure 7.2 what happens during the optimization process of $\mathcal{L}_{\text{rec}}$ on an example shape from the airplane category. The step count starts after the UNet has been initialized (see 6.1.3 for the detailed procedure). As expected, we notice that the mesh gets closer to the ground truth as optimization evolves. Moreover, the wings on the initial shape continuously move and deform towards their final position, which tends to show that the priors obtained at initialization are correctly exploited.

7.3.2 *Baselines*

We compare against XNOCS [SRV+19] with multi-view aggregation of 5 images and with separate networks trained for each shape category. To be comparable, we train our approach by randomly sampling (only) five images for the input views.

We also compare against a fixed single-template baseline. For this baseline, we randomly choose a template shape for each shape class and compare the chosen template against the validation set. (The shapes are oriented and scaled consistently across the dataset.) We report the average across ten randomly selected templates for each category. For a particular selected template, this baseline is an upper bound for CSM [KGT19].

7.3.3 *Results compared to baselines*

	Airplane		Car		Chair		Avg.	
Method	F1↑	CD↓	F1↑	CD↓	F1↑	CD↓	F1↑	CD↓
Single template	36.7	0.42	23.3	0.53	13.3	1.04	24.4	0.66
XNOCS [SRV+19]	**62.3**	0.08	33.2	0.19	**27.2**	**0.20**	40.9	0.16
Our full model	56.3	**0.04**	**41.7**	**0.07**	25.5	0.28	**41.2**	**0.13**

Table 7.1: **Multi-view shape reconstruction on the ShapeNet-COCO [SRV+19] validation set.** We report F1-score (F1) and squared-symmetric 100xChamfer distance (CD), averaged over each class.

We train our method (*Our full model*) for each shape category separately. We report results in Table 7.1. Averaging over the three shape categories, our approach outperforms the baselines for both the F1-score and the Chamfer distance criteria although we solve a harder task, i.e., meshing rather than point generation. Note that the template baseline performs surprisingly well. In fact, as noted in Tatarchenko *et al.* [TRR+19], there is large overlap of the shapes in the train/val splits. We show qualitative results in Figure 7.1, 7.3, 7.4, 7.5 and 7.6. On Figure 7.1, we also show the whole sequence of input images from which the model was created.

The point clouds generated by XNOCS are very noisy. We observe that creating a mesh from the output point clouds may be difficult, while our approach produces meshes directly.

Our method is able to recover specific shape details of the reconstructed shape (see the reactors on Figure 7.1 and 7.3). In the airplane (Figure 7.3) and car (Figure 7.4) categories, we notice that many models share similar shapes with very slight differences (e.g., car trunk shape, position of the airplane reactors). Our optimization step allows to more precisely recover these details. We notice that the chair category is harder to properly reconstruct. This is due to the fact that the transformation between a sphere and a chair requires additional distortion compared to the other categories, e.g., to create

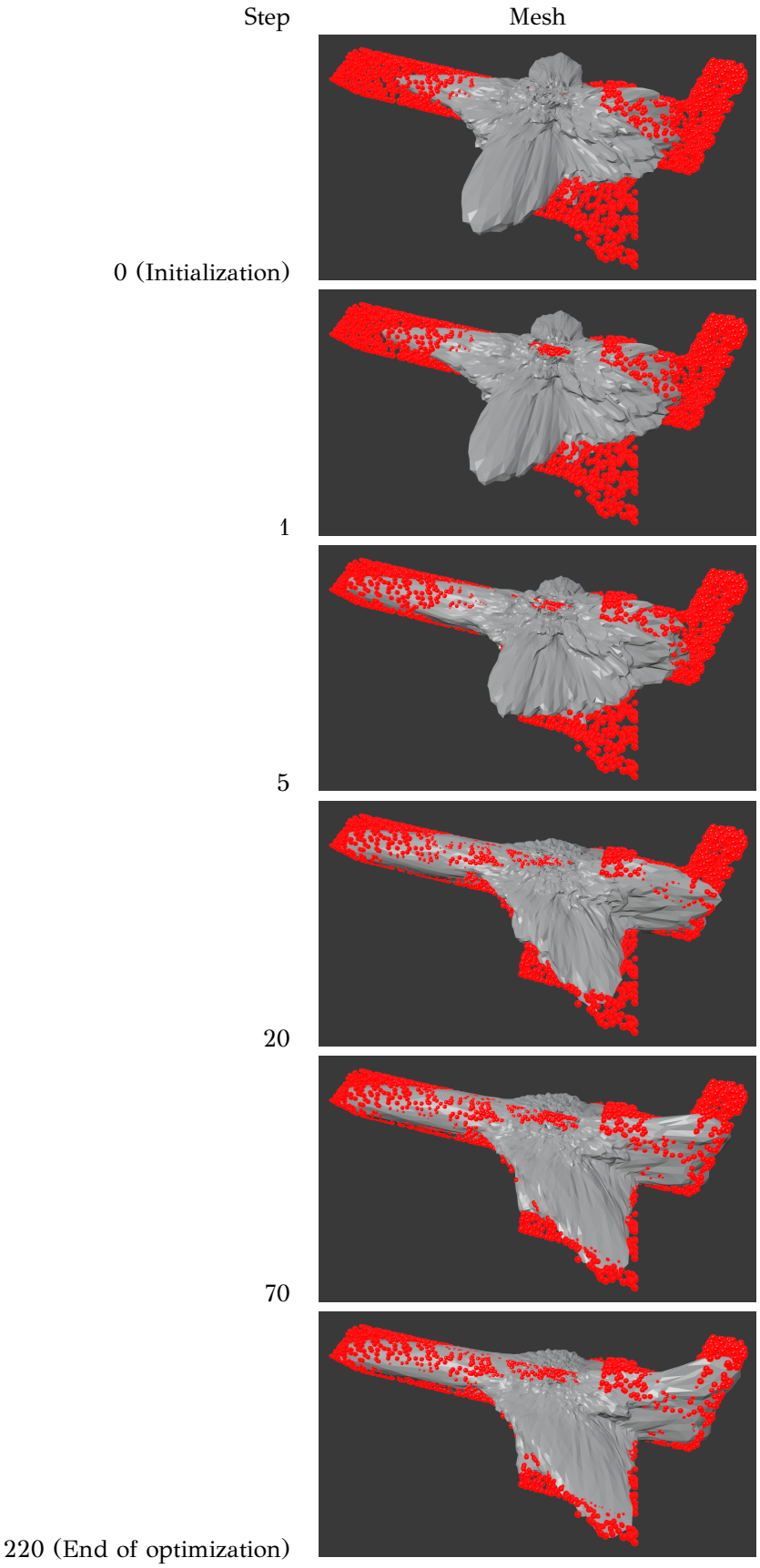
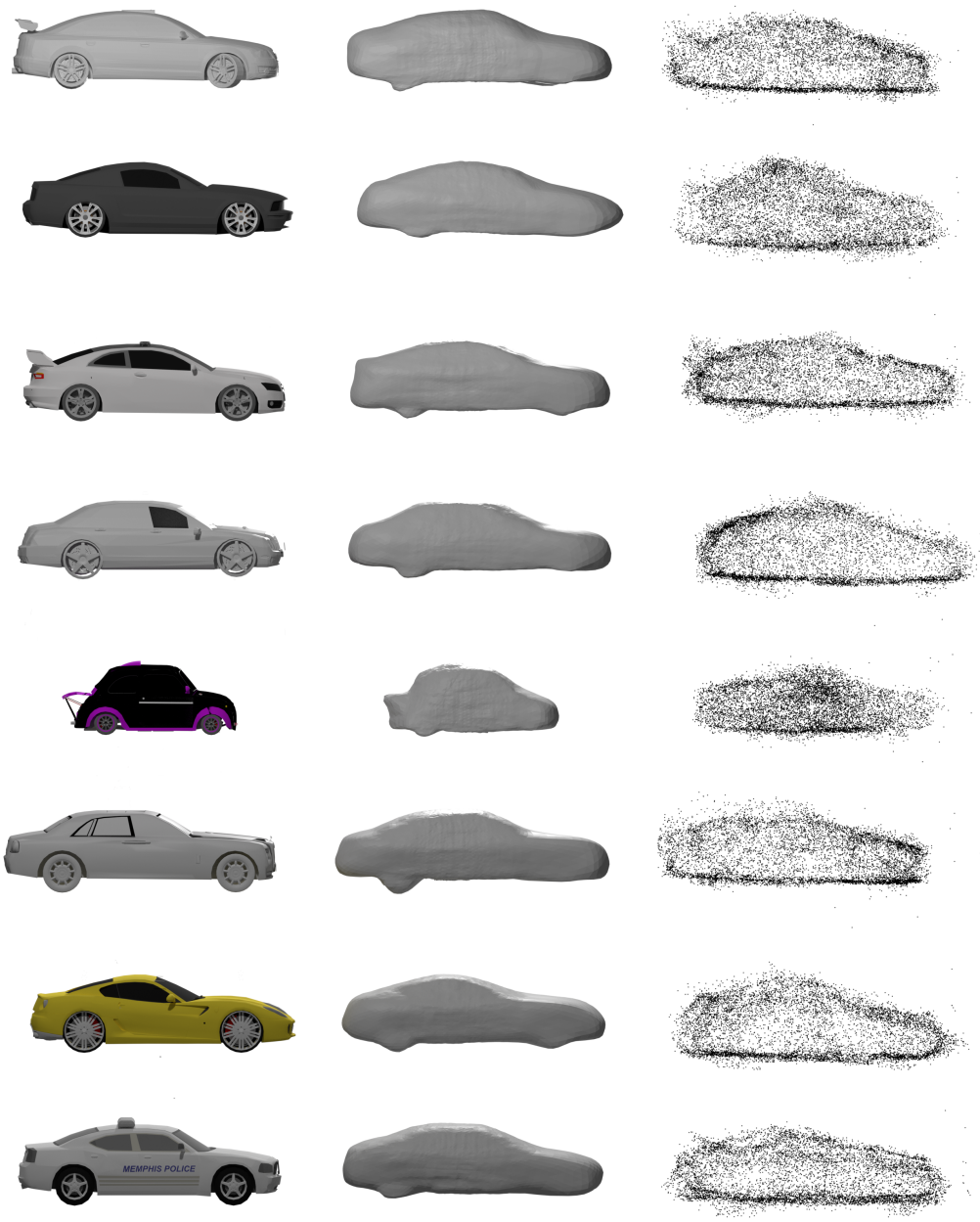


Figure 7.2: Representation of the optimization process. Grey: the mesh being optimized. Red: ground truth as a point cloud

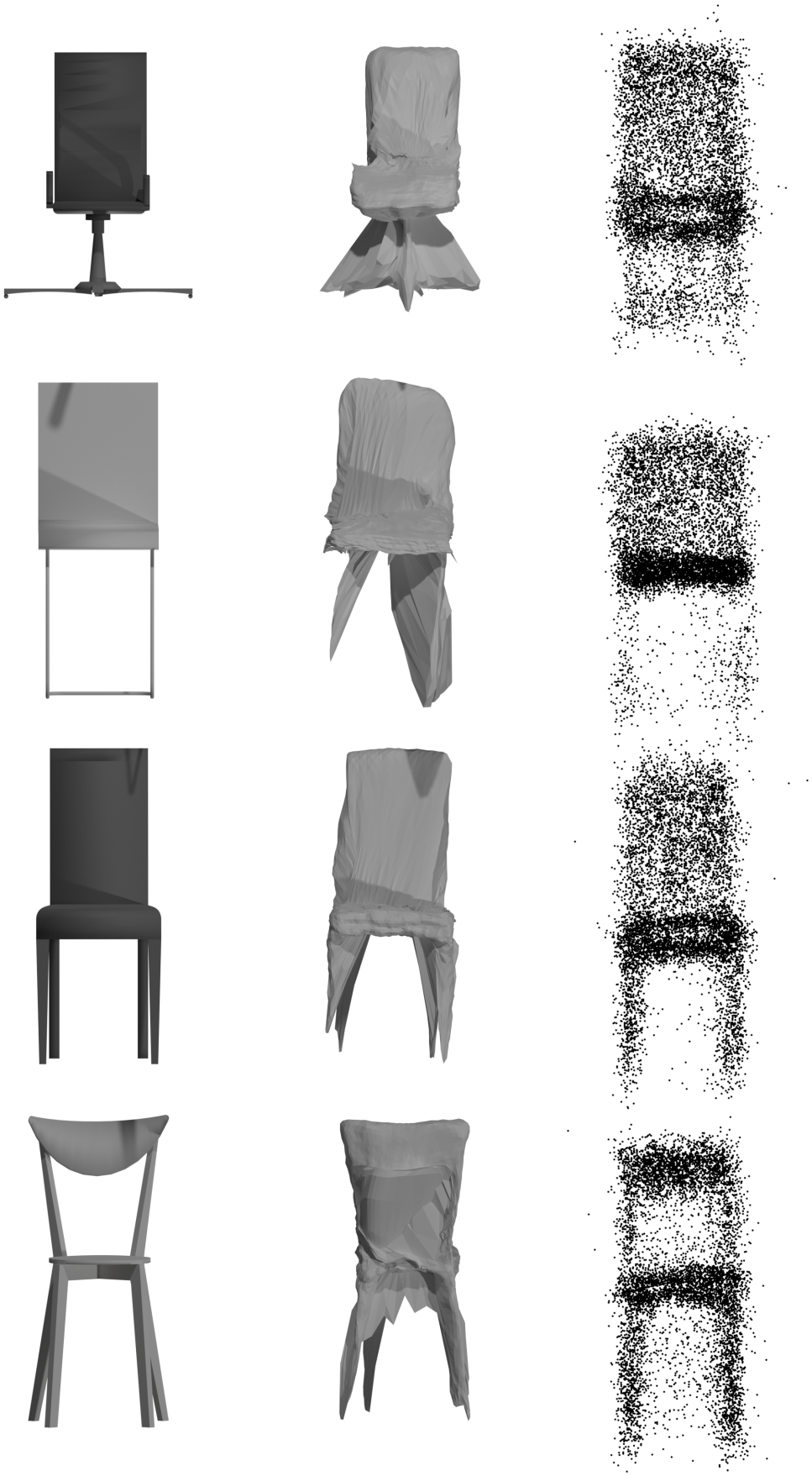
Ground truth Ours XNOCS [SRV+19]

Figure 7.3: Qualitative results on the airplane category.



Ground truth Ours XNOCS [SRV+19]

Figure 7.4: Qualitative results on the car category.



Ground truth Ours XNOCS [SRV+19]

Figure 7.5: Qualitative results on the chair category.

Ground truth Ours XNOCS [SRV+19]

Figure 7.6: Qualitative results on the chair category.

thin legs. Moreover, the models in this category include non genus 0 objects, which can't perfectly be reconstructed by a sphere. Therefore, optimization is harder and can more easily fall into local minima.

7.4 ABLATION STUDY

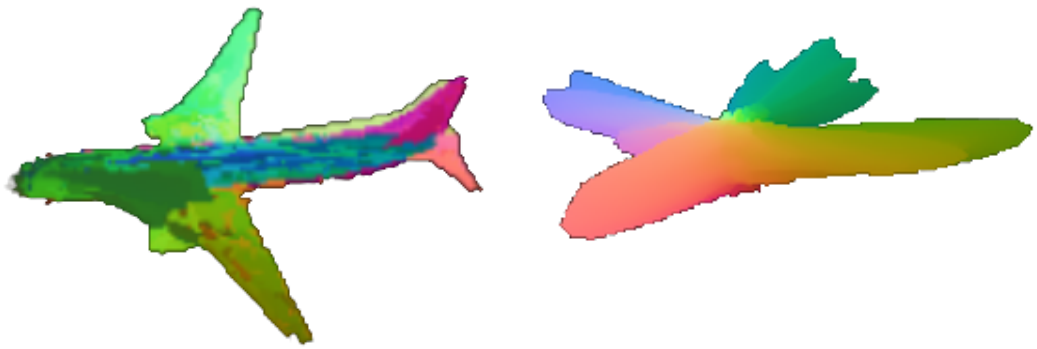
The results of an ablation study are presented in Table 7.2. We consider the method with or without the optimization step. We also consider using a Chamfer loss rather than the gnomonic loss.

7.4.1 *Impact of the optimization stage*

We see that the multi-view encoder performs reasonably well, in particular on planes and cars, but it is not sufficient to reach the state-of-the-art. It is however a good prior in the full setting: the model with optimization performs better than the multi-view encoder alone on every categories and metrics (from 16 to 24 points), given as input the estimated shape produced at the initialization step.

7.4.2 *Chamfer loss vs gnomonic loss*

We compare our $\mathcal{L}_{\text{shape}}$ loss defined with a gnomonic projection of the sphere to the Chamfer distance loss, that uses closest points to establish correspondences [GFK+18], which often leads to non-smooth and non-injective mappings with local self-intersections and foldovers. We first optimize the multi-view encoder and the AltlasNet decoder (*MV encoder*) and observe that the two models obtain similar scores. It seems that both losses are suitable for the task. However, taking the next step (*Full model*), we can observe that using the gnomonic projection leads to a higher F1-score. Our interpretation is that this is due the non-smooth surface reconstructed with the Chamfer loss. We show in Figure 7.7 two meshes reconstructed after MV-encoder training, with Chamfer distance (left) and gnomonic projection (right). The first model seems to better fit the plane shape, but has a lot of self-intersections (each color discontinuity) while the second model is very smooth. This smoothness is, in practice, a better starting point for the reconstruction optimization.



