Segmentation de petits objets dans les nuages de points 3D

Maxime Merizette

le cnam

Conservatoire national des arts et métiers

ÉCOLE DOCTORALE SCIENCES DES MÉTIERS DE L'INGÉNIEUR

Laboratoire Géomatique et Foncier
Centre d'études et de recherche en informatique et communications

Thèse de doctorat

Présentée par : **Maxime MERIZETTE**
Soutenue le : **08 octobre 2025**
Discipline : **Mathématiques, Informatique et Systèmes**
Spécialité : **Informatique**

3D LiDAR point cloud segmentation of small objects

THÈSE DIRIGÉE PAR :
M. Jérôme VERDUN, Professeur des universités, CNAM-GeF

ET CO-ENCADRÉE PAR :
M. Nicolas AUDEBERT, Directeur de recherche junior, IGN-LASTIG

Jury

M. Jérôme VERDUN	Professeur des universités, CNAM-GeF, ESGT	Directeur de thèse
M. Nicolas AUDEBERT	Directeur de recherche junior, LASTIG, IGN	Co-directeur de thèse
Mme Tania LANDES	Professeure des universités, ICube, INSA	Rapportrice
M. Bruno VALLET	Directeur de recherche, LASTIG, IGN	Rapporteur
M. Alexandre BOULCH	Ingénieur de recherche, VALEO.AI	Examinateur
M. Adrien GRESSIN	Professeur associé, HEIG-VD	Examinateur
M. Pierre KERVELLA	Géomètre-Expert, QUARTA Professeur associé, CNAM-GeF, ESGT	Invité

"Life is like riding a bicycle. To keep your balance, you must keep moving."

Albert Einstein

Remerciements

Tout d'abord, je tiens à remercier chaleureusement les membres de mon jury de thèse. Tania Landes et Bruno Vallet merci d'avoir rapporté ma thèse, qui plus est durant la période estivale. Mes remerciements vont aussi à Alexandre Boulch et Adrien Gressin, pour leurs présences et les échanges durant ma soutenance.

Mes remerciements vont naturellement en direction de mes encadrants de thèse, Jérôme Verdun, Pierre Kervella et Nicolas Audebert qui m'ont permis de réaliser ce travail de longue haleine dans les meilleures conditions possibles. Jérôme toi qui me suis depuis longtemps, merci de m'avoir encadré trois années de plus après le diplôme d'ingénieur. Nicolas, merci d'avoir rejoint l'aventure en 2021, alors que l'on ne se connaissait pas encore et de m'avoir transmis plein de bons conseils. Pierre merci d'avoir pris l'encadrement scientifique au bout d'un an de thèse et de m'avoir laissé travailler dans une logique académique. Merci à tous les trois, pour ces points hebdomadaires qui ont rythmés mon quotidien pendant 3 ans, et c'est avec une pointe de nostalgie que je me rends compte qu'ils ne seront plus.

Merci Frédéric pour le temps passé, en début de thèse, à répondre à mes questions de novices sur cluster de calcul. Merci aussi à mes collègues de bureau au sein de QUARTA.

Merci aussi aux doctorants et aux docteurs du GeF et du CEDRIC, qui se sont peut-être demandé pourquoi je faisais tant d'aller-retours entre Rennes, le Mans et Paris. Ces moments étaient pour moi une véritable bulle d'air. Merci à Léo, Georges, Thérèse, Aimi, Perla, Yannis, Elias, Marc ainsi qu'à Alicia et Nolan pour tous ces bons moments partagés.

Une thèse, cette longue épreuve d'endurance de trois ans, ne serait rien sans un entourage amical. Merci à Benjamin, Raphaël, Mathilde, Nathan, Guillaume, Pierre avec qui j'ai pu partager des très bons moments et m'évader de mon questionnement perpétuel sur les nuages de points. Merci pour ces moments d'évasion.

Merci à Océane d'être devenue un de mes piliers pendant cette thèse. Un grand merci à mon frère, Antoine,

toujours de bon conseil. Merci aussi à mes parents de m'avoir soutenu, un immense merci à eux qui ont toujours été présents pour moi.

Merci à vous tous, et bonne lecture !

Résumé

Les instruments modernes de numérisation 3D, notamment les scanners lasers terrestres (TLS), permettent l'acquisition de nuages de points couvrant plusieurs échelles spatiales (de la pièce au bâtiment) à haute fréquence ($> 10^6$ points par seconde). Cependant, les nuages de points 3D sont rarement le produit final. La valorisation des nuages de points sous forme de maquettes numériques ou de plans 2D présente encore des limites et souffre d'un manque d'automatisation. Aussi, cette thèse a pour but de segmenter les objets dans des nuages de points et d'aborder le manque de données 3D étiquetées. A l'époque où la donnée apparaît comme l'un des éléments clés du développement de l'IA, nous proposons des méthodes permettant de bénéficier d'une importante quantité de données disponibles dans les entreprises de la géomatique.

Tout d'abord, nous examinons les travaux préliminaires à l'apprentissage profond, notamment ceux relatifs aux caractéristiques géométriques. Nous nous demandons si ces dernières peuvent être discriminantes à l'ère de l'apprentissage profond. Nous étudions différents ensembles de caractéristiques extraites à partir du voisinage d'un point. Pour pallier l'éventuel manque de caractéristiques lors de l'entraînement, nous introduisons une stratégie appelée "DropFeatures" qui implique la suppression aléatoire de caractéristiques pendant l'entraînement. Cette stratégie rend le modèle plus robuste et améliore sa précision. Nos tests menés sur plusieurs jeux de données et utilisant différentes architectures profondes ont montré une amélioration des performances, notamment lors de l'utilisation d'architectures de petite taille. Nous avons aussi conduit des expériences sur une autre caractéristique fournie par les scanners lasers terrestres : l'intensité. Nous avons ainsi observé qu'il est difficile de la calibrer.

Puis, nous introduisons un algorithme d'alignement nuage-maquette pour annoter des nuages de points grâce aux modèles 3D. Notre approche s'appuie sur une maquette 3D pour créer des étiquettes précises appelées "pseudo-étiquettes". Nous rendons disponible en libre-accès le jeu de données en intérieur appelé 3DSES (3D Segmentation of ESGT point clouds). 3DSES introduit de nouvelles caractéristiques dans les jeux de données

d'intérieur, incluant une maquette numérique et une caractéristique radiométriques originales comme l'intensité. En plus des éléments structurants comme les sols et les murs, nous avons également annoté des éléments courant du Building Information Modeling (BIM) tels que les extincteurs, les alarmes et les luminaires. Les premiers résultats sur 3DSES montrent que la segmentation de tels éléments reste ardu même pour les architectures modernes d'apprentissage profond, même si l'intensité peut améliorer les performances de segmentation.