Chamfer loss Gnomonic mapping loss

Figure 7.7: **Visualization** of learned surface parameterization depending on the loss used. Discontinuities in colors indicate self-intersections.

Method	Loss	Airpl. F1↑	Car F1↑	Chair F1↑	Avg. F1↑
AtlasNet†	Chamfer	38.6	20.6	5.8	21.7
MV encoder only	Chamfer	33.0	14.6	14.9	20.8
MV encoder only	Gnomonic	32.9	24.2	9.2	22.1
MV encoder + optimization	Chamfer	49.7	34.2	24.3	36.1
MV encoder + optimization	Gnomonic	**56.3**	**41.7**	**25.5**	**41.2**

Table 7.2: **Ablation study.** Comparison of multi-view encoder and full model for Chamfer and gnomonic loss. †: AtlasNet (single view) makes inference in canonical coordinates as opposed to camera coordinates for our method.

7.5 DISCUSSION AND LIMITATIONS

While our approach outperforms the baselines, we found several limitations. First, while the gnomonic projection outperforms the Chamfer loss, there are still remaining reconstruction artifacts when mapping from a sphere to a target shape to due the lack of bijectivity; most shapes are not star-shaped volumes, even simple cars. Second, the spherical template shape restricts the outputs to the set of genus-zero shapes, which does not allow us to reconstruct well shapes with arbitrary topologies, which is the case of many chairs (cf. Figure 7.1). In fact, both issues are present in chairs which are harder to reconstruct for XNOCS too.

7.6 CONCLUSION

To wind up, we have presented a method to learn a dense UVW mapping of a parameterizable shape template. We introduce a multi-cycle loss to resolve ambiguities inherent in single image shape and UVW prediction, and show that our method outperforms a recent state-of-the-art approach. In addition, our method is able to reconstruct a mesh, rather than a point cloud, which makes it more useful for downstream 3D applications. An interesting research direction is to use UVWs predicted from multiple images along with traditional projection-based techniques to optimize for the final mesh textures.

Transition

In this part, we studied how machine learning can enhance the reconstruction task by incorporating meaningful priors into the reconstruction process. Thanks to our method, we are able to generate meshes out of low resolution calibrated images representing small objects with little to no texture. In the next part, instead of pictures, we are interested in point clouds obtained by LIDAR sensors, and we aim at inferring semantics in addition to a volume reconstruction. While the template deformation approach is promising, there are inherent difficulties to adapt it in the context of buildings: the genus of a building can be high; the deformation from a simple template is greater than in the case of objects and this 3D representation does not naturally handle multiple labels. In part II, we used a partition of the 3D space which, as we will see, more easily handles multiple labels. In the next part, we strive to use both machine learning and a partition of space to achieve both 3D volume reconstruction and semantics inference by focusing on the main obstacle: the lack of relevant data to learn from for this task.

Part IV

VASAD: A VOLUME AND SEMANTIC DATASET FOR BUILDING RECONSTRUCTION FROM POINT CLOUDS

8

Introduction

◂ Chapter 7
Chapter 9 ▸

Synopsis. 3D Scene Reconstruction has important applications in industrial contexts, and the advent of Building Information Models (BIM) has made it possible to drastically improve the operational costs of existing buildings. While the community has mostly focused on surface reconstruction or semantic segmentation as separate problems, the joint reconstruction of both volumes and semantics has little been discussed, mostly due to the lack of large scale volume datasets with semantic annotations.

In this work, we introduce a new dataset called VASAD, for Volume And Semantic Architectural Dataset. It is composed of 6 models, with full volume description and semantic labels. It represents approximately 62,000 m2 of building floors, making it large enough for the development and evaluation of learning-based approaches.

Additionally, we propose several methods to jointly reconstruct both geometry and semantics and evaluate on the test set of the dataset. We show that the proposed dataset is challenging enough to stimulate research.

8.1 3D reconstruction for buildings

Monitoring the life cycle of existing buildings is of great interest for the building industry. In this context, being able to reconstruct a *Building Information Model* (i.e., a digital representation) of the current state of a building is required [TB15]. It implies reconstructing both the volume geometry and the semantics of the building.

In industrial contexts, the first step to perform model reconstruction is often to capture a 3D scan using LIDAR technology. This type of sensors allow precise depth measurements and therefore provide high-quality geometric features. Moreover, normals are efficiently estimated on such raw data [BM12]. However, LIDAR sensors suffer from inherent flaws: there often are missing measurements due to mirrors or surfaces which are transparent to the LIDAR light frequency. Moreover, measurements are usually taken from a fixed point of view which generates a non uniform sampling with respect to the underlying surface. Finally, and as for every sensors, some surfaces may be completely missing from the input data because it was not seen from any point of view. Reconstructing building parts from partial data involves the use of strong priors. This motivates the use of machine learning based approaches to learn these priors directly from existing data.

In this context, many approaches have been developed for 3D semantic segmentation for point clouds [BPM20, QSMG17, TQD+19]. On the other hand, advances have been made for surface reconstruction [CXG+16, GFK+18, MON+19, PFS+19] and more recently for large scene reconstruction [PNM+20]. However few methods tackle the semantic volume reconstruction [SYZ+17] task and when it comes to full building reconstruction, related work is even scarcer [MvAB+20].

One of the main reasons is the lack of suitable datasets. The existing datasets usually tackle one task or the other, and datasets containing both geometric and semantic infor-

mation are focused on objects (i.e., furniture) rather than building components. Therefore, these components are usually presented as open surfaces instead of closed instance volumes which is required for digital building reconstruction.

This work intends to help filling the gap toward full digital mockup reconstruction both in terms of dataset and methodology. We introduce a new dataset called VASAD, for Volume And Semantic Architectural Dataset. It is composed of 6 full building models, with volume description and semantic labels of each building component. It represents approximately 62,000 m^2 of building floors, making it large enough for the development and evaluation of learning-based approaches. The objective of this synthetic dataset is to leverage machine learning techniques for architectural reconstruction, regarding both geometry and semantics.

Second, we present a deep neural network for joint semantic and geometric reconstruction. It is built over of a semantic feature extractor followed by a dense voxel-reconstruction network. Quantitative evaluations on VASAD show that it outperforms the state-of-the-art reconstruction method [PNM+20] that we also modified for semantic reconstruction purposes.

8.2 Related work

Surface reconstruction Reconstructing surfaces has been part of the earliest tasks that the computer vision community has tried to address. Assuming that the point clouds are directly sampled on the underlying surface, efficient methods [BMR+99] allow mesh reconstruction. However, the point clouds are usually noisy and a smoothness prior can help handling outliers [KBH06]. In the context of LIDAR scans in man-made environments, piecewise-planar reconstruction provides an idealized model of the scene [BdLGM14, NW17] or an hybrid idealized/free-form model [LA13]. To avoid using hand-crafted priors, recent methods tend to leverage big datasets of 3D objects [CFG+15] and scenes [GLU12, BGM+19, HSL+17, ASZS17, RDG18] to directly learn meaningful priors. Early works use a voxel-based representation [CXG+16] to apply similar neural network structures as in 2D. More recent works have enabled learned 3D reconstruction on meshes [GFK+18, WZL+18] and using implicit functions [PFS+19, MON+19, PNM+20]. Another research direction [CTZ20], leverages binary space partitioning, which generates simple piecewise-planar closed meshes that are similar to the output of the computer-aided design editors typically used to build BIM models. These methods usually involve an encoder/decoder neural network architecture, and make an important use of the PointNet [QSMG17] point cloud encoder.

Semantic segmentation of point clouds Early methods [NL13] formulate the point cloud segmentation problem as a region-growing [BJ88] problem, a model-fitting problem [SWK07], a clustering problem [BL08] or a graph-cut [GF09] optimization. These different formulations allow the user to hand-craft priors they have on their data: [SWK07] is efficient to recover piecewise-planar regions of a point cloud, and it is robust to noise; [GF09] allows to use local properties as descriptors and to use pairwise relationships as a propagation prior. While these methods can be very effective in particular cases, they fail at handling classes which involve complex priors.

Again, recent methods leverage deep learning to directly learn complex priors on massively available data. Pioneering work in this direction includes PointNet [QSMG17], a simple multi-layer perceptron which leverages the pooling operator to approximate any

continuous order-invariant function on a set of point coordinates with features. Follow-up work [QYSG17] focused on building a local analysis to improve performance, and progressively, proposals have been made to generalize the convolution operator to point clouds [LBS+18, TQD+19, WQF19, BPM20], pushing further the number of handled classes, the complexity of the detected classes, and the size of the handled point clouds. While these methods rely almost purely on point geometry, RGB-D sensors allow to jointly leverage photometric and geometric information to enhance the segmentation performance [DN18].

Joint reconstruction and semantic segmentation The intuition that both tasks of reconstruction and semantic segmentation can leverage each other becomes more and more popular. Pioneering work in this direction [SYZ+17] simultaneously complete and label voxels obtained from a single depth image with full supervision. Recent work try to directly leverage the synergy between both tasks in a single-view setup by affinity learning [ZCX+19], by supervising the reconstruction task thanks to a pre-trained segmentation network [GHL+20], by introducing a content consistency constraint [CLLW19] or by independently estimating the geometry on the segmented classes and adding a merging step [AAB19, WZW+20].

Implicit representation is a major tool to achieve both tasks, because it allows to formulate the problem as an optimization task over the whole 3D space whose data term can be hand-crafted [HP16] or learned [CSO+18]. In this context, some research projects began to get interested in buildings [DRB+18], but essentially stayed at the scale of a room and its furniture, and the lack of data has prevented researchers to address the reconstruction of the full structure of a building with semantics.

Need for a joint semantic/volume dataset Existing datasets mainly focus on a single task which is either surface reconstruction or semantic segmentation.

Semantic segmentation is usually handled at point level on various point cloud types such as car-embedded LIDARs, e.g., SemanticKITTI [GLU12, BGM+19] or nuScenes [CBL+19], outdoor LIDAR, e.g., Semantic3D [HSL+17] or NPM3D [RDG18], or indoor scenes, e.g., S3DIS [ASZS17], MatterPort3D [CDF+17] or ScanNet [DCS+17].

Surface reconstruction aims at reconstructing a 2-manifold based on partial view of the scene (either point cloud or images). Multiple datasets exist for this task, including ShapeNet [CFG+15], DFaust [BRPMB17] or MatterPort3D [CDF+17]. In the most general case, this task does not impose to reconstruct closed surfaces. However, several approaches, including implicit function estimation, tackle the task by learning a dense function over the whole 3D space which labels any 3D point as inside or outside; this function is learned using closed shapes, (volumes) as a supervision.

Volume datasets are often obtained after a heavy pre-processing step for closing shapes. In [MON+19, PNM+20], the shapes of ShapeNet are closed using the marching cubes algorithm [LC87].

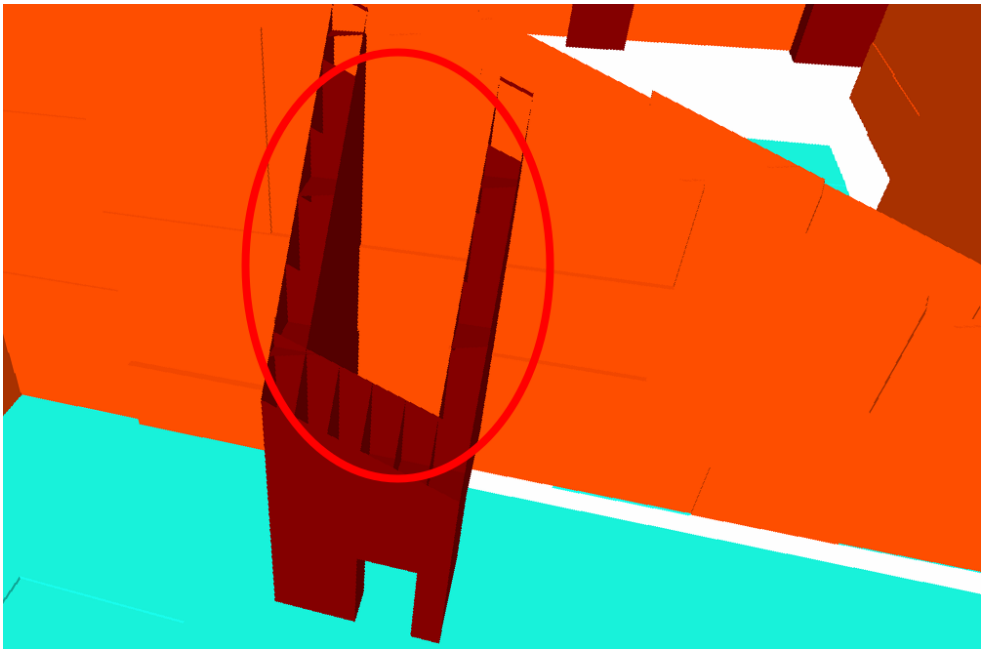
Semantic and volume approaches are very rare due to the lack of datasets containing both information.

In [PNM+20], the authors propose a synthetic dataset, SyntheticRooms, constructed with ShapeNet objects and planes with small thickness for walls and floor. Even if not used in [PNM+20], semantic information can be retrieved and used for learning. However, the dataset is object-oriented and is not realistic for the reconstruction of the full volume of a building: the building components are reduced to floor and walls with a fixed thickness.

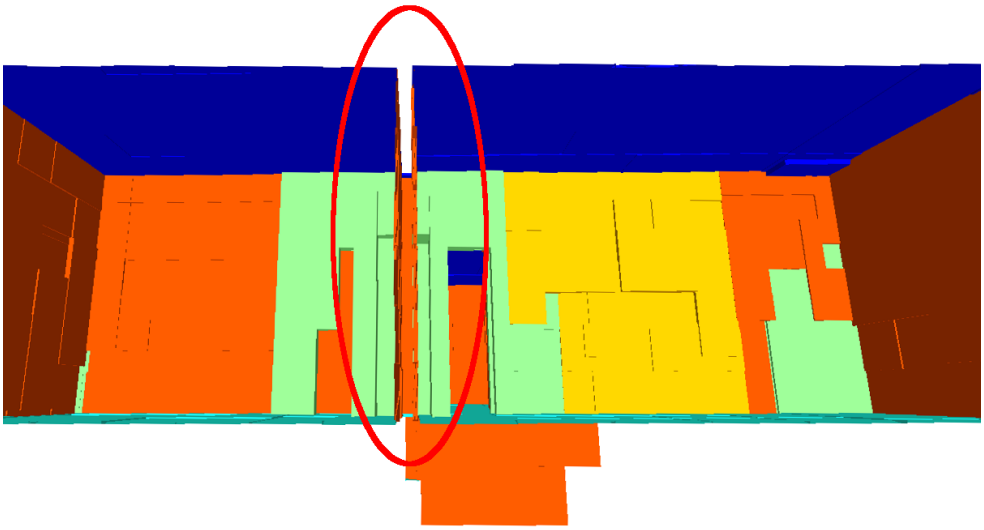
S3DIS [ASZS17] contains volume information encoded as a set of 3D boxes with

semantics. The classes are more numerous than in SyntheticRooms, including windows and doors, but the volume representation is mostly furniture focused i.e., it is not accurate at all for the building structure. For example, volumes are built for each room independently, thus, floor and walls are given arbitrary thicknesses; moreover, some holes (missing labels) make the date incomplete (see Figure 8.1).

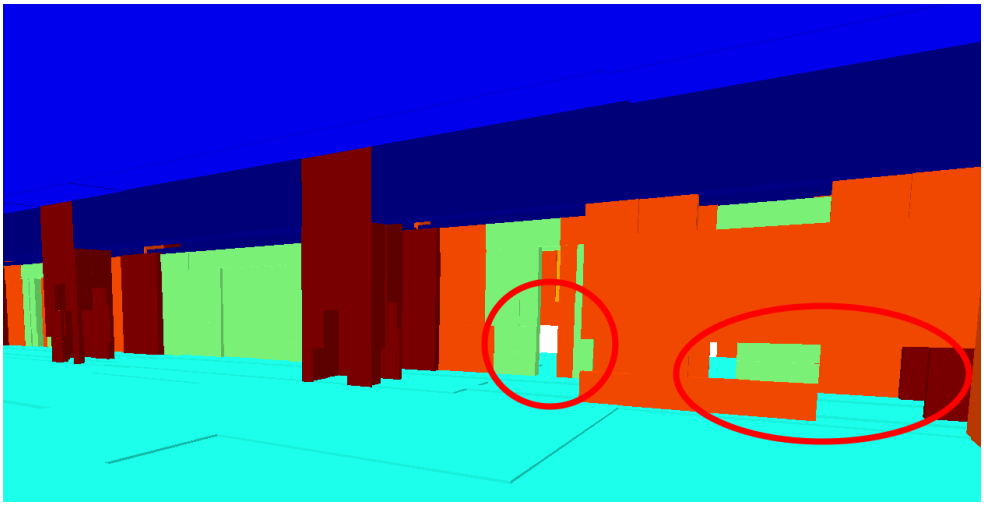
To address the lack of data for volume and semantic reconstruction, we present a new dataset.



Empty column



Lack of thickness in a wall



Holes in the walls

Figure 8.1: A few shortcomings of the volume annotations in S3DIS [ASZS17].

9

Dataset

◂ Chapter 8
Chapter 10 ▸

9.1 BUILDING INFORMATION MODELS

In recent years, the construction industry has developed a new type of digital model called *Building information model* (BIM) for better building conception, maintenance and modification. While 3D geometric models have essentially been used for building conception, BIMs include a richer information: the semantic class of each building component (e.g., wall, ceiling, floor...), as well as other technical information. While this semantic data is usually created during the design of a new building, it is especially useful for existing buildings. However, manually creating such semantized models is costly. The main objective of our dataset is to show that it is possible to use BIM models available in the wild to learn semantic and volume priors to perform reconstruction from point clouds which are typically obtained from LIDAR scans.

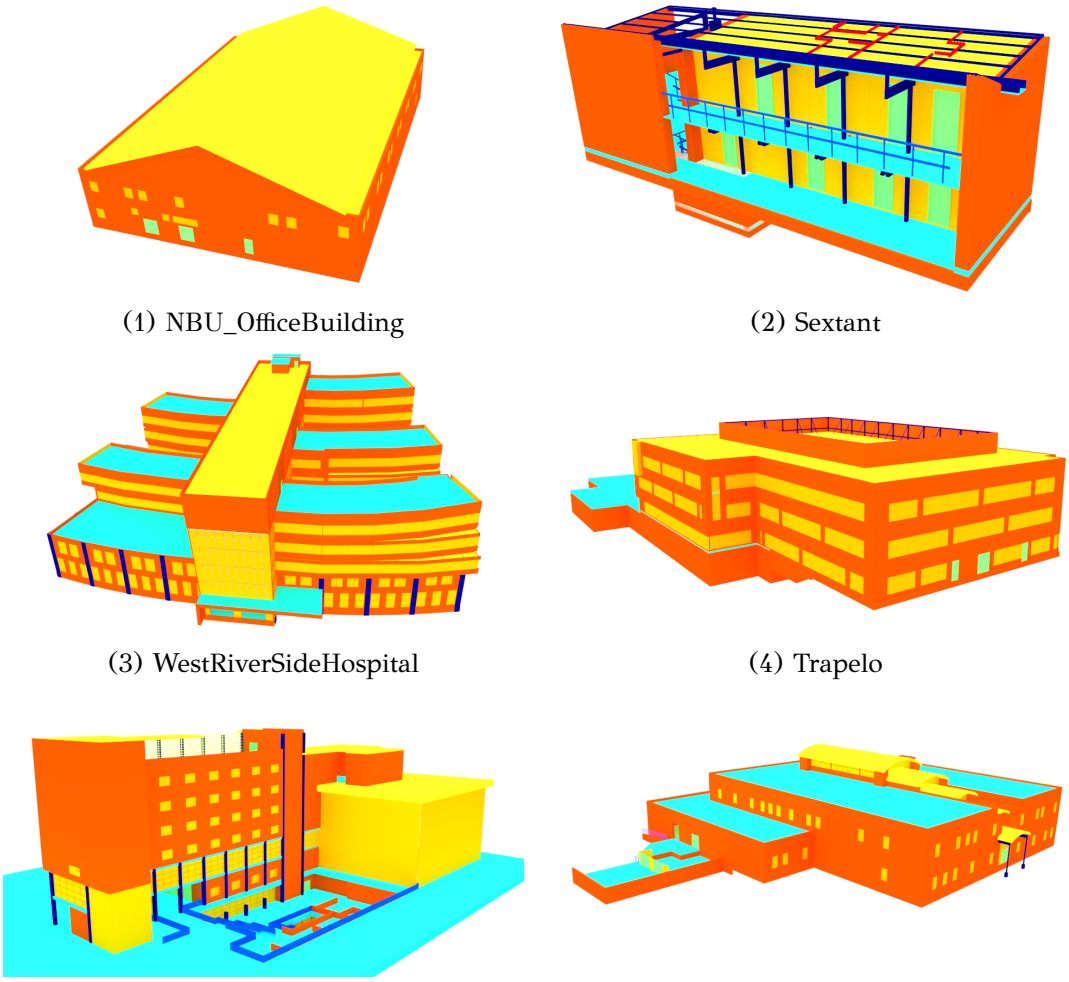
9.2 PRESENTATION OF THE DATASET

As opposed to previous datasets, our primary objective is to build a realistic database for building component reconstruction that is focused on the building's entire structure instead of furniture.

VASAD is derived from six freely available BIMs covering more than 62,000 m^2. The scenes vary from a large scale hospital to a small residential building. The six buildings are presented in Figure 9.1 (a) and (b), with interior views on Artwork 2, Artwork 3, Artwork 4, Artwork 5, Artwork 6 and Artwork 7. The semantic classes and their color representations are listed on Table 9.1. These classes have been obtained by filtering the components names in the original BIM models. We strove to build classes which are relevant to properly model arbitrary building structures, and to avoid the overlap between the classes, though some of them can be challenging to tell apart (e.g., partition vs bearing wall, slab vs ceiling, partition vs pillar). As opposed to S3DIS [ASZS17] which covers 6,000 m^2, VASAD contains the full structure information, e.g., the volumes represent the full structural components of the building and are not limited to a few millimeters from the surface.

Beam	Bearing wall	Ceiling	Door	Floor	Partition
Pillar	Railing	Roof	Stair	Window	

Table 9.1: Semantic classes in VASAD along with their color scheme.



(1) NBU_OfficeBuilding (2) Sextant

(3) WestRiverSideHospital (4) Trapelo

(5) OTC-ConferenceCenter (6) NBU_MedicalClinic

Figure 9.1: Dataset

(a) Visualization of the dataset

Id	Type	Name	Split	Surface (m^2)	#Objects
1	Office	NBU_OfficeBuilding	Train	3700	1241
2	Residential (villa)	Sextant	Train	228	1444
3	Hospital	WestRiverSideHospital	Train	29600	23661
4	Office	Trapelo	Train	5800	3657
5	Conference Center	OTC-ConferenceCenter	Train	18400	5500
6	Medical Clinic	NBU_MedicalClinic	Test	4500	3094

(b) Buildings composing our dataset

9.3 3D REPRESENTATION

Given an oracle which tells the class of an arbitrary 3D point on the scene, it is possible to generate a wide variety of 3D representations, e.g., meshes or voxels. In order to build such an oracle, an important condition is that the input shapes are closed. It is practice almost the case in VASAD, though some construction errors still remain (e.g., due to their complex structure, some stairs railing are not always closed meshes without self-intersections).

Our oracle is built on the idea that any ray starting from a point inside a given shape intersects its surface an odd number of times. In order to introduce some robustness with respect to the errors in the input meshes, we use 3 axis-aligned rays to determine a point's label. A majority vote helps deciding on the correct label. The detailed algorithm is described on Figure 9.2.

INPUT: a 3D point p and a mesh $\mathcal{M}$ of the BIM model, structured as a set of object meshes $\mathcal{M}_s$ with a semantic label

label = *label*(*void*) // *Initialize the point label to void (i.e., empty)*
queries $\leftarrow$ {Ray($p, \mathbf{e}_x$), Ray($p, \mathbf{e}_y$), Ray($p, \mathbf{e}_z$)}// *Consider 3 orthogonal directions*
for $\mathcal{M}_s \in \mathcal{M}$ **do** // *For every semantic instance $\mathcal{M}_s$ in the global mesh*
  $n_{odd\text{-}inter} \leftarrow 0$ // *Initialize the counter of rays having an odd number*
  // *of intersections with the object mesh $\mathcal{M}_s$*
  for query $\in$ queries **do**
    if $|\text{query} \cap \mathcal{M}_s| \bmod 2 = 1$ **do** // *Count one for each query ray*
      // *having an odd number of intersections with $\mathcal{M}_s$*
      $n_{odd\text{-}inter} \leftarrow n_{odd\text{-}inter} + 1$
    end if
  end for
  if $n_{odd\text{-}inter} \geq 2$ **do** // *If at least two rays vote for being inside $\mathcal{M}_s$*
    label $\leftarrow$ *label*($\mathcal{M}_s$) > // *Assign the label of $\mathcal{M}_s$*
    break
  end if
end for
OUTPUT: label for point p

Figure 9.2: 3D point labeling from a labeled mesh algorithm.

9.4 POINT CLOUD SIMULATION

In many methods [PNM+20], the input point clouds are created synthetically by uniformly sampling the ground-truth surface. In this work, we go one step closer to a realistic LIDAR scan by sampling points by ray shooting from a set of viewpoints.

9.4.1 *Viewpoint generation*

Modern acquisition methods of indoor and outdoor environments make use of UAVs [CTLES17] to allow point cloud capture and registration from almost arbitrary viewpoints. To sim-

ulate this kind of acquisition given a mesh, some constraints have to be satisfied. First, viewpoints should not be inside a full volume. Moreover, in a realistic setup, viewpoints are spread across the model so that most surface points are visible from at least one viewpoint. Enforcing the latter condition is computationally expensive. We propose a heuristic in order to tackle this task with a cheaper computational cost. When a surface is not seen from any viewpoint, the area in the void close to this surface is also unseen from the viewpoints already stored, therefore, we sample viewpoint candidates in the void and check that they are not seen from the viewpoints already stored. Once we are unable to find such points, the viewpoint generation algorithm stops. A pseudo code for the algorithm is available on Figure 9.3.

INPUT: mesh $\mathcal{M}$ representing the whole BIM model.
$\mathcal{V} \leftarrow \emptyset$ *// Initialize the set of viewpoints as empty*
while $|\mathcal{V}| < N_{\text{max}}$ **do**
  repeat N_{iter} **times**
    //// Sample a candidate viewpoint in void
    Pick $v \in bbox(\mathcal{M})$ *// Uniform sampling*
    while $label(v) = void$ **do** *// Using the algorithm described in Figure 9.2*
      Pick $v \in bbox(\mathcal{M})$
    end while
    is_valid $\leftarrow$ true
    for $v_c \in \mathcal{V}$ **do** *// Compare the current viewpoint to those we already have*
      if segment$(v, v_c) \cap \mathcal{M} = \emptyset$ **then** *// If the viewpoints "see" each other*
        is_valid $\leftarrow$ false *// Then the candidate viewpoint is discarded*
        break
      end if
    end for
    if is_valid **then**
      $\mathcal{V} \leftarrow \mathcal{V} \cup \{v\}$
      break
    end if
  end repeat
end while
OUTPUT: set of point of views $\mathcal{V}$

Figure 9.3: Viewpoint generation algorithm.

9.4.2 *Ray shooting*

From each viewpoint, directions are uniformly sampled to define a set of rays. The first intersection between each ray and the global mesh gives us the scanned point. We also save the normal of the surface at the scanned point, which can also be efficiently estimated in practice [BM12], as well as the label of the intersected instance in the global mesh.

This provides a more realistic point cloud than a uniform sampling on the global mesh. Not only does it prevents point sampling on invisible mesh surfaces, e.g. where two objects are in contact, but it also provides a typical non uniform distribution of points across the surface (see Figure 9.4).

While this method leads to a more realistic simulation, it could still be improved: LIDARs typically embed a rotor which makes the distribution of shooting directions around the viewpoint non isotropic. Moreover, the sampled point's position on the ray is usually noisy. The level of noise varies with the incidence angle with respect to the normal of the scanned surface, the distance between the sensor and the scanned surface, and the texture and albedo of the surface. Improvements on the baseline virtual scanning method we provide could lead to more realistic features and close the gap towards real data.

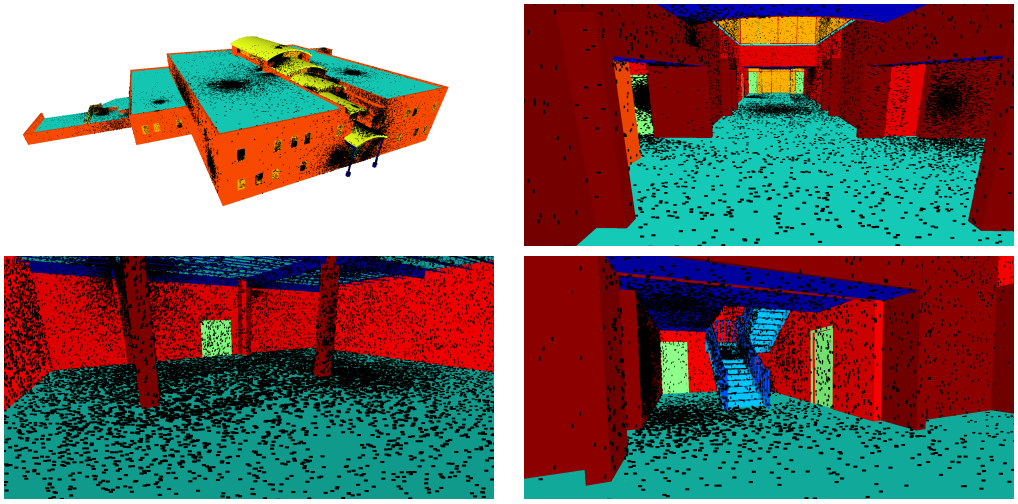
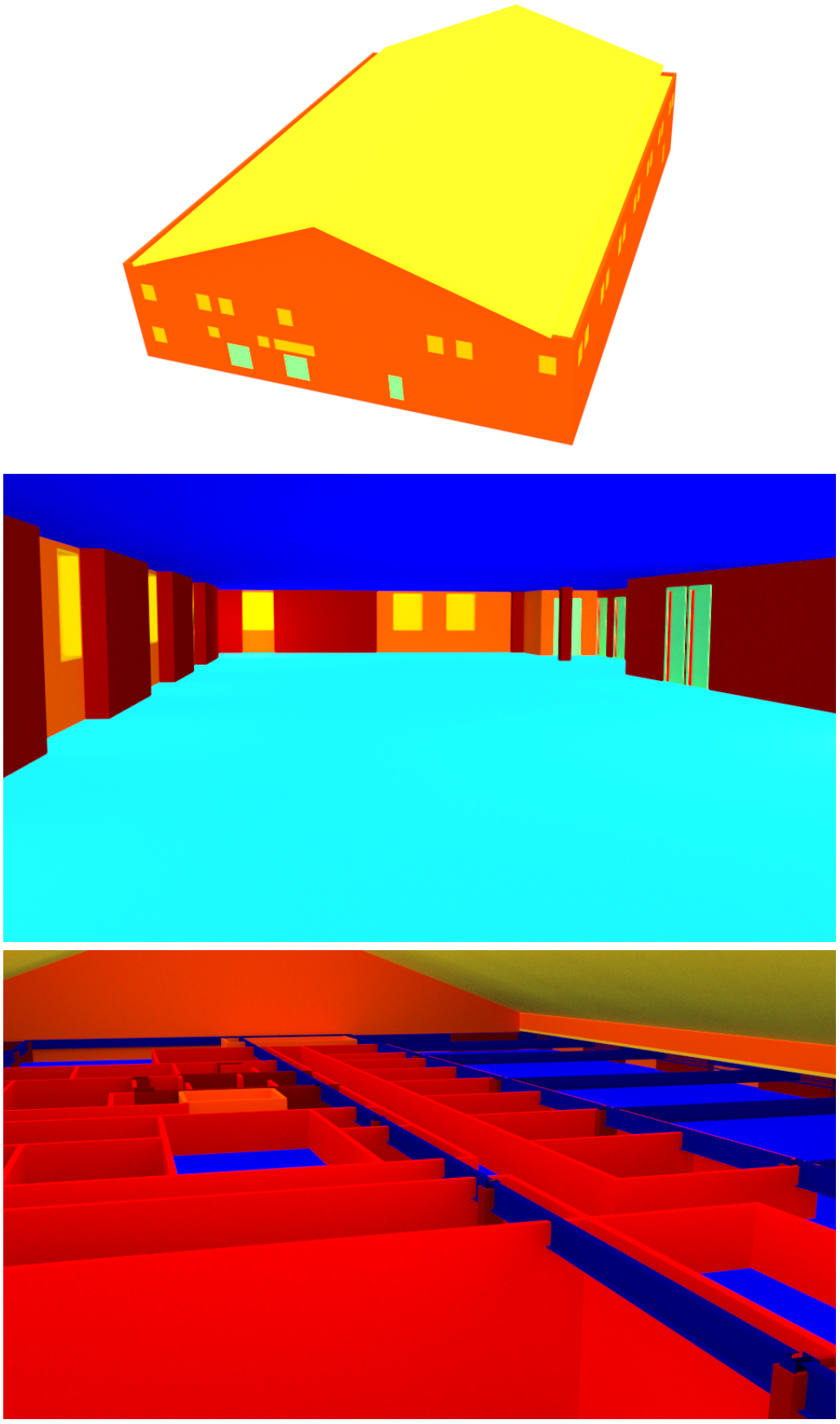


Figure 9.4: Representation of the sampled point cloud on the ground truth mesh (test set). The point distribution is not uniform: it is denser closer to viewpoints.

9.5 TRAIN/TEST SPLIT

S3DIS [ASZS17] includes only 6 large floors coming from 3 buildings. The train/test set is made as to avoid that similar parts are seen in both tests. We leverage the diversity of buildings in VASAD to propose a train/test split where the test set consists of a full building. Since our buildings have different sizes, we choose as a test building NBU_MedicalClinic which has a medium size and a variety of semantic components (see Figure 9.1).