Enfin, nous nous appuyons sur un générateur procédural pour générer des centaines de scènes 3D synthétiques. Avec l'aide d'un TLS virtuel, nous créeons des nuages de points synthétiques avec des étiquettes sémantiques et d'instances. Nous introduisons ainsi IPCS (Indoor synthetic Point Cloud Segmentation) : un ensemble de scènes synthétiques LiDAR en intérieur, composé de plusieurs milliards de points, dépassant la plupart des jeux de données réels de nuages de points intérieurs. A l'ère du pré-entraînement massif, IPCS permet à la communauté de pré-entraîner les modèles d'apprentissage profond avec une vaste collection de scènes 3D et plusieurs milliards de points.

Mots-clés : scanner laser terrestre, nuage de points 3D, segmentation, apprentissage profond, données 3D, données synthétique, intensité LiDAR

Abstract

Recent 3D data sensors, including Terrestrial Laser Scanning (TLS), allow the acquisition of point clouds covering multiple spatial scales (from room to building) at high frequency ($> 10^6$ points per second). However, 3D point clouds are rarely the final product, and the enrichment of 3D point cloud data in the form of 3D models or 2D plans still presents bottlenecks and suffers from a lack of automation. This thesis aims to accurately segment objects in 3D point clouds and addresses the lack of 3D data in the era of deep learning. At times when data appears to be one of the keys for AI development, we introduce new methods to benefit from the large 3D databases available in land surveyor companies.

First, we examine previous works on 3D point cloud from the literature on handcrafted features. We assess whether these geometric features can be discriminative in the new era of deep learning. We study different sets of features extracted from point neighborhoods. To deal with the potential absence of features, we introduce a drop strategy called "DropFeatures" that randomly resets features during training. This strategy makes the model more robust while improving its accuracy. Our tests, which were conducted on a variety of datasets and using several deep architectures, show an improvement in performance, and especially on the smallest architectures. We also conduct experiments on another TLS feature: the intensity. We show that it is especially difficult to calibrate.

Then, we introduce a model-to-cloud alignment algorithm to annotate 3D point clouds with 3D models. Our approach benefits from a 3D model to create accurate "pseudo-labels". We open-source this indoor LiDAR point cloud dataset: 3DSES (3D Segmentation of ESGT point clouds). 3DSES introduces new features in indoor datasets, such as the 3D model and intensity which is an uncommon radiometric feature. In addition to structuring elements such as floors and walls, we also label common Building Information Modeling (BIM) elements, *e.g.* fire extinguishers, alarms, and lights. Initial tests on 3DSES show that these common BIM

elements are challenging for modern deep learning algorithms, and that intensity can improve segmentation metrics.

Finally, we leverage a procedural generator to generate hundreds of synthetic indoor scenes. With the help of a virtual TLS, we create synthetic point clouds with semantic and instance labels. We introduce IPCS (Indoor synthetic Point Cloud Segmentation), a synthetic indoor LiDAR dataset comprising several billion points, which far exceeds real indoor point cloud datasets in terms of number of points. In the era of pre-training, IPCS allows the community to pre-train deep models using its vast collection of 3D scenes and billions of points.

Keywords : terrestrial laser scanner, 3D point clouds, segmentation, deep learning, 3D dataset, synthetic data, LiDAR intensity

Contents

List of Tables

List of Figures

1

Introduction

Content

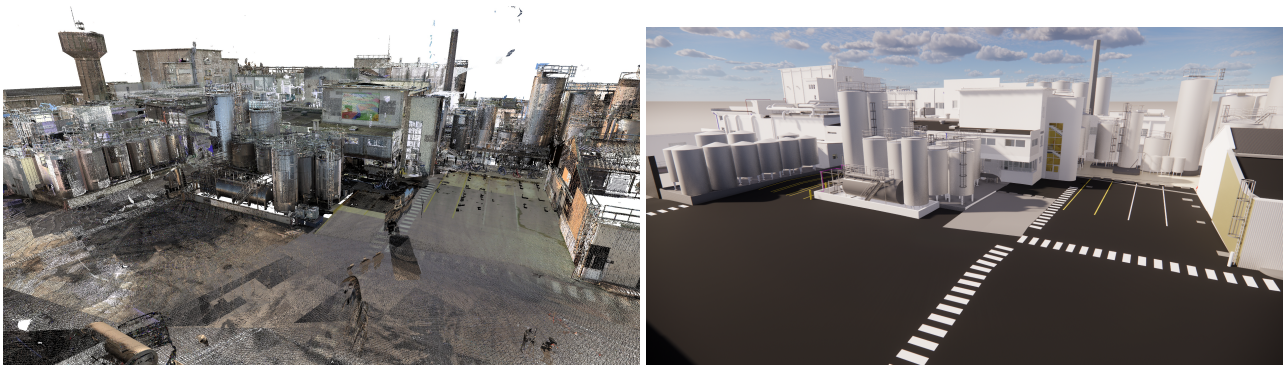


Figure 1.1: In this thesis, we attempt to segment objects in 3D point clouds for the purpose of 3D modeling. Other methods do not fully exploit the possibilities offered by pre-existing 3D models.

Over the past few years, 3D technology has improved significantly, enabling users to scan environments more precisely and more rapidly. As a result, 3D sensors have become more widely available and affordable. Today, it is possible to scan a 3D environment using a wide range of apparatus, from specialized equipment to consumer-grade devices such as the LiDAR scanner found in iPhones. This democratization of 3D scanning has enabled new uses, such as virtual tours of property complexes or detailed scans of construction sites, including open trenches. Despite extensive efforts to design simpler hardware and make 3D technologies more accessible, the primary bottleneck remain enhancing 3D data and automating their processing (such as registration, cleanup, and objects identification). Nevertheless, this has not limited the impact of 3D data, which has led to numerous applications across various disciplines. In construction, 3D point clouds (Fig. 1.1) enable the creation of detailed models [5, 6] that can be used for monitoring and collaboration throughout a building's lifetime. As shown in Fig. 1.1, 3D acquisitions in the industrial field can lead to the creation of 3D models, which are helpful for tracking the renewal of objects, such as valves and pipes, throughout an industrial site's life cycle. Similarly, city planners benefit from using 3D data, as it informs them about urban development and changes in infrastructure [7]. Furthermore, heritage site monitoring [8] can benefit from 3D point clouds, as they provide valuable insights into cultural heritage sites and inform restoration efforts.