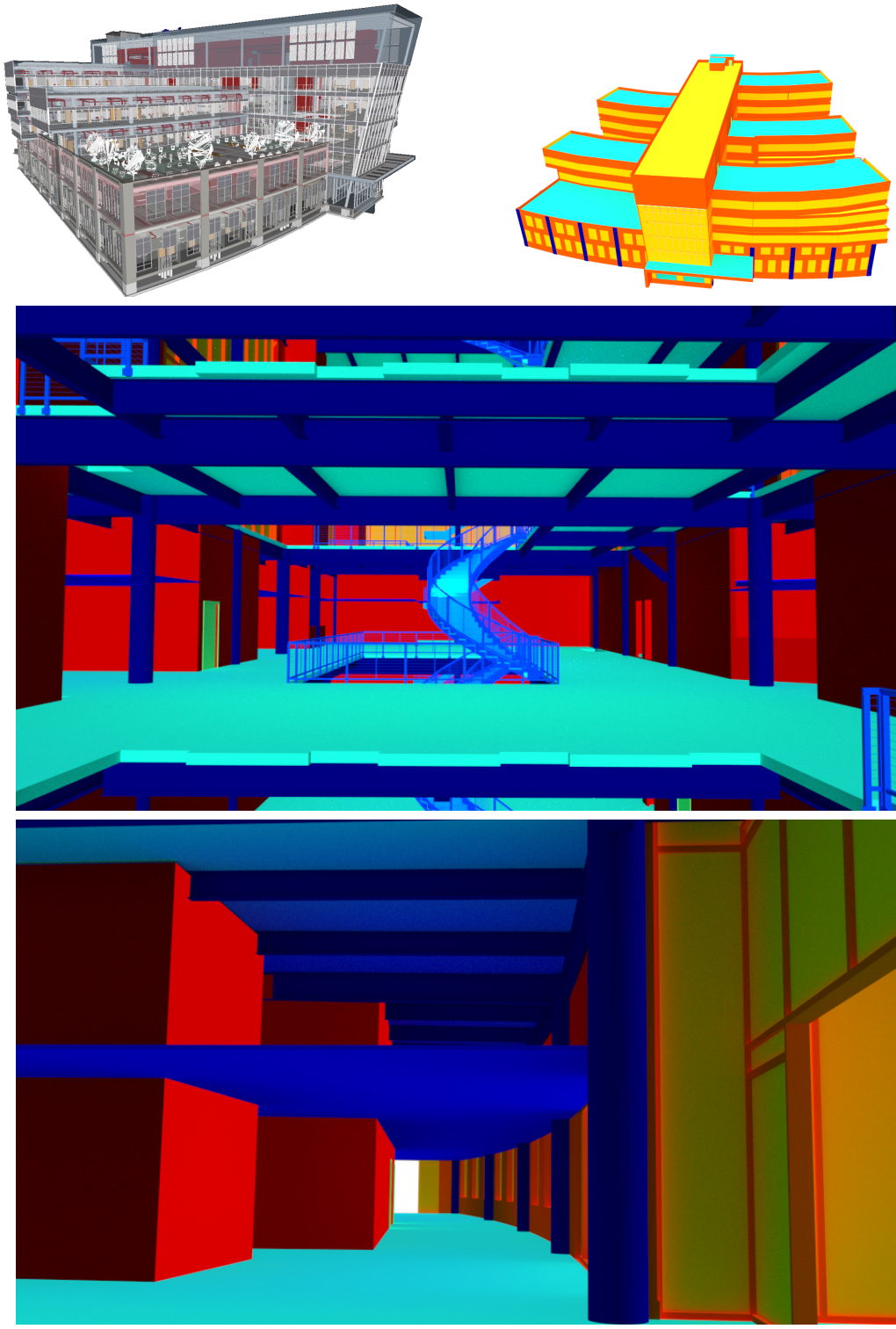


(1) NBU_OfficeBuilding

Artwork 2: Exterior overview and interior details of NBU_OfficeBuilding.

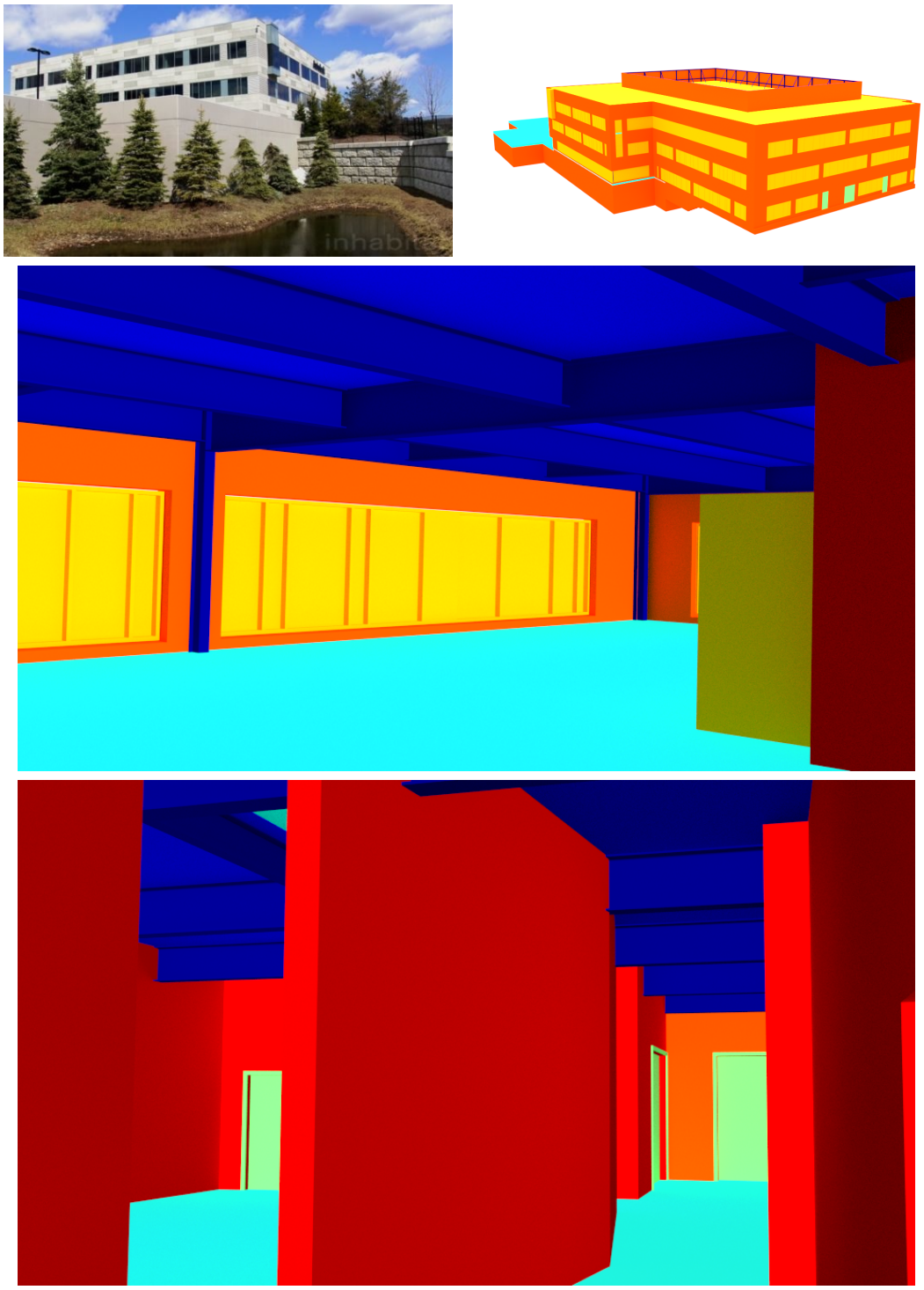
(2) Sextant

Artwork 3: Real [Com17], exterior overview and interior details of Sextant.



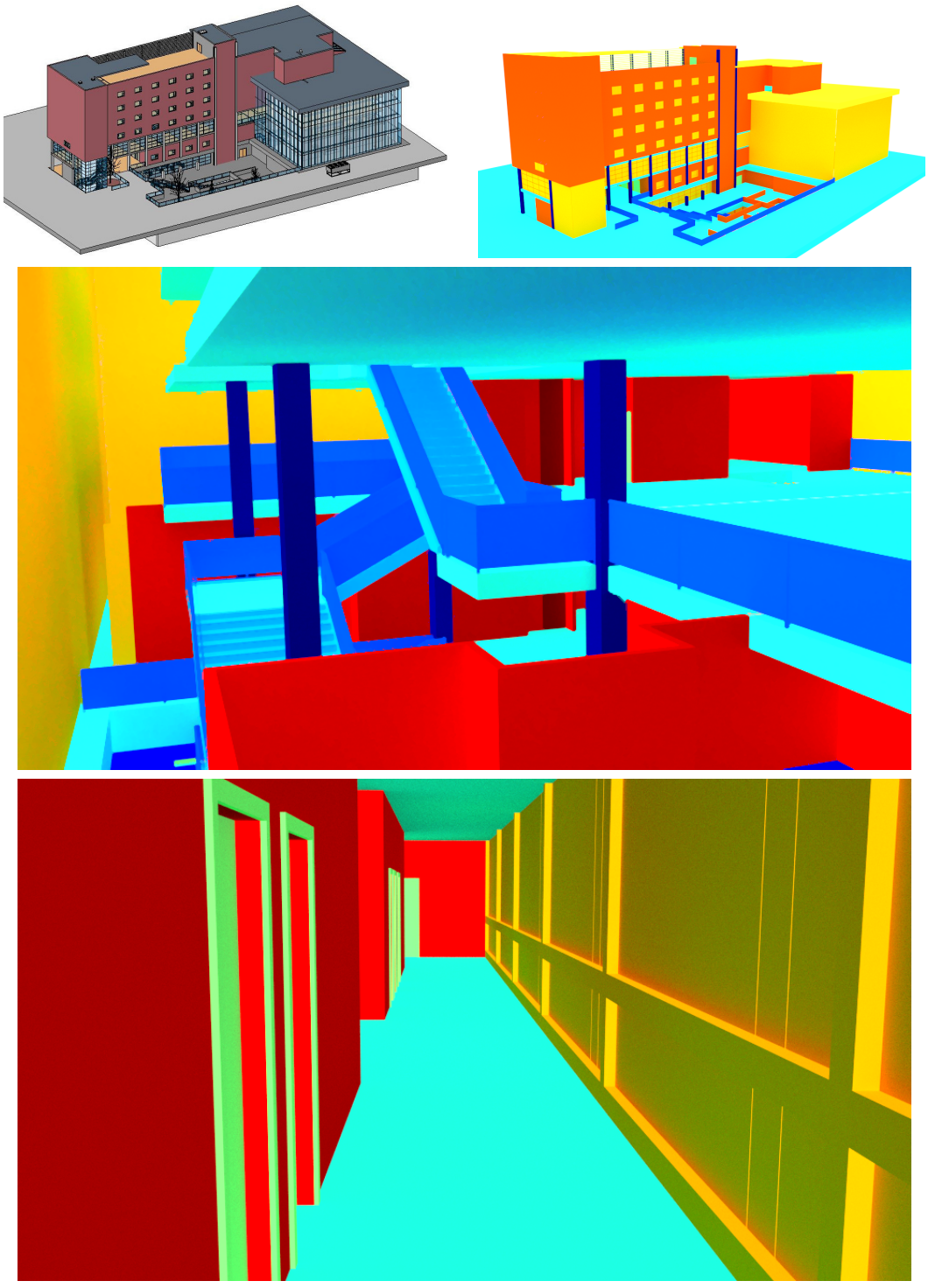
(3) WestRiverSideHospital

Artwork 4: Exterior overview and interior details of WestRiverSide-Hospital.



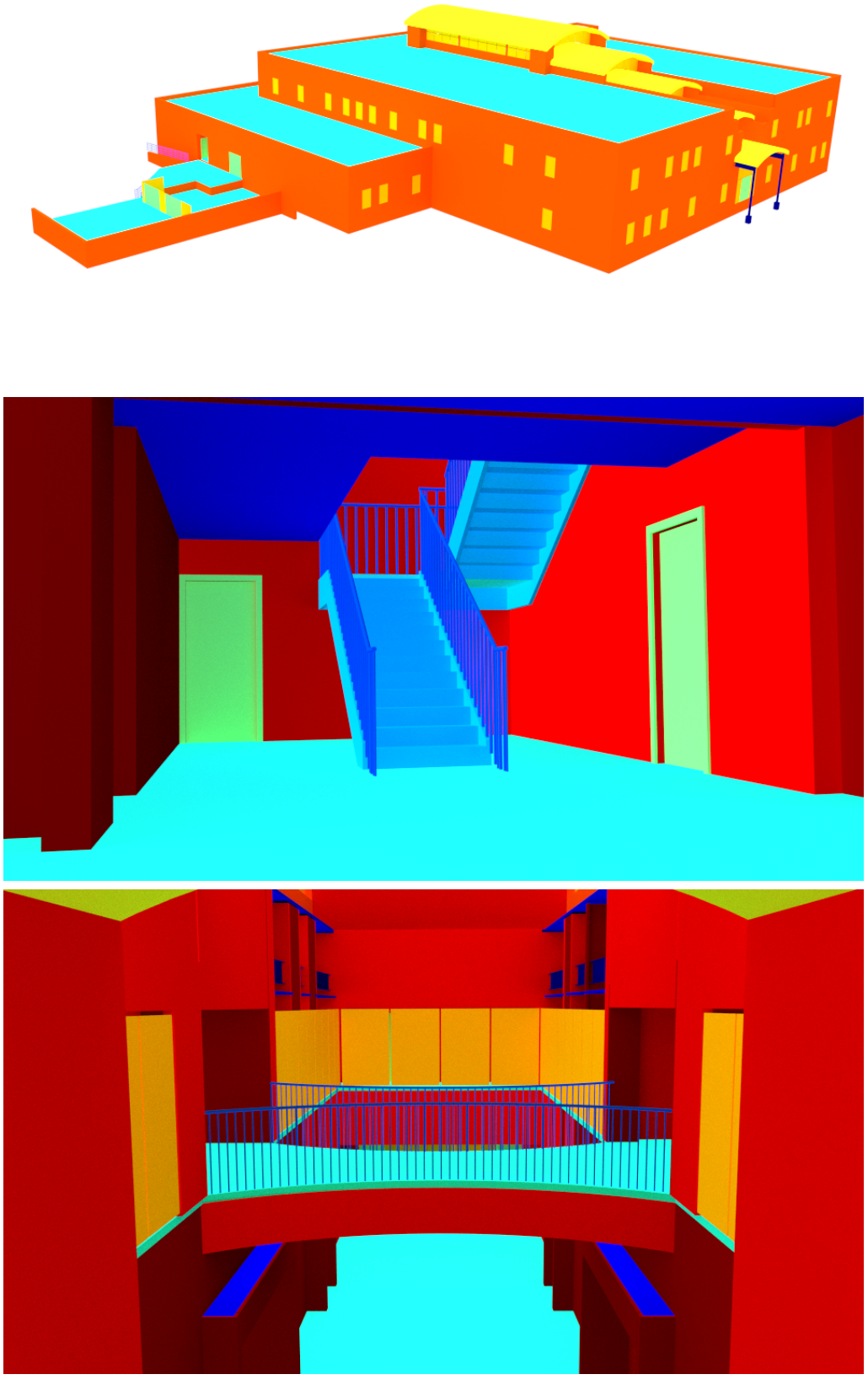
(4) Trapelo

Artwork 5: Real, exterior overview and interior details of Trapelo.



(5) OTC-ConferenceCenter

Artwork 6: Exterior overview and interior details of OTC-ConferenceCenter.



(6) NBU_MedicalClinic

Artwork 7: Exterior overview and interior details of NBU_MedicalClinic.

10

Method

◂ Chapter 9
Chapter 11 ▸

10.1 RECONSTRUCTION APPROACHES

As previous approaches essentially focused on surface reconstruction only, the task of reconstructing both the volume and the semantics that we propose is novel. We introduce two methods to perform volume reconstruction with semantics. The first one is a composite point-voxel model which extracts rich semantic features with a raw-point network and then handles semantic reconstruction with a voxel network. The second one is a direct extension of the state-of-art ConvONet method for surface reconstruction [PNM+20] to handle semantics.

10.2 PVSRNET

Voxels are a very natural way to apply learning methods to 3D reconstruction because they allow a direct generalization of the methods used for images. In particular, the UNet [RFB15] architecture has successfully been applied for image segmentation. Its direct 3D generalization has already been used in order to perform a dense volume segmentation from sparse annotations [ÇAL+16]. One of the main drawbacks of the voxel representation is its memory consumption. In practice, attempts have been made to exploit the sparsity of the reconstructed space [TDB17]. This approach involves leaving the decision whether to subdivide the space to reconstruct to the network. In our case, we keep a vanilla UNet and we leverage the existence of point convolution networks [BPM20] which are able to process a larger part of space at once and to therefore leverage an additional context. This is particularly important in the context of semantic segmentation of buildings: locally, a ceiling and a slab can look very similar, and the relative position of these elements in the building can help disentangling this ambiguity.

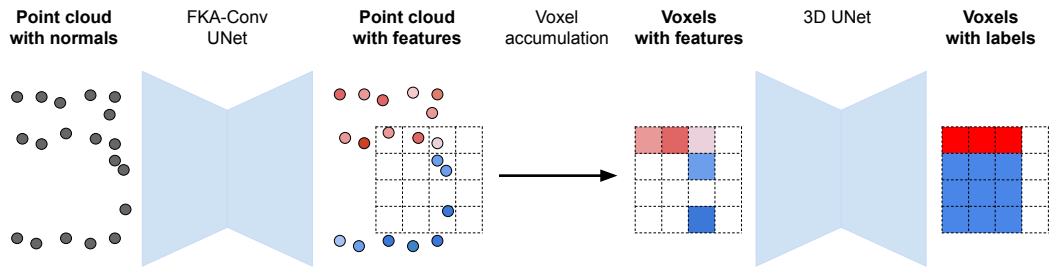


Figure 10.1: PVSRNet is composed of two networks, one operating on a raw point cloud for rich semantic feature extraction (FKAConv sub-network) and the second one on voxels for semantic volume occupancy (3D UNet).

The method is summarized on Figure 10.1. The point cloud with normals is first processed at large scale by a FKAConv-based [BPM20] UNet. A feature is computed per point at this stage. Since the 3D space area is too big to be fully processed as voxels, only a smaller part (in practice a cubic chunk of 2m40 side) is accumulated in a set of

48x48x48 voxels: when several points fall in the same voxel, the voxel feature is computed as the average of the points' features. At this stage, only a subset of the voxels close to the surface is filled with information. Then, the voxels are fed in a 3D UNet whose role is to propagate the information across the whole volume.

In theory, this model is end-to-end trainable, but this was not practically tractable with our computation means. Therefore, we first trained the FKA-Conv UNet thanks to the point's label that are recorded during the virtual scan (See 9.4.2). The loss we use is a cross-entropy loss and the features we keep are the raw logits (before the softmax). The obtained point clouds with features are then used to build chunks of voxels. Each voxel chunk is a cube whose side has 48 voxels of 5cm each. The second step consists in using these chunks to train the 3D Unet thanks to, again, a cross entropy loss applied on the voxels.

For simplicity, we refer to this approach as *PVSRNet* for Point-Voxel Semantic Reconstruction Network.

10.3 SEMANTIC CONVOLUTIONAL OCCUPANCY NETWORK

ConvONet [PNM+20] reconstructs volumes as an implicit representation from a 3D point cloud. It is based on the simpler Occupancy Networks [MON+19] which consist in a multi layer perceptron mapping the 3D space to an occupancy (typically with value 1 inside the shape and 0 outside), conditioned on a latent representation of the whole input. A major limitation of Occupancy Networks [MON+19] is that they make use of a unique latent vector to represent the whole input point cloud, regardless of its scale and complexity which prevents it to be used for scene reconstruction in practice.

In ConvONet [PNM+20], the authors first subdivide the input point cloud into large voxels. For each of these large voxels, a PointNet [QSMG17] encoder takes as input the points located in the considered voxel and provides a local latent vector. This set of latent vectors is then further refined thanks to a 3D UNet [ÇAL+16]. Finally, for each arbitrary point in the 3D space, occupation is obtained by trilinear interpolation of the latent vector, which is fed to a decoding multi-layer perceptron similar to the one used in [MON+19].

To obtain semantic information on top of pure geometry, we modify the predicted occupancy function: instead of only predicting a binary occupancy, for a given point, we predict a semantic class, including void. The network is then trained with a standard cross-entropy loss. At inference time, the authors of [PNM+20] directly apply the marching cube [LC87] algorithm on the output logit, which allows a smooth interpolation between the marching cube voxels. In our case, we first apply a hard argmax on the logits, and the marching cubes algorithm is applied for each non-void semantic class against all other classes. Therefore, the global reconstructed mesh includes one mesh per class. Using a hard softmax does not allow a smooth interpolation (hence the voxelized aspect of the output), but it allows to perfectly stitch two volumes of different class, without in-between void or overlap, which is a typical requirement for BIM applications. If a smoother mesh is needed for a specific application, it is possible to use a finer marching cubes resolution, at the cost of additional computation time.

10.4 DATA PREPARATION

Input and outputs are different for PVSRNet and SemConvONet (the semantized version of ConvONet). This section describes how the VASAD dataset is prepared for training

with each method.

- **PVSRNet**: As described in section 10.2, PVSRNet is composed of two sub-networks. One dealing with points (FKAConv) and the other processing voxels (3D-UNet). In FKAConv, a 6m-diameter ball is used to select subsets of points at training time. The obtained point features are then aggregated in voxels of 5cm. Each chunk of voxels has 48 voxels per side, which yields a 2.40m square. In order to label the voxels, we follow a strategy that has a bias towards the full space, as to balance the fact that 70% of the models is composed of void space. For each voxel, we sample 27 points on a regular grid and get their labels thanks to the algorithm described in Figure 9.2. If and only if all points are labeled with void, the voxel gets a void label. Otherwise, the voxel gets the value of the most represented full class among the points. This allows us to recover thin objects such as windows which often fail to be represented by the majority of the 27 points. For each sampled point on the surface, the line segment between the point and the point of view from which it was sampled constitutes a visibility information. This information is stored at voxel level thanks to a binary variable which stores whether a given voxel is intersected by a visibility segment or not. The routine to efficiently compute this information is described on Figure 10.2.

- **Semantic Convolutional Occupancy Network**: We strive to mimic the data format used in the original implementation. We randomly sample cubes of 4m side in the 3D space and apply a random Y-axis rotation. We select the input points which lie in this cube. As a supervision, and to be fair with the bias towards full spaces we introduced when preparing the data for PVSRNet, we first sample points on a 5cm voxel grid, and 27 points per voxel on an inner regular grid. This generates more points than needed in the original algorithm, we therefore apply the following selection: for each 5cm-side voxel, we make sure to have at least one point per class that is represented in the voxel. This typically yields more than half of the 1M we wish to consider per chunk. The remaining points are randomly selected among the rest.

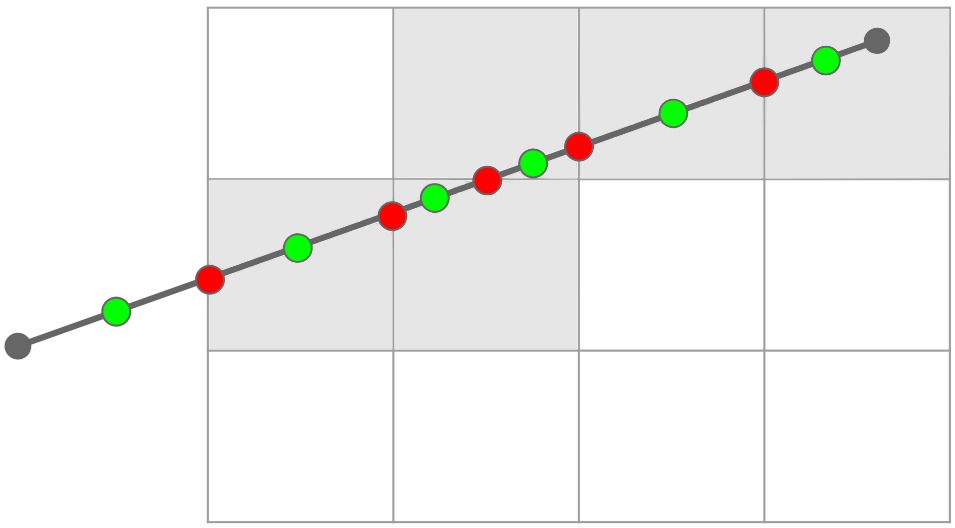


Figure 10.2: Fast visibility computation. The voxels considered on a line of sight (highlighted) are computed by querying the voxels in which lie the mid-points (green) of the line segment end-points and the intersections (red) of the line segment and the voxels facets, all of which are easily computed.

11

Results

◂ Chapter 10
Chapter 12 ▸

11.1 METRICS

The task of volume reconstruction with semantic is manifold, therefore it needs different metrics to quantitatively assess each aspect of the resulting 3D model. In particular, both geometric and semantic aspects need to be evaluated. Moreover, we not only propose metrics that evaluate the volume aspect of the reconstruction, but we also propose surface metrics that evaluate the quality of the reconstructed surface.

11.1.1 *Volume metrics*

Evaluation is carried out by uniformly sampling points $\mathcal{P}_v$ in the union of the bounding box of the ground-truth and the predicted 3D models. In practice, we sample 10 million points and we noticed that increasing the number of points left the metrics stable. We then find the ground-truth label $l_{p,gt}$ and predicted label $l_{p,pred}$ for each point $p \in \mathcal{P}_v$ thanks to the algorithm presented in Figure 9.2. All the volume metrics that we build are based on these labels. We denote by $void$ the void label.

- **Semantic IoU** (intersection over union): Among each point in $\mathcal{P}_v$ that are labeled as full in either the ground truth or the prediction, we look at the ratio of these points which have the same label in both the ground truth and the prediction.

$$IoU_s = \frac{|\, p \in \mathcal{P}_v \text{ such that } l_{p,gt} \neq void \text{ and } l_{p,gt} = l_{p,pred} \,|}{|\, p \in \mathcal{P}_v \text{ such that } l_{p,gt} \neq void \text{ or } l_{p,pred} \neq void \,|}$$

- **Geometric IoU**: Some models can predict a correct partition of the 3D space between the binary labels (full/void), but misclassify the full space (e.g., wall vs window confusion). In this case, even if the semantic is incorrect, the geometry is still recovered. We evaluate this thanks to the geometric IoU:

$$IoU_g = \frac{|\, p \in \mathcal{P}_v \text{ such that } l_{p,gt} \neq void \text{ and } l_{p,pred} \neq void \,|}{|\, p \in \mathcal{P}_v \text{ such that } l_{p,gt} \neq void \text{ or } l_{p,pred} \neq void \,|}$$

- **Confusion matrix**: when working with classification tasks, the error is usually not evenly spread across the different classes. The best way to quantitatively visualize this is by building a confusion matrix C whose term $C_{i,j}$ counts the number of points in $p \in \mathcal{P}_v$ whose ground truth class is i and whose predicted class is j. A perfect classification would therefore generate a diagonal confusion matrix.

$$C_{i,j} = |\, p \in \mathcal{P}_v \text{ such that } l_{p,gt} = i \text{ and } l_{p,pred} = j \,|$$

  For the sake of visualization and understanding, each row C_i is normalized by the total number of points whose ground truth class is i: $\sum_j C_{i,j}$.

11.1.2 *Surface metrics*

Our surface metrics evaluate the quality of the geometry of the reconstructed surface. To do so, we sample points $\mathcal{P}_{s,gt}$ on the ground-truth mesh and points $\mathcal{P}_{s,pred}$ on the predicted mesh. In practice, $|\mathcal{P}_{s,gt}| = |\mathcal{P}_{s,pred}| = 10$ millions.

We then evaluate the distance between each point in $\mathcal{P}_{s,pred}$ and its nearest neighbour in $\mathcal{P}_{s,gt}$:

$$D_{pred \to gt} = \{||p - NN_{\mathcal{P}_{s,gt}}(p)|| \text{ for each } p \in \mathcal{P}_{s,pred}\}$$

where $NN_Y(x)$ is the nearest neighbour of x in the set Y.

Symmetrically, we evaluate the distance between each point in $\mathcal{P}_{s,gt}$ and its nearest neighbour in $\mathcal{P}_{s,pred}$:

$$D_{gt \to pred} = \{||p - NN_{\mathcal{P}_{s,pred}}(p)|| \text{ for each } p \in \mathcal{P}_{s,gt}\}$$

All the surface metrics we build are based on these distances.

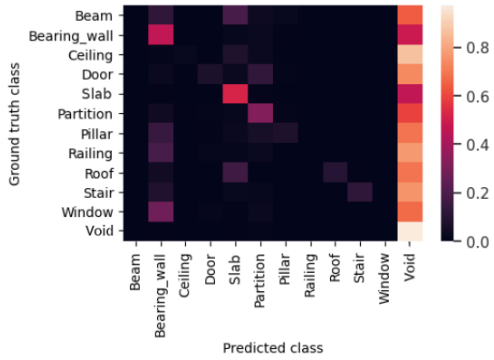
- **Mean surface distance**: the average surface error is given by averaging all the nearest neighbour distances we calculated. This distance is usually called the Chamfer distance in the literature.
$$\text{Av. dist.} = \text{mean}(D_{pred \to gt} \cup D_{gt \to pred})$$
- **Max surface distance**: the worst surface error that we make, also called the Metro distance.
$$\text{Max. dist.} = \max(D_{pred \to gt} \cup D_{gt \to pred})$$
- **Precision**: this metric tells how close the reconstructed surface is to the ground truth. To avoid considering the worst errors, a threshold of 95% is taken to discard the 5% worst distances:
$$\text{Prec.} = percentile_{95\%}(D_{pred \to gt})$$
- **Accuracy**: this metric evaluates to what extent the ground truth surface is reconstructed. As well as for the precision, a threshold of 95% is used to discard the 5% worst distances:
$$\text{Acc.} = percentile_{95\%}(D_{gt \to pred})$$

11.1.3 *Metric Oracle*

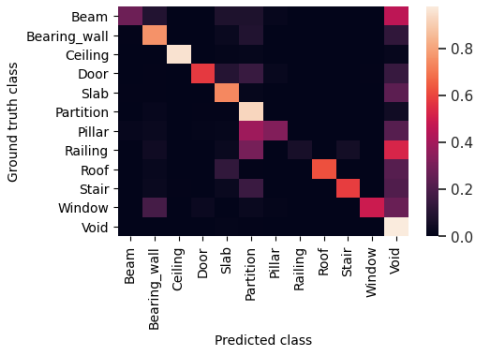
Since all our metrics involve a discretization through point sampling, we also provide an oracle which computes each metric for the validation model against itself on Figure 11.1.

11.2 SURFACE RECONSTRUCTION

We present quantitative (Table 11.1) and qualitative results (see Figure 11.2, Figure 11.3, Figure 11.4, Figure 11.5, Figure 11.6, Figure 11.7 and Figure 11.8) on the VASAD dataset. In both methods, the use of voxels on one hand, and the marching cubes algorithm on the other hand generate highly complex meshes. Because the amount of triangle we can display at once is limited, we had to decimate the meshes thanks to the "Quadric Edge Collapse Decimation" available on the software Meshlab [CCR08]: for each 2.40m-side chunk, we set a maximum budget of 2500 faces. This simplification was used only for



Convolutional Neural Networks [PNM+20]



PVSRNet (normals + point semantic features)

Figure 11.1: Left: the confusion matrix for SemConvONet (with normals). Right: the confusion matrix for PVSRNet with normals and surface semantics as input.

Method	Input Nls	Input Vis.	Input Surf. Sem.	Semantic	Surface metrics Acc. ↓	Surface metrics Prec. ↓	Surface metrics Av. ↓ dist.	Surface metrics Max. ↓ dist.	Volume metrics IoU Sem. ↑	Volume metrics IoU Geo. ↑
Oracle	-	-			0.07	0.07	0.03	0.21		
ConvONet (points)	✗	✗	✗	✗	0.16	0.67	0.13	5.69	-	0.31
	✗	✗	✗	✓	0.21	0.16	0.08	5.80	0.25	0.29
	✓	✗	✗	✓	0.19	0.17	0.08	4.30	0.23	0.28
PVSRNet (voxels)	✗	✗	✗	✓	0.13	0.13	0.06	**1.98**	0.37	0.49
	✓	✗	✗	✓	**0.11**	**0.11**	**0.06**	2.61	0.41	0.55
	✓	✓	✗	✓	**0.11**	0.18	0.07	6.46	0.41	0.55
	✗	✓	✓	✓	0.13	0.18	0.09	2.31	0.51	0.56
	✓	✗	✓	✓	0.12	0.14	0.07	6.87	**0.53**	**0.59**

Table 11.1: Quantitative results: we evaluate both the reconstructed volume and surface. Metrics: accuracy, precision, average distance, maximum distance, semantic intersection over union, and geometric intersection over union (considering only full/void).

display purposes, and the quantitative evaluation was carried on the raw output data. In the figures representing the input point clouds, each point is colored darker as it gets closer to the viewer, in order to better perceive the underlying geometry. Again, this only has a visualization purpose and it does not carry any additional information.

The scanned point cloud is given to all the tested networks as input, but additional input data are also considered: the normal vector associated to each point, the visibility information (see Figure 10.2), and the per-point classification vector obtained thanks to FKAConv [BPM20]. Table 11.1 summarizes all the tested combinations of input.

We first report the scores for two baselines: the oracle and ConvONet [PNM+20]. The latter is trained on our data by aggregating all the non-void semantic labels into a full label. The task being to segment the space between inside and outside the objects volume.

We train 2 versions of SemConvONet: first with raw point clouds as input, and second with additional normal information as an input to the PointNet encoders. Compared to the original, purely geometric algorithm, we notice that only the accuracy slightly worsens. The rest of the metrics tends to improve.

As an ablation study for our approach, we trained three models, not relying on point cloud semantics. This is tantamount to using the voxel sub-network of PVSRNet alone. Similarly to the SemConvONet experiment, we consider variants with and without the input normals at the voxel level. In spite of the voxel discretization inherently restricting the precision of the reconstructed surface (see the difference in reconstruction quality on the floor on Figure 11.2), our method does better than any version of ConvONet, whether semantics is used or not. Experiments also show that Convolutional Occupancy Network is very sensitive to the input sampling (cf. Figure 11.2, 4th row). As a matter of fact, experiments originally conducted by the authors of [PNM+20] include a dense uniform sampling on the input meshes, as opposed to our dataset in which the input points are more realistically sampled from virtual point of views.

For information, we reported two experiments in which we included the visibility information, though PVSRNet did not seem to benefit from it. A possible explanation is

that the oriented normal information already indicates locally in which direction the void is located.

Finally, we report the scores of PVSRNet in the last row of Table 11.1. It leverages both the normals and the rich semantic features from the FKAConv sub-network at voxel level. With respect to the surface metrics, it is slightly worse but comparable to the other variants, except for the maximal distance, due to very rare, but very penalizing missing parts such as the parking lot of Figure 11.2, last row. The main reason of this local failure is the extremely low point density on this part of the building.