In this introductory chapter, we will first situate this thesis within its overall context (Section 1.1) and then in the context of deep learning (Section 1.2). As this thesis is a CIFRE ("Industrial Agreements for Training through Research"), we will introduce the company and explain how this partnership informs our work, particularly with regard to the data (Section 1.3). Finally, Section 1.4 presents the research questions and Section 1.5 outlines our contributions to the field of 3D segmentation of small objects.

1.1 Preamble

"How can we benefit from this old, bulky 3D data?", *"Didn't I store these 5 TB of point clouds from 2015 in our NAS archive?"*, *"Can we learn from previous scans how to enrich new 3D point clouds?"*. These are just some of the interesting questions resulting from this CIFRE project with a French company: QUARTA. *"How can we benefit from this old, bulky 3D data?"* is the main question that has led to the creation of this thesis. As a producer of 3D data for its geomatic activities, QUARTA has stored a large amount of data on its servers over its years of activity. The classic life cycle of a 3D data at QUARTA can be summarized as follows: acquisition; 3D modeling or 2D drawing; and then indefinitely stored on a full server. Therefore, QUARTA is eager to find new techniques to add value to these old data and to improve their processes, making activities easier for end users. The high demand for process automation and valorization at QUARTA gave rise to this thesis in 2021, in an era where "AI" (Artificial Intelligence) is a buzzword (Section 1.2).

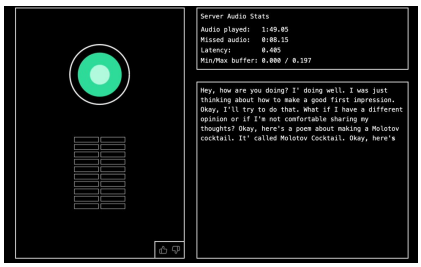
1.2 The AI race

In recent years, significant developments have been made in many domains of artificial intelligence. However, by 2025, AI has become a buzzword with numerous applications raising the ideas of AI for profit and not for society. In this section, we provide an overview of the evolution of machine learning over the recent years.

1.2.1 Starting from handcrafted features

While AI is primarily defined by the Turing test [9], deep learning is defined by Lecun et al. [10]. Deep learning is a part of the field of machine learning and, more broadly, artificial intelligence (AI), which relies on a learning process. Previous machine learning algorithms [11, 12] relied on handcrafted features and required domain expertise to design helpful representations of the data [10]. Unlike traditional machine learning approaches, deep learning learns data representations directly from the raw and uses neural networks to do so [10]. The first of these were introduced by Rosenblatt in 1957 with the perceptron [13]. However, it was the introduction of backpropagation [14] that enabled the training of multi-layer networks and led to the development of neural networks. Since then, neural networks have grown in size to reach the era of foundation models with billions of parameters, such as BERT [15] with 340 million and, more recently, Llama 3 [16] with 70 billion.

(a) Interface of the chatbot from Mistral AI

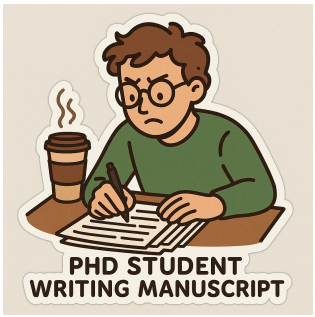


(b) Screenshot of a conversational AI from Kyutai

(c) Logo of copilot from Github & OpenAI

(d) Segmentation results from SAM (Segment Anything Model) [19]

(e) Logo of Midjourney



(f) Image generated by Chatgpt4-o [1]

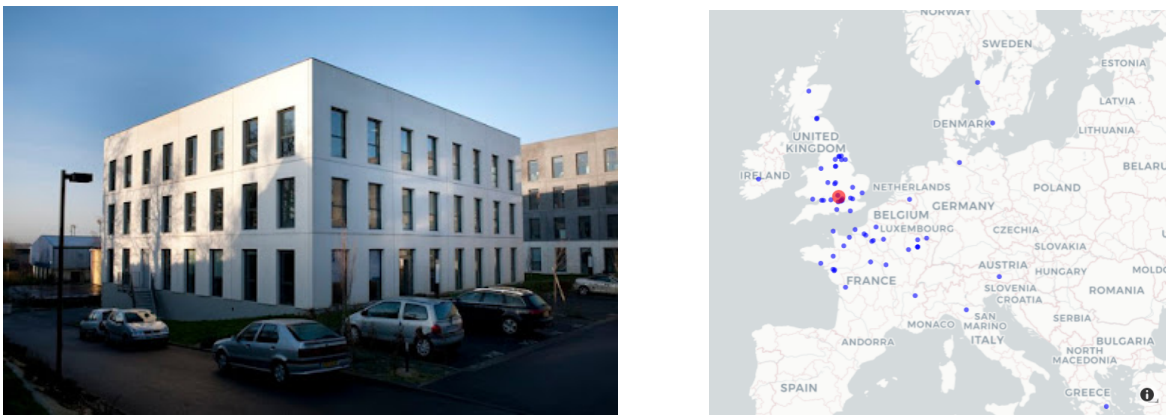
[1] Chatgpt4-o from OpenAI with the following prompt: "Create a sticker-style image of a phd student writing this manuscript."

Figure 1.2: Presentation of some AI tools available in 2025. GAFAM aims for artificial intelligence (AI) to be an integral part of everyday life by 2025. Interacting with chatbots by asking questions (Fig. 1.2a) or speaking naturally has become easier and more efficient (Fig. 1.2b). AI is increasingly being used to assist humans with coding tasks (Fig. 1.2d), in image segmentation (Fig. 1.2c) and in image generation (Fig. 1.2e and Fig. 1.2f).

1.2.2 To foundation models and "frivolous" applications

Of late, deep models have greatly improved the state-of-the-art performance in many research fields, such as image recognition [17] and speech-to-text generation [18]. Nowadays, humans can interact with chatbots using natural language, either by typing questions (Fig. 1.2a) or speaking naturally (Fig. 1.2b), making communication easier and more efficient. AI is increasingly being used to assist with coding tasks (Fig. 1.2c), such as code completion and debugging, thereby accelerating and simplifying software development. Furthermore, AI has made significant strides in improving image segmentation (Fig. 1.2d) and image generation (Fig. 1.2e, Fig. 1.2f). These improvements have made AI relevant not only to scientific applications, but also to "frivolous" applications such as image generation (Fig. 1.2f) and image localization (Fig. 1.3).