11.3 SEMANTIC SEGMENTATION

The semantic classes in buildings are highly non uniformly distributed in terms of volume. Structural classes such as bearing walls, partition and slabs are more represented than smaller features such as beams, doors, pillars, railings, stairs or windows. Consequently, we see that in spite of the bias towards full spaces in our supervision (see Figure 10.4) which increases the supervision signal on the smaller features, they are harder to infer while the majority classes are still well recovered by both methods. However, the Semantic Convolutional Network is not as good as PVSRNet at resolving ambiguities (see the ambiguity partition/bearing wall on Figure 11.3).

Other classes sometimes have an inherent ambiguity which is well observed in the results (see the confusion matrix in Figure 11.1):

- Partition / bearing wall: partitions tend to be thiner than bearing walls, and bearing walls tend to be mostly on the outermost part of buildings. However, exceptions remain which can make these classes hard to distinguish.

- Bearing wall / window: windows tend to be located on bearing walls, and the transition between the two classes can be hard to see on the point cloud. Variations in the normals and thickness changes are typical markers of a transition between a bearing wall and a window.

- Partition / door: similarly, and for the same reasons as for bearing walls and partitions, a close door can be a hard feature to distinguish from its neighboring partition at point cloud level.

- Slab / roof: In modern architecture, flat roofs are becoming more and more common. It's even possible for people to access these parts of the building. As a consequence, there is an intrinsic ambiguity between the two classes.

In spite of these ambiguities, we notice that the Semantic Convolutional Occupancy Network is not able to properly recover the small classes. In particular, we can notice the windows on Figure 11.2, the door and the beams on Figure 11.3, the railings on Figure 11.4 and the windows on Figure 11.6 and Figure 11.7. The IoU scores on Table 11.1 as well as the confusion matrix on Figure 11.1 also show that our method benefits a lot from the FKAConv [BPM20] point features (+10% in IoU).

One major limitation of PVSRNet is that the transition between the semantic classes is not entirely satisfactory yet. This is especially true when the transition does not include a significant change in the surface normal: doors/partitions, windows/bearing walls for instance. Moreover, PVSRNet is limited by the voxel resolution which prevents it to properly represent classes with small details such as railings (see Figure 11.4).

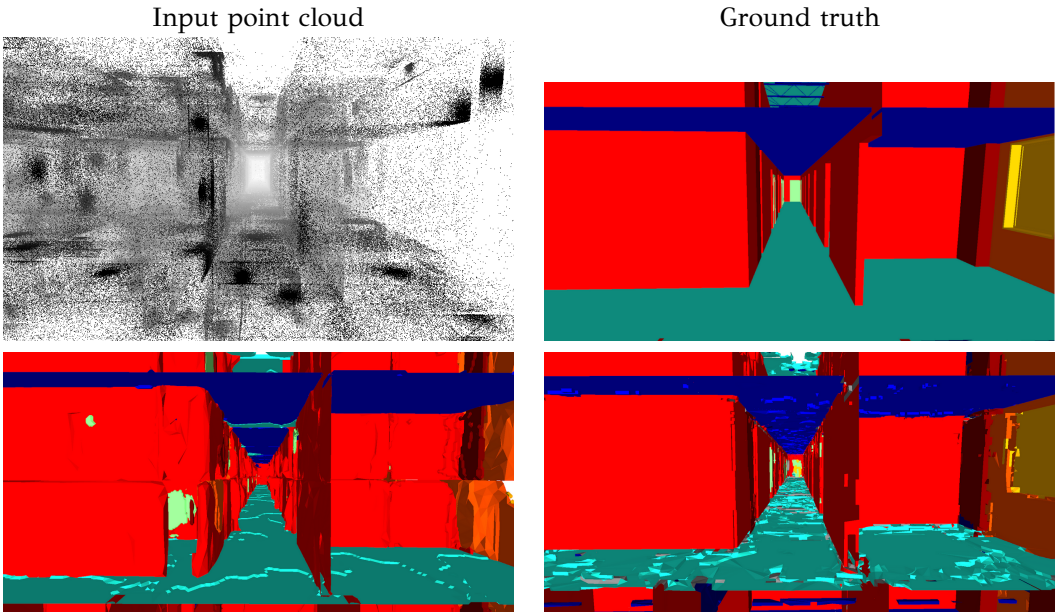


Figure 11.2: Qualitative results on the test set (NBU_MedicalClinic).

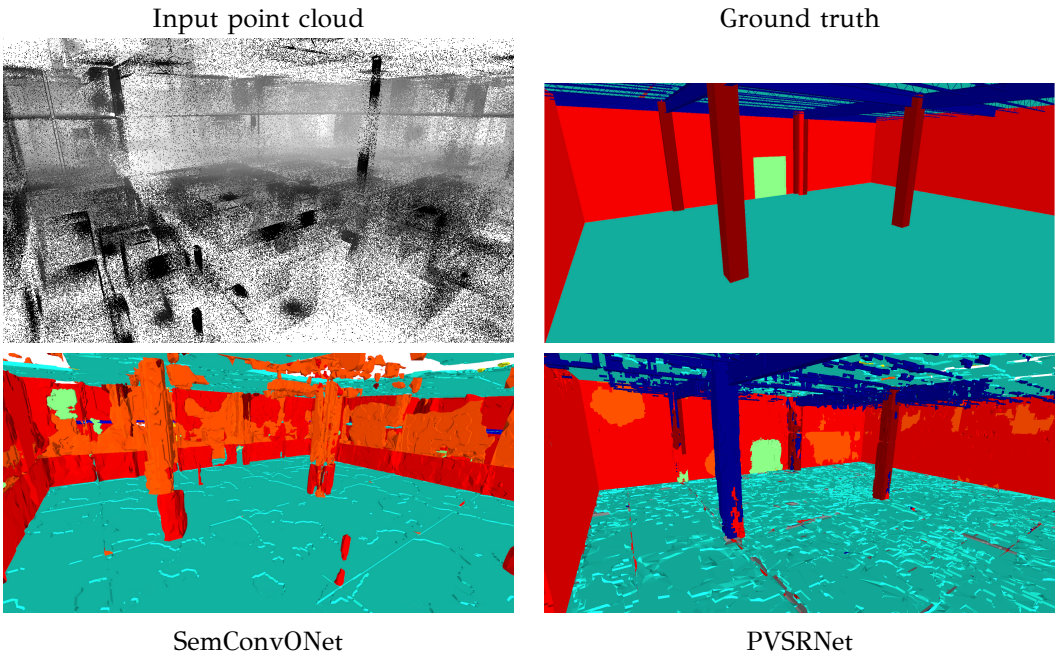


Figure 11.3: Qualitative results on the test set (NBU_MedicalClinic).

11.4 DISCUSSION

11.4.1 *Inherent ambiguities*

The dataset is built from 6 building information models designed by various architects all of which can model a same building with slight differences. As previously mentioned in Section 11.3, the classes can have an inherent ambiguity, especially when only considering the point cloud as an input. A perspective for dataset improvement, besides augmenting it with new buildings, is to enrich the models with texture and/or material simulation. The LIDAR simulation could then be more sophisticated and produce intensity based on the

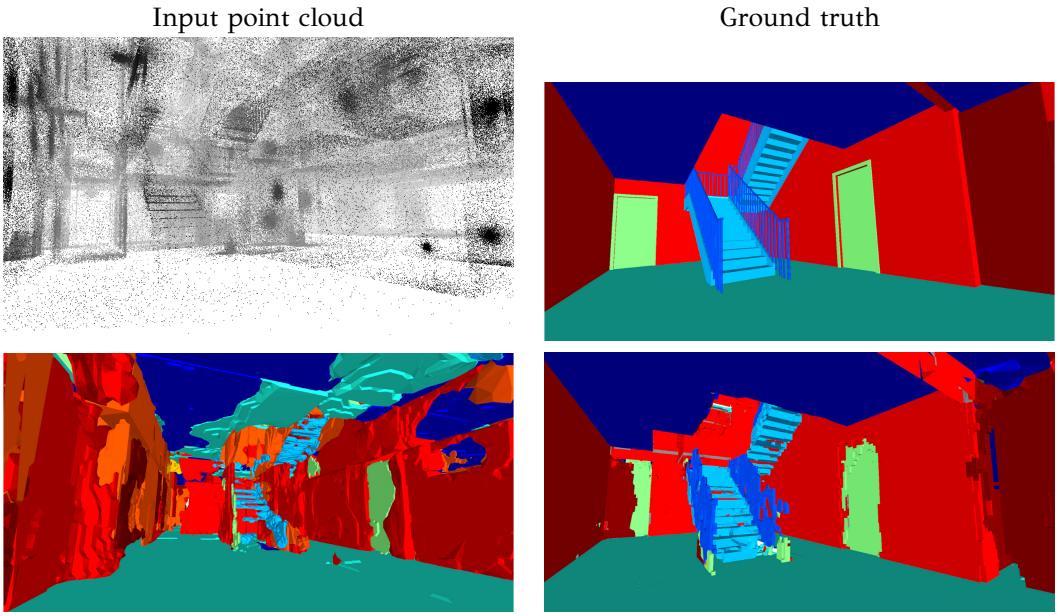


Figure 11.4: Qualitative results on the test set (NBU_MedicalClinic).

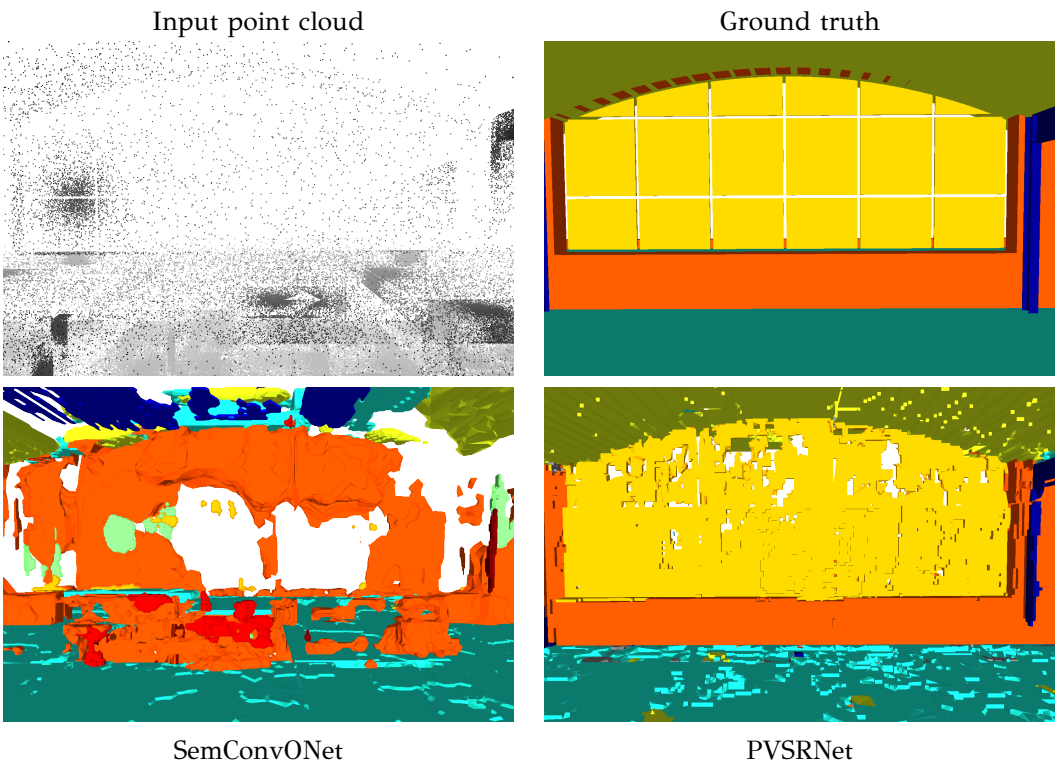


Figure 11.5: Qualitative results on the test set (NBU_MedicalClinic). The sampling is highly non uniform in this area. Our method is more robust to the sampling inconsistencies

material.

11.4.2 *PVSRNet*

Based on our observations, the main limitation of PVSRNet is the voxel size. The voxel resolution prevents it from properly representing classes with small details such as railings (see Figure 11.2, 3rd row). Implicit representations are an appealing alternative

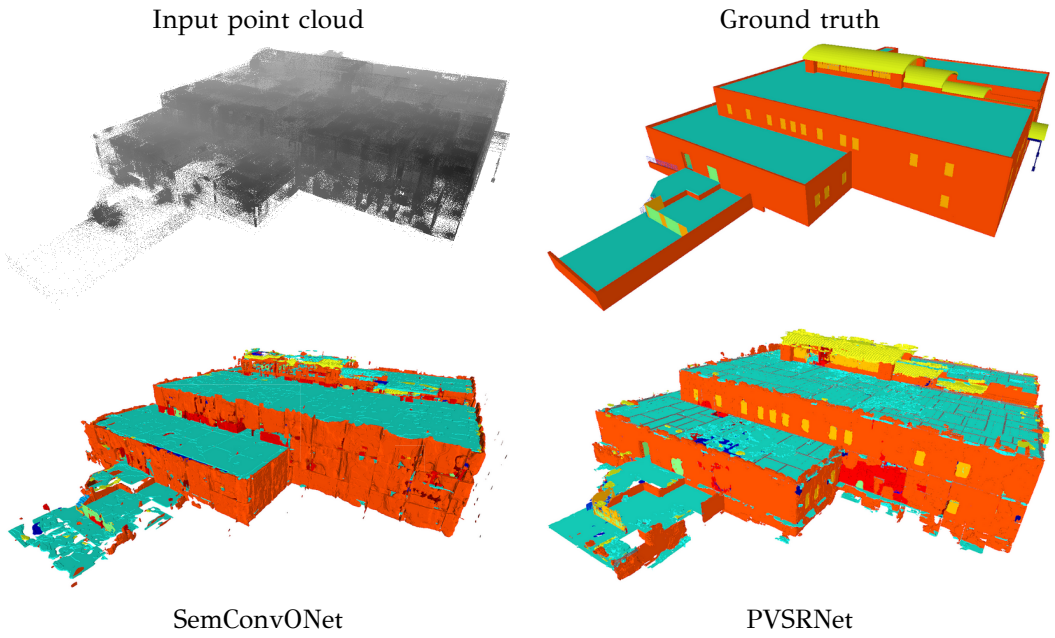


Figure 11.6: Qualitative results on the test set (NBU_MedicalClinic).

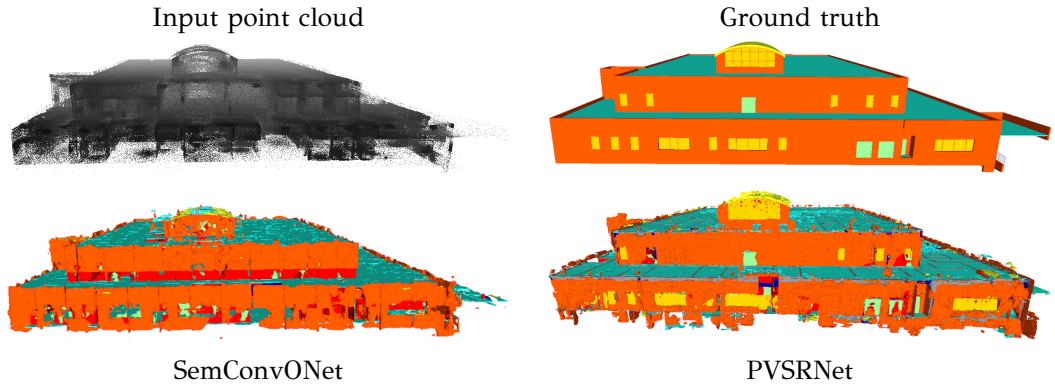


Figure 11.7: Qualitative results on the test set (NBU_MedicalClinic).

to voxel based approach. However, by using ConvONet as a baseline, we bring forward some limitations of the implicit representations: in spite of their inherent ability to represent any arbitrary closed surface, we show that irregularly-sampled surfaces are more efficiently reconstructed by a simpler voxel-based method. The issue is not limited to surface reconstruction, we also notice that the smaller classes are better recovered by our method and that the ambiguities are better resolved. One key difference between the voxel-based methods and the implicit representations lies in the loss: while the cross-entropy loss on voxels is dense (it provides supervision on each point of the 3D space), supervising an implicit representation involves a discretization strategy. It is future work to investigate methods with hybrid voxel/implicit supervision in order to take advantage of both representations.