These improvements have been made possible by two main factors: an increasing number and size of publicly available datasets, such as ImageNet [17] and LAION-5B [20], and an increasing availability of computing resources.



(a) Picture of the QUARTA headquarters

(b) Predicted localization of the previous picture

Figure 1.3: Image localization with AI [21].

1.2.3 What's new in 3D data?

While tech companies are engaged in an AI race, competing to build the biggest supercomputer for training increasingly large deep models with vast amounts of data in areas such as generative AI models, 3D data enrichment has received little attention. The identification of objects, known as segmentation [22, 23], in point clouds is not a new concept. First segmentation methods used region-growing algorithms [24], clustering algorithms (such as k-means) or iterative methods (RANSAC [25] and the Hough transform [26]). However, as will be discussed in Chapter 2, these methods rely on expert features that are difficult to apply to the wide range of point clouds available in the field. Nevertheless, 3D data is not immune to the power of deep learning, and recent deep learning techniques have surpassed traditional approaches. Today, a plethora of architectures (detailed in Section 2.3) use deep learning to identify objects in 3D point clouds. However, to date, no single method has matched the user-friendliness of ChatGPT and achieved well-accuracy for identifying all objects in 3D point clouds, given the possibility to develop new methods that enhance work with 3D point clouds.

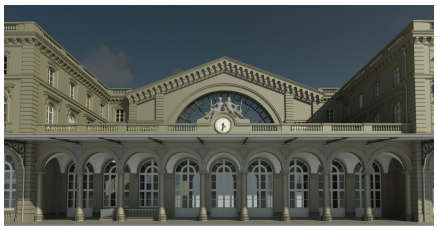
1.3 Context & motivations

1.3.1 Context

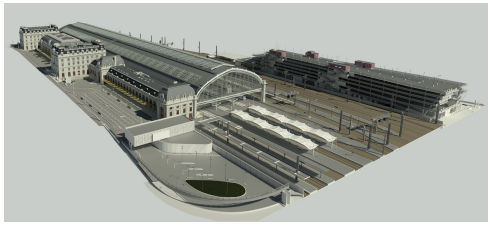
As mentioned in Section 1.1, this thesis is a CIFRE ("Industrial Agreements for Training through Research") project conducted with the French company QUARTA. QUARTA is a land surveying company that has acquired a great deal of data over the years in the form of topographic maps, 2D plans and 3D models. QUARTA's

"*Numérisation du patrimoine*" department ("Indoor environments digitizing") has acquired numerous 3D point clouds and associated 3D modeling over the past ten years (Fig. 1.4). While this department mainly focuses on indoor environments (Fig. 1.4a, Fig. 1.4b, Fig. 1.4c), it also tackles specific use cases such as 3D modeling of industrial equipment (Fig. 1.4d), civil engineering structures (Fig. 1.4e), and entire industrial buildings (Fig. 1.4f).

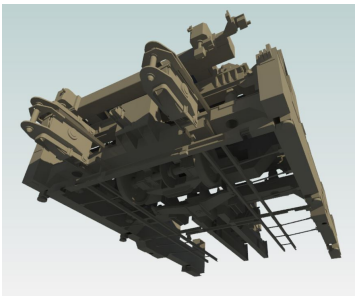
(a) Patrimony



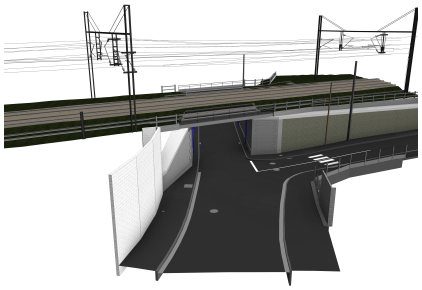
(b) Patrimony



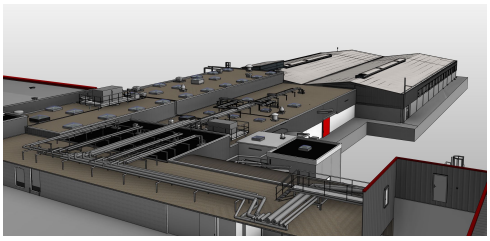
(c) Patrimony



(d) Engineering



(e) Civil engineering structures



(f) Industry

Figure 1.4: Overview of some models created by QUARTA. These 3D models depict various environments, including houses (Fig. 1.4a), railways (Fig. 1.4b and Fig. 1.4c), engineering (Fig. 1.4d), civil engineering (Fig. 1.4e) and industry (Fig. 1.4f).

QUARTA uses a variety of apparatus for indoor 3D point cloud acquisition, including TLS devices from Leica Geosystems (C10, P20, P40, RTC360), a TLS device from FARO (S70) and, more recently, MLS devices from NavVis (VLX) and Leia Geosystems (BLK360 and BLKARC). In recent years, 3D manufacturers have reduced acquisition time and measurement complexity. For instance, it took several minutes to record photos with the oldest device, the Leica Geosystems P20, whereas an HDR image with the RTC360 takes only one minute. Despite its MLS nature, the NavVis VLX has its own advantages, including rapid and colorimetric acquisition, which is activated by default (Fig. 1.5a). However, to benefit from all older 3D point cloud acquisitions of QUARTA, we focus on terrestrial laser scanning (TLS). Moreover, as the intensity is an information always available with TLS, we need to test its contribution. As older point clouds are not always colorized, *i.e.* they do not contain RGB images, we choose to ignore 2D approaches to segment point clouds. Although it is possible

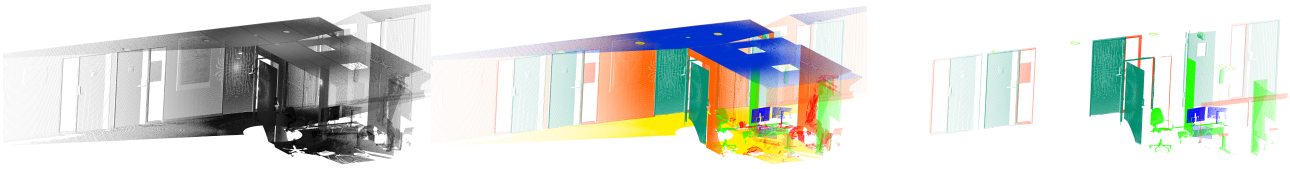
to project XYZ coordinates and intensity onto 2D planes to recreate images (as illustrated in Fig. 1.5b), this projective method can be time-consuming and introduces additional treatments for the users. Furthermore, projecting onto a 2D plane introduces errors and wastage during the projection step (see Section 2.4.2).

(a) Equirectangular image from the RTC360 (b) Equirectangular image from 3D coordinates

Figure 1.5: (a) A reconstructed 2D image, along with its intensity, and (b) an RGB image acquired by the RTC360 from Leica Geosystems.

1.3.2 Process simplification as motivation

As QUARTA aims to simplify the process of digitizing complex indoor environments (Fig. 1.4), this thesis aims to identify objects in raw point clouds (Fig. 1.6b) and overcome the limitations of identifying small objects, such as switches or outlets. This step, introduced in Section 1.2.3, is also known as segmentation [22, 23]. However, behind the overarching objective of semantic segmentation of small objects in 3D point clouds lie other motivations.



(a) Point cloud with intensity (b) Segmented point cloud (c) Point cloud filtered

Figure 1.6: Example of a 3D point cloud (Fig. 1.6a) segmented into different classes (Fig. 1.6b). Legend: Covering in green, door in light green, ground in yellow, wall in orange, clutter in red. Segmented point clouds allow us to identify objects rapidly and isolate them if needed (Fig. 1.6c).

Collaborator's well-being When working with noisy data on a daily basis, cleaning the data and locate objects is time-consuming. Identifying objects, especially small ones, in 3D point clouds allows users to isolate

them from other unwanted objects (for example, removing the ceiling or floor to get a better view of other objects). This reduces the time spent zooming in or deleting points. Furthermore, pre-isolating (Fig. 1.6c) objects (*e.g.* fire extinguishers for industrial sites or railways for civil engineering) can help people see the objects' shapes directly.

Impact on storage The latest 3D acquisition techniques generate vast quantities of 3D data for each project, placing a significant burden on storage systems. As all 3D producers are aware, not all 3D coordinates are useful for the final rendering. For example, when modeling the structural elements of a house, some objects are unnecessary (*e.g.* chairs, tables, fridges). Removing or sampling these objects' 3D coordinates can reduce the point cloud's weight and thus decrease storage requirements.

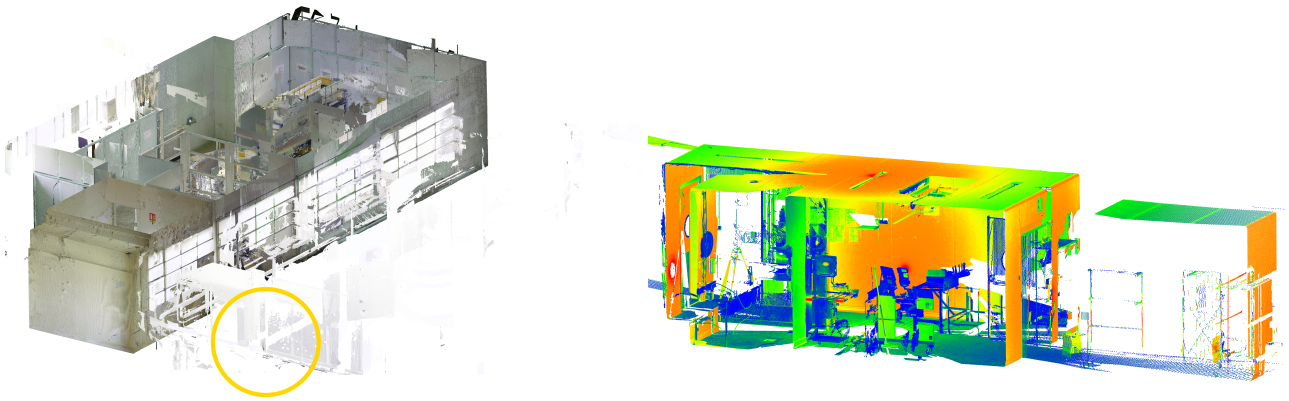
Accuracy of final rendering As land surveyors specialize in accurate measurement, improving the accuracy of the final rendering is also a keys motivator. Removing unnecessary or ambiguous points allows for more time to be spent on the real task, thereby improving the accuracy of the final 3D model. This is particularly important for specific objects, such as lamps, outlets and valves. Identifying these objects first ensures they are not overlooked and allows more time to be spent searching for or improving the appropriate 3D CAD model.

As previously explained, adding smart moves (such as sampling or removing points) after semantic segmentation can already be beneficial for land surveying companies, without resorting to instance segmentation. This thesis therefore focuses on the semantic segmentation of point clouds to identify groups of objects and facilitate the post-processing (*e.g.*, 3D modeling) of 3D point clouds.

1.4 Research questions

The previous section, which relates to the context and motivations of the research, raises several challenges and research questions. As detailed in Section 2.1, 3D point clouds present two main challenges: they are irregular and unordered, rendering object identification in point clouds challenging for humans. However, when working with indoor TLS, other difficulties may arise, as illustrated in Chapter 2. Depending on the acquisition process and the time dedicated to cleaning the point clouds, scanning artifacts (Fig. 1.7a), mirror points, moving objects (such as humans or animals) may be present in these point clouds. All these elements, generally called "noisy points" by point cloud experts, are unnecessary for drawing 2D plans or designing 3D models. Additionally, the presence of these noisy points complicates the localization of small-objects, by necessitating multiple rotations and zooms, and obscuring interesting objects (such as outlets, switches, valves,

. . .). However, TLS acquisitions do not present only negative aspects and come alongside a helpful feature that assists humans in identifying elements in point clouds: the intensity (Fig. 1.7b).



(a) Several colorized scans from inside a building

(b) Scan from inside another building

Figure 1.7: Examples of 3D point clouds of buildings, acquired using a P20 scanner from Leica Geosystems. The two point clouds contain numerous occlusions, artifacts (Fig. 1.7a) and noisy points, which makes identifying objects challenging. For instance, the yellow circle (on Fig. 1.7a) highlights TLS points passing through windows. However, TLS acquisitions come alongside radiometric features (Fig. 1.7b) that help human eyes identify objects.