11.4.3 *Towards BIM models*

As an output, we have a set of closed forms within which the semantic label is constant. This is probably sufficient to perform thermal or acoustic simulations. However, in the case of building modification, it may be necessary to have access to an idealized shape (e.g., a true parallelepiped instead of a set of voxels). This can be achieved thanks to

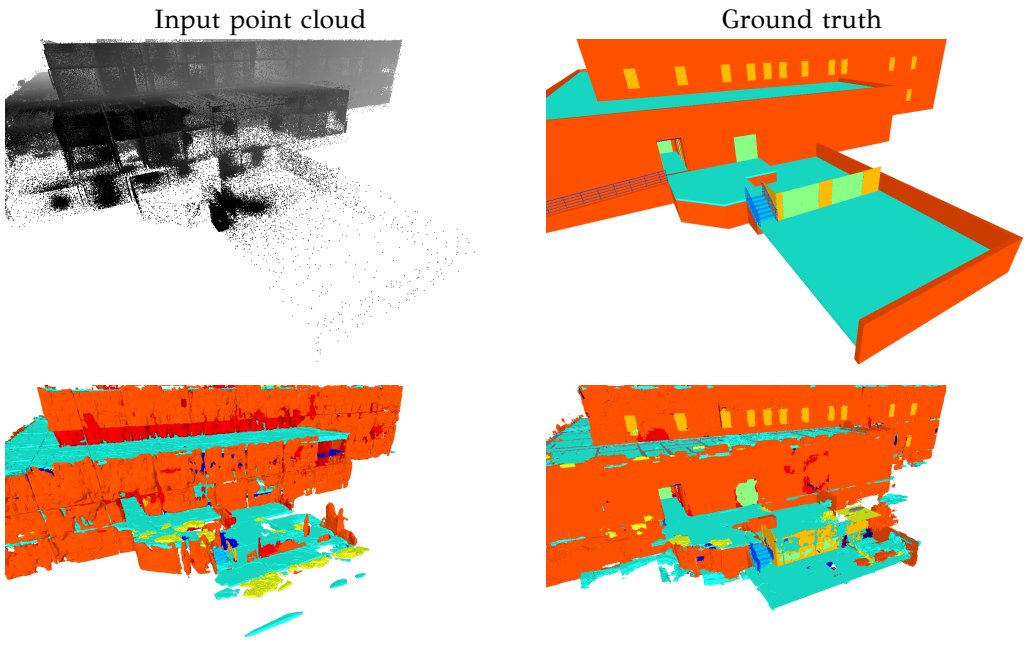


Figure 11.8: Qualitative results on the test set (NBU_MedicalClinic).A failure case: due to the point of view being far away, the point sampling is too sparse, leading to a missing chunk.

specific mesh simplifications algorithms [CMS98]. Moreover, it maybe necessary to further separate each instances (e.g., two perpendicular walls may be composed of 2 separate pieces). In any case, the BIM format does not define a unique way to represent a building and as such, this post processing step heavily depends on the downstream application.

11.5 CONCLUSION

We introduced VASAD, a dataset for geometric and semantic reconstruction of buildings. This dataset aims at addressing the lack of data focused on building structures which is a key components of modern BIM models in architecture and civil engineering. This dataset includes 6 clutterless (empty) buildings. A strategy to sample point of views in these buildings has been developed which further allows to simulate LIDAR scans. Moreover, a routine to label any arbitrary 3D point allows to build raw data for a large variety of 3D representations. We provide the full implementation to produce the data from the raw IFC file, and we therefore allow this dataset to be enriched from more IFC files. We have shown that this dataset is challenging enough for modern machine-learning-based reconstruction methods. We were able to highlight the limitations of learned implicit representations and we were able to propose a voxel-based method to perform semantic reconstruction of building which is able to recover small classes and to address some of the inherent class ambiguities. However, the non-uniform sampling of the input point cloud makes it challenging to reconstruct a smooth surface, and the transition between classes is not always clean. As such, we believe that this dataset paves the way for contributions that will eventually enable fully automatic so-called *Scan-to-BIM* methods.

Part V

EPILOGUE

12

Conclusion

◂ Chapter 11

12.1 LOOKING BACK

In terms of computer vision, automatizing the generation of a BIM model from sensors such as cameras or LIDARs involves the task of 3D reconstruction with semantic inference. 3D reconstruction is a complex task. As we have recalled, various input sensors are available to produce meaningful input data (images and point clouds for instance), and the choice of the 3D output representation (mesh, voxel, implicit representation, among others) plays a central role. Moreover, as for most of the tasks dealing with real-world data, the algorithms performance depend on the environment: in the case of buildings we highlighted two specificities in particular: the overwhelming presence of textureless surface, and the highly-structured geometry (large piecewise-planar areas, big volumes of the same semantic class such as walls, ceilings and roofs). With these observations in mind, we studied how to design a 3D reconstruction algorithm with semantics adapted to buildings.

- In part II, we focused on pure geometric reconstruction from a set of images with reasonable quality. We strove to understand and tackle the specificities of 3D reconstruction in the context of building (presence of textureless surfaces, presence of visible line segments due to the geometric regularity of the environment) and we were able to design a new algorithm which is able to outperform previous approaches on both synthetic and real data [LBM19].

- In part III, we took a step away from the building environments to better understand how to learn meaningful priors from a large dataset. We designed an algorithm which outputs a mesh from a set of low-quality calibrated images of an object using template deformation [LFW+21]. This approach competes with state of the art methods and is able to directly output a mesh.

- In part IV, we directly address the task of volumetric geometric reconstruction of buildings with semantics from LIDAR scans by introducing VASAD, a dataset specifically designed to learn this task. We introduce effective baselines to solve the task and show that structured 3D representations are the most promising to solve this task.

These contributions allow to push the boundaries of reconstruction algorithms and pave the way for more contributions in the generation of BIM models. We found that the use of machine learning is particularly effective to achieve this goal and that some of the specific corner cases met when dealing with buildings can be addressed.

With this work, we get closer to a *scan-to-BIM* algorithm. However, several aspects of the problem still need to be explored. In part IV, we studied the 3D reconstruction of buildings with semantics from artificial LIDAR scans. As discussed, a next step is to check that our assumptions hold well on real data. Moreover, LIDAR sensors remain expensive. It would therefore be relevant to generalize our algorithm to images as input, but again, there is a lack of relevant data to learn this task at building scale. When working on part IV, we noticed that the BIM files are usually not strongly constrained: for instance, there are several possibilities to split a continuous bearing wall (see Figure 12.1), depending on the construction method and the downstream task for which the model will be used. Clarifying these ambiguities or introducing conventions could pave the way for the introduction of instance segmentation in which not only would the parts be annotated by class, but each instance would be separated as in current BIM files.

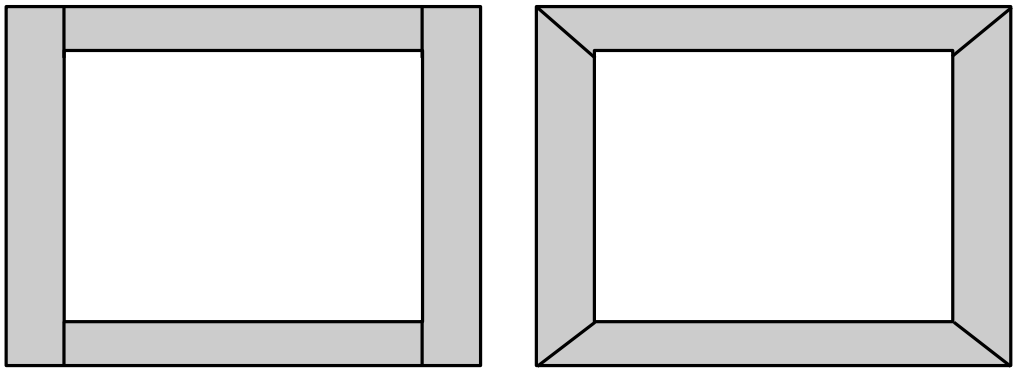
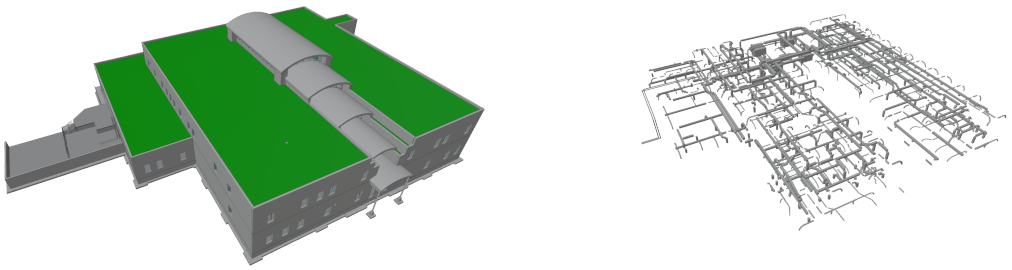


Figure 12.1: A few different partition possibilities for a bearing wall (sectional view from above).

As mentioned previously, the BIM digital representations usually include more than the geometry and the semantics of the visible parts. In particular, they can include information on the different networks (electrical, hydraulic), even if they are mostly hidden in the walls and slabs (see Figure 12.2). Recovering such invisible network would require additional sensors (e.g., Hall-effect based sensors among others) and a specific data processing.



Full BIM Model Hydraulic network only

Figure 12.2: Most of the hydraulic network of VASAD's Medical Clinic is not directly visible.

Finally, and closer to this work, promising 3D representations could allow a future breakthrough regarding the purely geometric and semantic aspects. The authors of [KAB+21] experiment 3D reconstruction as a set of 3D boxes which could be an interesting way to represent the very regular shapes of buildings, and which is a volume representation, and the neural render fields [MST+20] allow a promising representation of a scene learned from images.

References

[AAB19] Amir Atapour-Abarghouei and Toby P Breckon. Monocular segment-wise depth: Monocular depth estimation based on a semantic segmentation prior. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4295–4299. IEEE, 2019.

[ASZS17] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints*, February 2017.

[BdLGM14] Alexandre Boulch, Martin de La Gorce, and Renaud Marlet. Piecewise-planar 3D reconstruction with edge and corner regularization. *Computer Graphics Forum (CGF 2014)*, 2014.

[BdLGMK17] Amine Bourki, Martin de La Gorce, Renaud Marlet, and Nikos Komodakis. Patchwork stereo: Scalable, structure-aware 3D reconstruction in man-made environments. In *IEEE Winter Conference on Applications of Computer Vision (WACV 2017)*, March 2017.

[BENV06] H. Bay, A. Ess, A. Neubeck, and L. Van Gool. 3D from line segments in two poorly-textured, uncalibrated images. In *3rd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT 2006)*, pages 496–503, June 2006.

[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.

[BGM+19] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[BJ88] P.J. Besl and R.C. Jain. Segmentation through variable-order surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(2):167–192, 1988.

[BL08] Josep Miquel Biosca and José Luis Lerma. Unsupervised robust planar segmentation of terrestrial laser scanner point clouds based on fuzzy clustering methods. *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(1):84–98, 2008. Theme Issue: Terrestrial Laser Scanning.

[BL18] Jean-Philippe Bauchet and Florent Lafarge. KIPPI: kinetic polygonal partitioning of images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3146–3154, Salt Lake City, UT, USA, June 2018.

[BL20] Jean-Philippe Bauchet and Florent Lafarge. Kinetic shape reconstruction. *ACM Transactions on Graphics (TOG)*, 39(5):1–14, 2020.

[BM12] Alexandre Boulch and Renaud Marlet. Fast and robust normal estimation for point clouds with sharp features. *Computer Graphics Forum*, 31:1765–1774, 08 2012.

[BMR+99] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[BPM20] Alexandre Boulch, Gilles Puy, and Renaud Marlet. FKAConv: Feature-Kernel Alignment for Point Cloud Convolution. In *15th Asian Conference on Computer Vision (ACCV 2020)*, 2020.

[BRPMB17] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[BTL16] S. Baadel, F. Thabtah, and J. Lu. Overlapping clustering: A review. In *SAI Computing Conference (SAI 2016)*, pages 233–237, July 2016.

[BTS+14] Matthew Berger, Andrea Tagliasacchi, Lee Seversky, Pierre Alliez, Joshua Levine, Andrei Sharf, and Claudio Silva. State of the Art in Surface Reconstruction from Point Clouds. In *Eurographics 2014 - State of the Art Reports*, volume 1 of *EUROGRAPHICS star report*, pages 161–185, Strasbourg, France, April 2014.

[BZ99] C. Baillard and Andrew Zisserman. Automatic reconstruction of piecewise planar models from multiple views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2559–2565, Ft. Collins, CO, USA, 1999.

[ÇAL+16] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, pages 424–432, Cham, 2016. Springer International Publishing.

[CBL+19] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[CCO15] C. Cabo, S. Garcia Cortes, and C. Ordonez. Mobile laser scanner data for automatic surface detection based on line arrangement. *Automation in Construction*, 58:28 – 37, 2015.

[CCR08] Paolo Cignoni, Massimiliano Corsini, and Guido Ranzuglia. MeshLab: an Open-Source 3D Mesh Processing System. *ERCIM News*, 2008(73), 2008.

[CDF+17] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D Data in Indoor Environments. *International Conference on 3D Vision (3DV)*, 2017.

[CFG+15] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository, 2015.

[CKY09] Sunglok Choi, Taemin Kim, and Wonpil Yu. Performance Evaluation of RANSAC Family. In *British Machine Vision Conference (BMVC 2009)*, 2009.

[CLF02] Umberto Castellani, Salvatore Livatino, and Robert B Fisher. Improving environment modelling by edge occlusion surface completion. In *1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT 2002)*, pages 672–675. IEEE, 2002.

[CLLW19] Po-Yi Chen, Alexander H Liu, Yen-Cheng Liu, and Yu-Chiang Frank Wang. Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2624–2632, 2019.

[CLP10] A. L. Chauve, P. Labatut, and J. P. Pons. Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1261–1268, June 2010.

[CMS98] Paolo Cignoni, Claudio Montani, and Roberto Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37–54, 1998.

[Com12] Wikimedia Commons. Villa le sextant, 2012. Licence: https://creativecommons.org/licenses/by-sa/3.0/deed.en.

[Com17] Wikimedia Commons. Villa le sextant vue de la rue, 2017. Licence: https://creativecommons.org/licenses/by-sa/3.0/deed.en.

[CSO+18] Ian Cherabier, Johannes L Schonberger, Martin R Oswald, Marc Pollefeys, and Andreas Geiger. Learning priors for semantic 3D reconstruction. In *IEEE European Conference on Computer Vision (ECCV)*, pages 314–330, 2018.

[CTLES17] Kai-Wei Chiang, Guang-Je Tsai, Yu-Hua Li, and Naser El-Sheimy. Development of lidar-based uav system for environment reconstruction. *IEEE Geoscience and Remote Sensing Letters*, 14(10):1790–1794, 2017.

[CTZ20] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. BSP-net: Generating compact meshes via binary space partitioning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 45–54, 2020.

[CXG+16] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *IEEE European Conference on Computer Vision (ECCV)*, pages 628–644. Springer, 2016.

[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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009.

[DN18] Angela Dai and Matthias Nießner. 3DMV: Joint 3D-multi-view prediction for 3D semantic scene segmentation. In *IEEE European Conference on Computer Vision (ECCV)*, pages 452–468, 2018.

[DRB+18] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. ScanComplete: Large-scale scene completion and semantic segmentation for 3D scans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4578–4587, 2018.

[EE11] Ali Elqursh and Ahmed Elgammal. Line-based relative pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3049–3056, June 2011.

[EKS83] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, July 1983.

[EPF14] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[FCSS09] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Manhattan-world stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1422–1429, June 2009.

[FH15] Yasutaka Furukawa and Carlos Hernández. Multi-view stereo: A tutorial. *Foundations and Trends in Computer Graphics and Vision (CGV 2015)*, 9(1-2):1–148, 2015.

[GF09] Aleksey Golovinskiy and T. Funkhouser. Min-cut based segmentation of point clouds. In *IEEE International Conference on Computer Vision (ICCV)*, pages 39 – 46, 11 2009.

[GFK+18] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3D surface generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 216–224, 2018.

[GHL+20] Vitor Guizilini, Rui Hou, Jie Li, Rares Ambrus, and Adrien Gaidon. Semantically-guided representation learning for self-supervised monocular depth. *arXiv preprint arXiv:2002.12319*, 2020.

[GLU12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[GvGJMR12] Rafael Grompone von Gioi, Jérémie Jakubowicz, Jean-Michel Morel, and Gregory Randall. LSD: a line segment detector. *Image Processing On Line (IPOL 2012)*, 2:35–55, 2012.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[HMB14] Manuel Hofer, Michael Maurer, and Horst Bischof. Improving sparse 3D models for man-made environments using line-based 3D reconstruction. In *2nd International Conference on 3D Vision (3DV 2014)*, volume 1, pages 535–542, December 2014.

[HMB16] Manuel Hofer, Michael Maurer, and Horst Bischof. Efficient 3D scene abstraction using line segments. *Computer Vision and Image Understanding (CVIU 2016)*, 3 2016.

[HMK+18] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[HNZ21] M.Q. Huang, J. Ninić, and Q.B. Zhang. Bim, machine learning and computer vision techniques in underground construction: Current status and future perspectives. *Tunnelling and Underground Space Technology*, 108:103677, 2021.

[Hof16] Manuel Hofer. Line3D++, 2016. https://github.com/manhofer/Line3Dpp.

[Hor74] Berthold KP Horn. Determining lightness from an image. *Computer graphics and image processing*, 3(4):277–299, 1974.

[HP16] Christian Häne and Marc Pollefeys. An overview of recent progress in volumetric semantic 3d reconstruction. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3294–3307. IEEE, 2016.

[HS12] Keisuke Hirose and Hideo Saito. Fast line description for line-based SLAM. In *British Machine Vision Conference (BMVC 2012)*, 2012.

[HSL+17] Timo Hackel, N. Savinov, L. Ladicky, Jan D. Wegner, K. Schindler, and M. Pollefeys. SEMANTIC3D.NET: A large-scale point cloud classification benchmark. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-1-W1:91–98, 2017.

[HWB13] Manuel Hofer, Andreas Wendel, and Horst Bischof. Incremental line-based 3D reconstruction using geometric constraints. In *British Machine Vision Conference (BMVC 2013)*, 2013.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[IB12] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *International Journal of Computer Vision (IJCV 2012)*, 97(2):123–147, Apr 2012.