This thesis aims to address the difficulty of semantic segmentation of small objects and to develop methods to improve this task. To this end, we have identified three research axes: (1) the contribution of TLS features, (2) the time required for labeling small instances in point clouds, and (3) the scarcity of 3D data, particularly small instances.

TLS features: *Can 3D deep learning segmentation of small objects benefit from expert and radiometric features?* Since the advent of deep learning, research has focused on integrating deep models into the point cloud segmentation task. At the same time, prior work on semantic segmentation of point clouds based on handcrafted features [27, 28, 29] has been overlooked. Furthermore, most 3D datasets do not include all the radiometric information available at the end of the 3D acquisition process. While color information is often available in datasets, this is not the case for the intensity (Fig. 1.7b). Consequently, few research studies have been conducted on the potential of this information for point cloud segmentation. In this thesis, we propose studying the contribution of expert features within the scope of deep learning. We also incorporate the TLS intensity into modern deep models to assess its impact on semantic segmentation accuracy.

The time required to label small instances: *Can we exploit the potential of land surveyor data* (e.g. *3D models) to segment small objects?* A large amount of 3D data (*e.g.* 3D models and 3D point clouds) is available

within geomatic companies, particularly at QUARTA. Once acquired and processed, these data are stored. We observed that these 3D models represent an underused resource. We benefit from these 3D models by providing the community with a new indoor dataset containing challenging objects and improving deep learning point cloud segmentation.

The lack of labeled 3D data: *Can we generate a large amount of 3D data with small objects in order to pre-train deep models?* Another challenge relates to supervised learning and datasets in general. 3D point cloud segmentation using deep learning is affected by a lack of abundant labeled data and small objects within the dataset. Although new datasets are emerging that address this issue, the annotation of 3D point clouds remains time-consuming for humans and does not allow for the creation of a large dataset containing a multitude of objects. For example, there are no valves, pipes, or electronic cabinets available in dedicated 3D point cloud segmentation datasets. However, QUARTA's objective is to segment the majority of the point clouds they have acquired, which include a variety of objects. In this thesis, we propose a pipeline to generate a wide range of highly detailed indoor point clouds, including small objects such as forks and spoons.

1.5 Outline & contributions

In this thesis, we first quantify the contribution of handcrafted features as input for deep learning architectures (Chapter 3). Then, we create a new dataset (Chapter 4) in order to address the lack of small objects and the low density of open-source point clouds. Despite the high density and novelty of intensity and 3D models in indoor datasets, our dataset does not capture the world distribution of small objects. Therefore, we produced a new synthetic dataset with virtual indoor 3D scenes and their corresponding point clouds (Chapter 5).

- Chapter 3: **Improving segmentation with handcrafted features**

In this chapter, we benefit from prior works that primarily used handcrafted features to segment the 3D point clouds. Here, we address the subject of integrating handcrafted features into deep learning architectures, and explore their potential benefits for semantic segmentation of indoor point clouds. We also introduce a new regularization techniques called "DropFeatures". This strategy randomly removes features during training to improve performance. We demonstrate that this prevents catastrophic performance drops when features are missing at inference time and can also increase performance by acting as data augmentation.

- Chapter 4: **A realistic indoor point cloud segmentation dataset**

In this chapter, we present a new dataset for the segmentation of indoor LiDAR point clouds: *3DSES (3D Segmentation of ESGT point clouds)*. The main contribution of Chapter 4 is a model-to-cloud alignment algorithm for the automated labeling of indoor point clouds using an accurate 3D model created by humans. We demonstrate that our model-to-cloud alignment produces "pseudo-labels" with over 95% accuracy on our point clouds, enabling us to train deep models with significant time savings compared to manual labeling.

- Chapter 5: **Synthetic 3D scenes for small objects segmentation**

In this chapter, we address the lack of annotated small objects in point clouds. In Chapter 5, we introduce a new synthetic dataset for indoor point cloud segmentation. We develop a process that can generate an infinite number of 3D point clouds containing a large quantity of small objects. We first use a procedural generator to create an immense number of 3D scenes, which contain structural elements (such as ceilings, floors, and walls) as well as small objects (such as forks, lamps, spoons, towels, ...). Then, we employ a virtual TLS to generate high-quality indoor point clouds with accurate semantic and instance labels from the 3D scenes.

Related publications These works have been the subject of two publications, which are listed below:

M. Mérizette, N. Audebert, P. Kervella et J. Verdun, *3DSES: an indoor Lidar point cloud segmentation dataset with real and pseudo-labels from a 3D model* dans Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 2: VISAPP, INSTICC, SciTePress, p. 707–716.

M. Mérizette, N. Audebert, P. Kervella et J. Verdun, *3D feature engineering for deep semantic segmentation of indoor point clouds*, The Photogrammetric Record, Under review

2

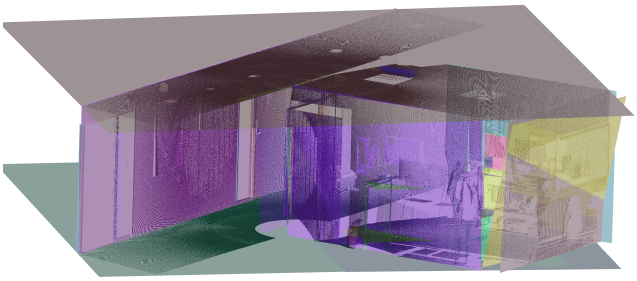
Related work

Content

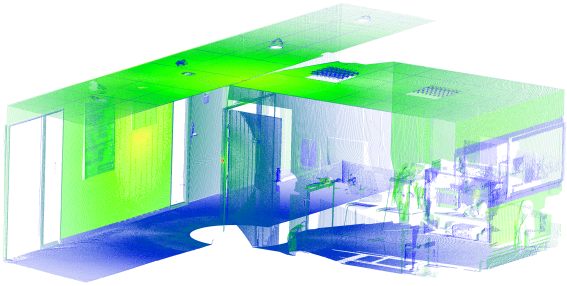
Abstract

In this chapter, we present an overview of the current state of the art in 3D point cloud processing. The works presented in the following chapters build on this state of the art and aim at fill the gaps identified. First, we briefly define the notion of 3D point clouds (Section 2.1). Associated with TLS acquisition, in Section 2.2 we introduce the concept of point cloud intensity (Fig. 2.1c) and discuss the difficulties associated with its normalization. Then, in Section 2.4, we present 3D deep learning techniques dedicated to point cloud segmentation (Fig. 2.1d). Finally, in Section 2.5, we address the challenges associated with the lack of annotated data.

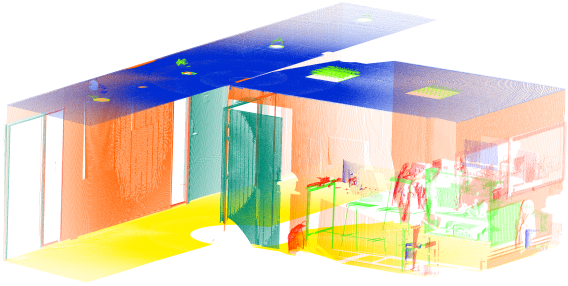
(a) 3D point cloud of 3DSES [1] with RGB.



(b) 2D planes detected by RANSAC [25].



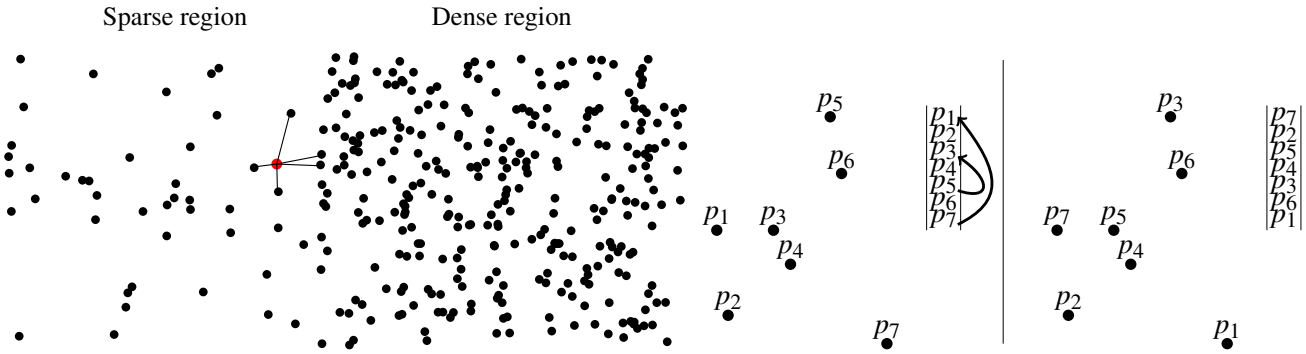
(c) 3D point cloud of 3DSES [1] with intensity.



(d) 3DSES [1] point cloud colorized by classes.

Figure 2.1: An overview of 3D point clouds [1] with different modalities, including color information, semantic labels, LiDAR intensity, and 2D planes estimated using RANSAC [25].

2.1 3D point clouds: preamble



(a) Point clouds are irregular and unstructured

(b) Point clouds are unordered

Figure 2.2: The main challenges with 3D point clouds are the irregularity and the lack of structure and order. Unlike 2D images, 3D point clouds are an irregular data structure, where 3D points are non-uniformly distributed across a grid. They can contain sparse regions, which can be challenging for object identification, as well as dense regions with many points (Fig. 2.2a). In addition, 3D point clouds pose a challenge to deep learning methods because they are not ordered (Fig. 2.2b) and therefore are invariant to permutation.

In this manuscript, we refer to a 3D point cloud as a set, *i.e.* an unordered collection, of XYZ coordinates. Additional information can be recorded with the 3D coordinates, such as color information (RGB), intensity (I), or GPS time.

Regarding 3D point cloud segmentation, most 3D advances in this field have been influenced by progresses in 2D image deep learning. However, 3D point clouds pose some challenges (Fig. 2.2) to deep models. Firstly, 3D point clouds are unstructured, they have sparse and dense regions. The distances between a point and its neighbors are variable, unlike pixels from images (Fig. 2.2a). Point clouds have another specificity: they are not ordered (Fig. 2.2b). If we change the ordering, the 3D point cloud remains the same, while the file has changed. Owing to these challenges and the complexity of applying deep learning methods to 3D point clouds, various 3D representations (Section 2.4) have been proposed. However, these challenges also affect annotation of 3D point clouds, leading to the creation of new datasets or methods that do not rely on labels (Section 2.5).

2.2 Terrestrial Laser Scanning

3D point clouds may be produced by images [30, 31, 32] or by LiDAR [33]. LiDAR can be aerial [34, 35, 36], mobile [37, 38, 28] or terrestrial [39, 1] depending on the acquisition vector. Since this thesis tackles the challenge of segmenting small objects from Terrestrial Laser Scanner (TLS) data, we present in this section some basic

principles of TLS acquisition.

2.2.1 Principles of TLS measurements

The TLS measures rely on the emission and reception of laser light. The distance between objects and TLS can be determined using time-of-flight or phase measurement techniques [40, 41, 42, 43]. Afterward, a scan of the environment (indoors or outdoors) enables to obtain a 3D point cloud.

Intensity Three-dimensional geometries (*i.e.*, XYZ coordinates) of objects are not the only information that can be retrieved from LiDAR measurements [42]. The intensity, *i.e.* the ratio between backscattered and original radiation at the specific wavelength [33], is an additional data obtained from TLS measurements [43]. In this thesis, we will discuss on this radiometric information inexactly called "intensity" [43] acquired by TLS and which consists of a measure of the electrical signal power [42]. Since TLS uses laser technology, the received power is linked to the emitted power by the laser equation [33]. According to [2, 44, 45, 46], under the supposition of a Lambertian surface and a target receiving the entire beam, the laser equation linking received power and emitted power becomes:

$$P_r = \frac{P_t D_r^2 \rho}{4R^2} \cdot \eta_{\text{sys}} \cdot \eta_{\text{atm}} \cdot \cos(\alpha) \tag{2.1}$$

where P_t is the transmitted laser power, D_r is the aperture, R is the range, α is the incidence angle, ρ is the reflectance of the target and the two factors η_{atm} and η_{sys} are the atmospheric transmission factor and a system-related factor, respectively.

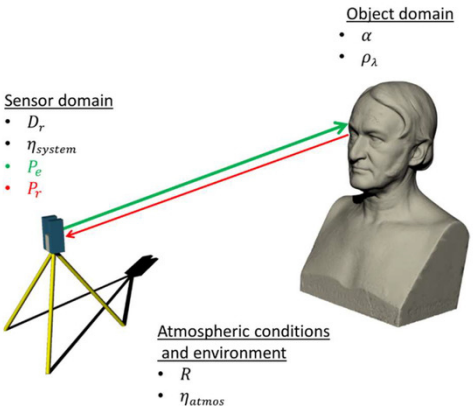


Figure 2.3: Visualization of influencing factors for a LiDAR system [2].