[IS17] Naoto Ienaga and Hideo Saito. Reconstruction of 3D models consisting of line segments. In Chu-Song Chen, Jiwen Lu, and Kai-Kuang Ma, editors, *Asian Conference on Computer Vision workshops (ACCVw 2017)*, pages 100–113. Springer International Publishing, 2017.

[JKTS10] A. Jain, C. Kurz, T. Thormählen, and H. P. Seidel. Exploiting global connectivity constraints for reconstruction of 3D line segments from images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1586–1593, June 2010.

[JP11] Michal Jancosek and Tomás Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3121–3128, 2011.

[KAB+21] Florian Kluger, Hanno Ackermann, Eric Brachmann, Michael Ying Yang, and Bodo Rosenhahn. Cuboids Revisited: Learning Robust 3D Shape Fitting to Single RGB Images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13070–13079, 2021.

[KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *4th Eurographics Symposium on Geometry Processing (SGP 2006)*, pages 61–70, 2006.

[KGT19] Nilesh Kulkarni, Abhinav Gupta, and Shubham Tulsiani. Canonical surface mapping via geometric cycle consistency. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[KHM17] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *Advances in neural information processing systems*, pages 365–376, 2017.

[KKZ09] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM ransactions on Knowledge Discovery from Data (TKDD 2009)*, 3(1):1:1–1:58, March 2009.

[KPZK17] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (TOG)*, 36(4), 2017.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[KTEM18] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *IEEE European Conference on Computer Vision (ECCV)*, 2018.

[LA13] Florent Lafarge and Pierre Alliez. Surface reconstruction through point set structuring. *Computer Graphics Forum*, 32, 05 2013.

[Lan21] Pierre-Alain Langlois. Github repositories of Pierre-Alain Langlois, 2021. https://github.com/palanglois.

[LBM19] Pierre-Alain Langlois, Alexandre Boulch, and Renaud Marlet. Surface Reconstruction from 3D Line Segments. In *2019 International Conference on 3D Vision (3DV)*, pages 553–563, Québec City, Canada, September 2019. IEEE.

[LBS+18] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on X-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In Maureen C. Stone, editor, *SIGGRAPH*, pages 163–169. ACM, 1987.

[LFR+20] Pierre-Alain Langlois, Matthew Fisher, Bryan Russell, Oliver Wang, and Vladimir Kim. Reconstructing three-dimensional scenes using multi-view cycle projection, March 2020.

[LFW+21] Pierre-Alain Langlois, Matthew Fisher, Oliver Wang, Vladimir Kim, Alexandre Boulch, Renaud Marlet, and Bryan Russell. 3D Reconstruction By Parameterized Surface Mapping. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3273–3277, Anchorage, USA, September 2021.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *IEEE European Conference on Computer Vision (ECCV)*, 2014.

[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 *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[LPK09] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Robust and efficient surface reconstruction from range data. *Computer Graphics Forum (CGF 2009)*, 28(8):2275–2290, 2009.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[LSS+19] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Transactions on Graphics (TOG)*, 38(4):65:1–65:14, July 2019.

[LWC+11] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. GlobFit: Consistently fitting primitives by discovering global relations. In *ACM SIGGRAPH 2011*, pages 52:1–52:12, New York, NY, USA, 2011. ACM.

[LWR+19] Chen-Hsuan Lin, Oliver Wang, Bryan C Russell, Eli Shechtman, Vladimir G Kim, Matthew Fisher, and Simon Lucey. Photometric mesh optimization for video-aligned 3D object reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[MDR18] Pedro Miraldo, Tiago Dias, and Srikumar Ramalingam. A minimal closed-form solution for multi-perspective pose estimation using points and lines. In *IEEE European Conference on Computer Vision (ECCV)*, 2018.

[MF14] Luca Magri and Andrea Fusiello. T-linkage: A continuous relaxation of J-linkage for multi-model fitting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3954–3961, 2014.

[MG16] Gerhard Mentges and Rolf-Rainer Grigat. Surface reconstruction from image space adjacency of lines using breadth-first plane search. In *IEEE International Conference on Robotics and Automation (ICRA 2016)*, pages 995–1002, May 2016.

[MGMH04] Mukesh C Motwani, Mukesh C Gadiya, Rakhi C Motwani, and Frederick C Harris. Survey of image denoising techniques. In *GSPX*, volume 27, pages 27–30, 2004.

[MK10] Branislav Mičušík and Jana Košecká. Multi-view superpixel stereo in urban environments. *International Journal of Computer Vision (IJCV 2010)*, 89(1):106–119, August 2010.

[MMBM15] Aron Monszpart, Nicolas Mellado, Gabriel J. Brostow, and Niloy J. Mitra. RAPter: Rebuilding man-made scenes with regular arrangements of planes. *ACM Transactions on Graphics (TOG)*, 34(4):103:1–103:12, 2015.

[MON+19] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[MST+20] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *CoRR*, abs/2003.08934, 2020.

[MvAB+20] Zak Murez, Tarrence van As, James Bartolozzi, Ayan Sinha, Vijay Badrinarayanan, and Andrew Rabinovich. Atlas: End-to-End 3D Scene Reconstruction from Posed Images. In *IEEE European Conference on Computer Vision (ECCV)*, 2020.

[NF15] I. Nurutdinova and A. Fitzgibbon. Towards pointless Structure from Motion: 3D reconstruction and camera parameters from general 3D curves. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2363–2371, 2015.

[NL13] Anh Nguyen and Bac Le. 3D point cloud segmentation: A survey. In *6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 225–230, 2013.

[NW17] L. Nan and P. Wonka. Polyfit: Polygonal surface reconstruction from point clouds. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2372–2380, October 2017.

[PFS+19] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019.

[PH03] Emil Praun and Hugues Hoppe. Spherical parametrization and remeshing. In *ACM Transactions on Graphics (TOG)*, 2003.

[PNM+20] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *IEEE European Conference on Computer Vision (ECCV)*, 2020.

[QSMG17] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[RDG18] Xavier Roynard, Jean-Emmanuel Deschaud, and François Goulette. Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. *The International Journal of Robotics Research*, 37(6):545–557, 2018.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015.

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[Sal17] Yohann Salaün. LineSfM, 2017. https://github.com/ySalaun/LineSfM.

[SCD+06] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2006.

[SDK09] Ruwen Schnabel, Patrick Degener, and Reinhard Klein. Completion and reconstruction with primitive shapes. *Computer Graphics Forum (CGF 2009)*, 28(2):503–512, 2009.

[SF16] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[SMM16a] Yohann Salaün, Renaud Marlet, and Pascal Monasse. Multiscale line segment detector for robust and accurate SfM. In *23rd International Conference on Pattern Recognition (ICPR 2016)*, pages 2000–2005, December 2016.

[SMM16b] Yohann Salaün, Renaud Marlet, and Pascal Monasse. Robust and accurate line- and/or point-based pose estimation without Manhattan assumptions. In *IEEE European Conference on Computer Vision (ECCV)*, 2016.

[SMM17] Yohann Salaün, Renaud Marlet, and Pascal Monasse. Robust SfM with little image overlap. In *5th International Conference on 3D Vision (3DV 2017)*, Qingdao, China, October 2017.

[SRV+19] Srinath Sridhar, Davis Rempe, Julien Valentin, Sofien Bouaziz, and Leonidas J. Guibas. Multiview aggregation for learning category-specific shape reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[SSG+17] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[SSN09] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3D: Learning 3D scene structure from a single still image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):824–840, May 2009.

[SSS09] Sudipta N. Sinha, Drew Steedly, and Richard Szeliski. Piecewise planar stereo for image-based rendering. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1881–1888, September 2009.

[STH+19] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. DeepVoxels: Learning persistent 3D feature embeddings. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[STO15] Takayuki Sugiura, Akihiko Torii, and Masatoshi Okutomi. 3D surface reconstruction from point-and-line cloud. In *International Conference on 3D Vision (3DV 2015)*, pages 264–272, October 2015.

[SWK07] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum (CGF 2007)*, 26(2):214–226, June 2007.

[SYZ+17] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1746–1754, 2017.

[SZ97] C. Schmid and A. Zisserman. Automatic line matching across views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 666–671, Washington, DC, USA, June 1997.

[TB15] Charles Thomson and Jan Boehm. Automatic geometry generation from point clouds for BIM. *Remote Sensing*, 7(9):11753–11775, 2015.

[TDB17] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[TF08] Roberto Toldo and Andrea Fusiello. Robust multiple structures estimation with J-linkage. In *IEEE European Conference on Computer Vision (ECCV)*, pages 537–547, 2008.

[TQD+19] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPconv: Flexible and deformable convolution for point clouds. In *IEEE International Conference on Computer Vision (ICCV)*, pages 6411–6420, 2019.

[TRR+19] Maxim Tatarchenko, Stephan Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3D reconstruction networks learn? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[TXFY18] Chunwei Tian, Yong Xu, Lunke Fei, and Ke Yan. Deep learning for image denoising: a survey. In *International Conference on Genetic and Evolutionary Computing*, pages 563–572. Springer, 2018.

[ULW19] Kaleem Ullah, Irene Lill, and Emlyn Witt. An Overview of BIM Adoption in the Construction Industry: Benefits and Barriers. In *10th Nordic Conference on Construction Economics and Organization*, 05 2019.

[vKvLV11] M. van Kreveld, T. van Lankveld, and R. C. Veltkamp. On the shape of a set of points and lines in the plane. *Computer Graphics Forum (CGF 2011)*, 30(5):1553–1562, 2011.

[vKvLV13] M. van Kreveld, T. van Lankveld, and R. C. Veltkamp. Watertight scenes from urban LiDAR and planar surfaces. *Computer Graphics Forum (CGF 2013)*, 32(5):217–228, 2013.

[VLA15] Yannick Verdie, Florent Lafarge, and Pierre Alliez. LOD Generation for Urban Scenes. *ACM Transactions on Graphics (TOG)*, 34(3):15, 2015.

[WM14] Jonas Witt and Gerhard Mentges. Maximally informative surface reconstruction from lines. In *IEEE International Conference on Robotics and Automation (ICRA 2014)*, pages 2029–2036, May 2014.

[WQF19] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3D point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9621–9630, 2019.

[WWH09] Zhiheng Wang, Fuchao Wu, and Zhanyi Hu. MSLD: A robust descriptor for line matching. *Pattern Recognition (PR 2009)*, 42(5):941–953, 2009.

[WWW11] Min Wan, Yu Wang, and Desheng Wang. Variational surface reconstruction based on Delaunay triangulation and graph cut. *International Journal for Numerical Methods in Engineering*, 85(2):206–229, 2011.

[WZL+18] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. In *IEEE European Conference on Computer Vision (ECCV)*, 2018.

[WZLF19] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2Mesh++: Multi-view 3D mesh generation via deformation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1042–1051, 2019.

[WZW+20] Lijun Wang, Jianming Zhang, Oliver Wang, Zhe Lin, and Huchuan Lu. Sdc-depth: Semantic divide-and-conquer network for monocular depth estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 541–550, 2020.

[XQL+19] Yang Xiao, Xuchong Qiu, Pierre-Alain Langlois, Mathieu Aubry, and Renaud Marlet. Pose from shape: Deep pose estimation for arbitrary 3D objects. In *British Machine Vision Conference (BMVC)*, 2019.

[ZBKB08] Lukas Zebedin, Joachim Bauer, Konrad F. Karner, and Horst Bischof. Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In *IEEE European Conference on Computer Vision (ECCV)*, pages 873–886, Marseille, France, October 2008.

[ZCX+19] Zhenyu Zhang, Zhen Cui, Chunyan Xu, Yan Yan, Nicu Sebe, and Jian Yang. Pattern-affinitive propagation across depth, surface normal and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4106–4115, 2019.

[ZK14] Lilian Zhang and Reinhard Koch. Structure and motion from line correspondences: Representation, projection, initialization and sparse bundle adjustment. *Journal of Visual Communication and Image Representation (JVCIR 2014)*, 25(5):904–915, 2014.

[ZKM05] Marco Zuliani, Charles S. Kenney, and B. S. Manjunath. The multiRANSAC algorithm and its application to detect planar homographies. In *IEEE International Conference on Image Processing (ICIP 2005)*, pages 153–156. IEEE, September 2005.

[ZPK+02] Yan Zhang, J Paik, Andreas Koschan, Mongi A Abidi, and David Gorsich. Simple and efficient algorithm for part decomposition of 3-D triangulated models based on curvature analysis. In *Proceedings. International Conference on Image Processing*, volume 3, pages III–III. IEEE, 2002.

[ZRK12] Aamer Zaheer, Maheen Rashid, and Sohaib Khan. Shape from angle regularity. In *IEEE European Conference on Computer Vision (ECCV)*, pages 1–14, Florence, Italy, October 2012.

[ZTCS99] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape-from-shading: a survey. *IEEE transactions on pattern analysis and machine intelligence*, 21(8):690–706, 1999.

[ZTF+18] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018.

[ZYL11] Y. Zhang, Heng Yang, and Xiaolin Liu. A line matching method based on local and global appearance. In *4th International Congress on Image and Signal Processing (ICISP 2011)*, pages 1381–1385, October 2011.

Part VI

APPENDIX

13

Additional visualizations for part III

We provide additional qualitative comparisons between our reconstruction method and XNOCS [SRV+19] on Figure 13.1 and Figure 13.2.

Figure 13.1: Additional qualitative results for our reconstruction method by parameterized surface mapping. 1st column: ground truth; 2nd column: ours; 3rd column: XNOCS [SRV+19]

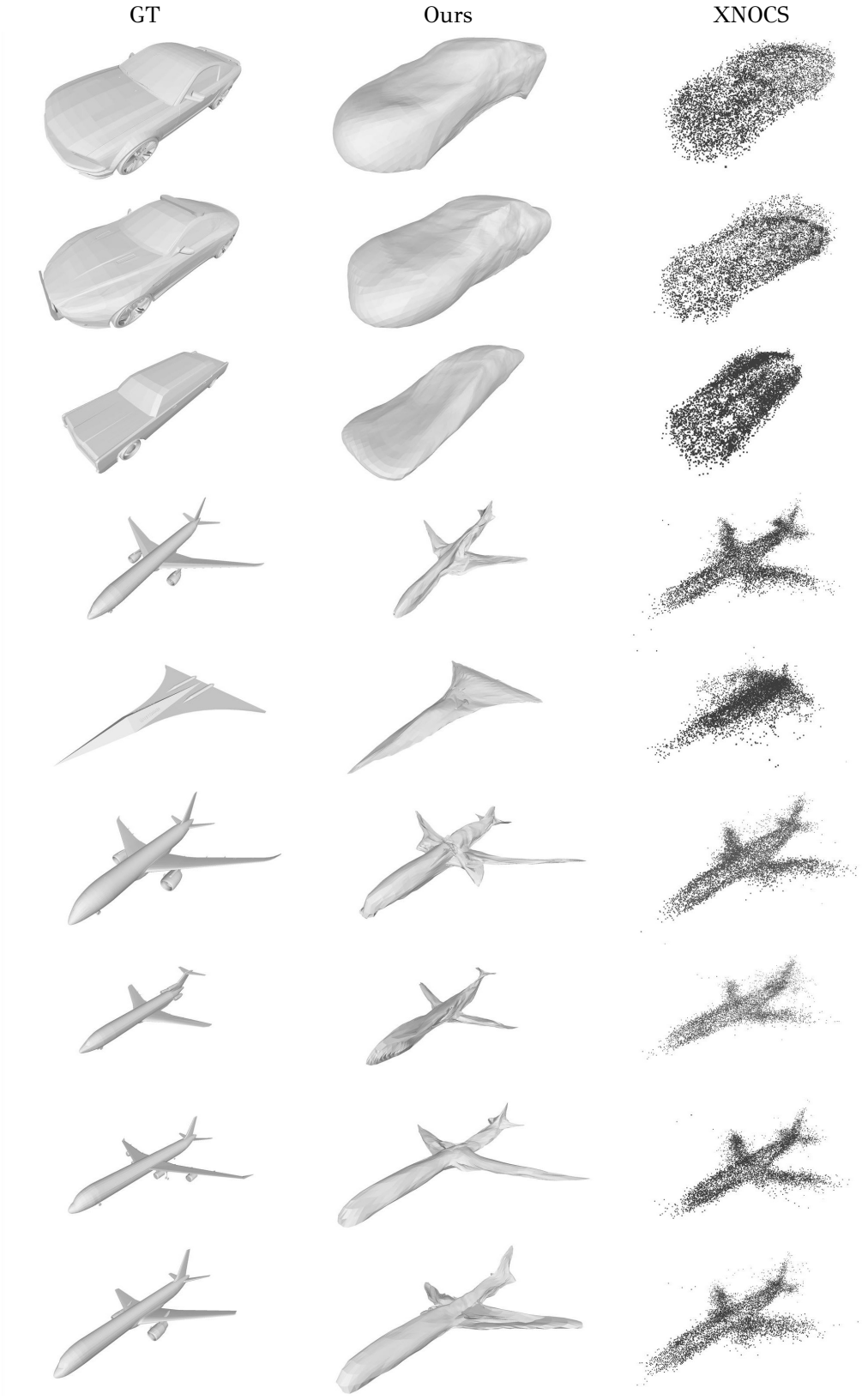


Figure 13.2: Additional qualitative results for our reconstruction method by parameterized surface mapping. 1st column: ground truth; 2nd column: ours; 3rd column: XNOCS [SRV+19]