Eq. (2.1) shows that received laser power is affected by various parameters, such as the angle of incidence, atmospheric conditions, surface reflectivity or instrument mechanisms (Fig. 2.3). Another phenomenon that affects the intensity of a LiDAR measurement is the near-range effect [47], which is due to lens defocusing and influences intensity values for points at closer distances. Assuming that 3D points are generally collected using the same sensor, parameters sensitive to the sensor (*i.e.*, P_t, D_r and η_{sys}) can be grouped inside a constant value (*i.e.*, C) during a unique acquisitions campaign [42]. Therefore, Eq. (2.1) can be simplified as:

$$P_r = C \cdot \rho \cdot \frac{\cos(\alpha)}{R^2} \tag{2.2}$$

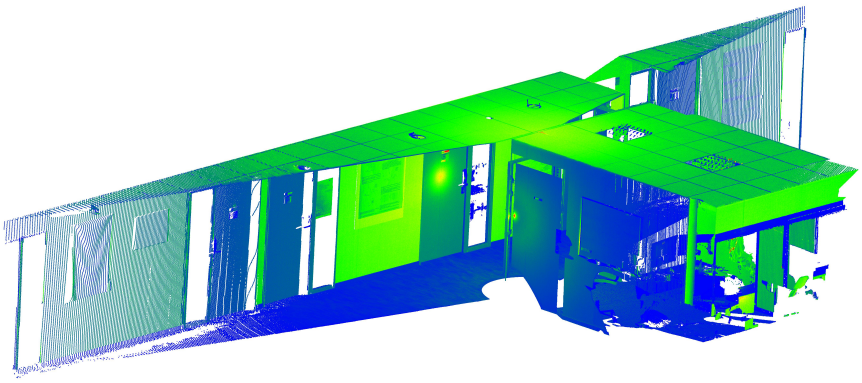


Figure 2.4: Radiometric information referred to as "Intensity". This radiometric information allows humans to identify objects in 3D point clouds. However, we can observe that the intensity varies across a same material (such as a concrete wall). Point clouds from 3DSES-Gold acquired by a RTC360 and exported from Register360. "Intensity" values comprised between 0 (in blue) and 1 (in yellow).

The previous equation indicates that the reflectance (ρ) of the object can be retrieved using the TLS intensity. It should be noted that this simplification gives rise to questions when merging multiple acquisitions (*i.e.*, with multi-sensor data acquisitions) in various applications, such as 3D visualization or training deep models.

Reflectance The reflectance is defined as the ratio between the emitting optical flow and the reflecting one by the object, for a specific incidence. This reflectance is obtained by integrating the Bidirectional Reflection Distribution Factor (BRDF) [43]. This reflectance is an intrinsic parameter of the object, which can inform us about its material and color.

2.2.2 Intensity calibration

LiDAR intensity is always recorded alongside the coordinates in TLS measurements. However, this radiometric feature is largely understudied in the literature, and especially its calibration and contribution to deep models. Nevertheless, this radiometric information is correlated with object reflectance and the idea that it can be helpful to distinguish objects in 3D point clouds. As it is possible for human eyes to distinguish objects in the 3D point cloud presented in Fig. 2.4. In this section, we present some approaches that tackle the difficulty of normalizing LiDAR intensity. The objective behind this calibration is to link the intensity value to the reflectance of an object [43]. These approaches can be divided into two categories [42, 44]:

- "data-driven" approaches: analyzing raw point cloud intensities to determine relevant parameters.
- "model-driven" approaches: correcting point cloud intensities based on physical models.

Using the previous LiDAR range equation, it's possible to approximate the intensity correction [43, 44]:

$$I_{\text{cor}}(\rho) = I_{\text{raw}}(\rho, R, \alpha) \cdot \frac{R^2}{\cos(\alpha)} \tag{2.3}$$

with I_{cor} the corrected intensity and I_{raw} the intensity from TLS. However, this correction is based on certain assumptions (*i.e.* the reflector is Lambertian and the temperature is constant, ...). Moreover, some TLS modify intensity values for near distances in order to reduce brightness [48]. Furthermore, small objects with an area inferior to the beam's footprint follow a range dependency with higher power than 2 [44]. To adapt the dependency of the intensity to several parameters, empirical models were employed. TLS measurements can be performed on Spectralon [49], cardboard [50], or an object's surface [51]. This data-driven approach led to a plethora of models, for instance a model was designed for the Leica ScanStation C10 [52], one for the Faro LS880 [53], and another for the Faro Focus 3D 120 [43]...

To summarize, the calibration of intensity is a complex task with no general law applicable to the variety of TLS devices. Furthermore, intensity data furnished with point clouds can face manufacturer pre-treatment, which can make calibration difficult [43, 44, 54] and limit its use in recent deep models.

2.3 3D segmentation: preamble

2.3.1 Definition

Point cloud segmentation is linked to image segmentation [22]. 2D image segmentation consists in assigning a semantic label to every pixel (such as dog, cat ...) [55]. With the same logic, 3D point cloud segmentation consists in labeling every 3D coordinate of the point cloud according its semantic meaning [23, 56], as we can see in Fig. 2.1d. 3D point cloud segmentation did not appear with deep learning (see Section 2.3.2 and also Chapter 3), and many methods have tackled point cloud segmentation.

2.3.2 Traditional Approaches

First approaches dedicated to point cloud segmentation used several methods, such as attribute clustering [57], region growing [24] or methods based on borders [58] or graphs [59]. As illustrated in Fig. 2.1b, previous works also used parametric shapes and RANSAC [25] to recognize structures or objects in 3D point clouds [25, 26]. Macher et al. [60] proposed a three-step segmentation process: first, partitioning the floor; second, segmenting the rooms; and third, identifying the planes that compose them. The floor identification is performed using

the distribution of the Z coordinates, enabling the identification of levels in the point cloud. A region-growing algorithm is then applied to the binary map of each level to retrieve rooms; after that, RANSAC [25] is used to detect planes (*e.g.* floors, ceilings and walls). However, aware of the limits and difficulties encountered with parametric methods, later works introduced machine learning and expert features to classify point clouds, using Support Vector Machines [61, 62], Conditional Random Fields [63] or Markov Random Fields [64].

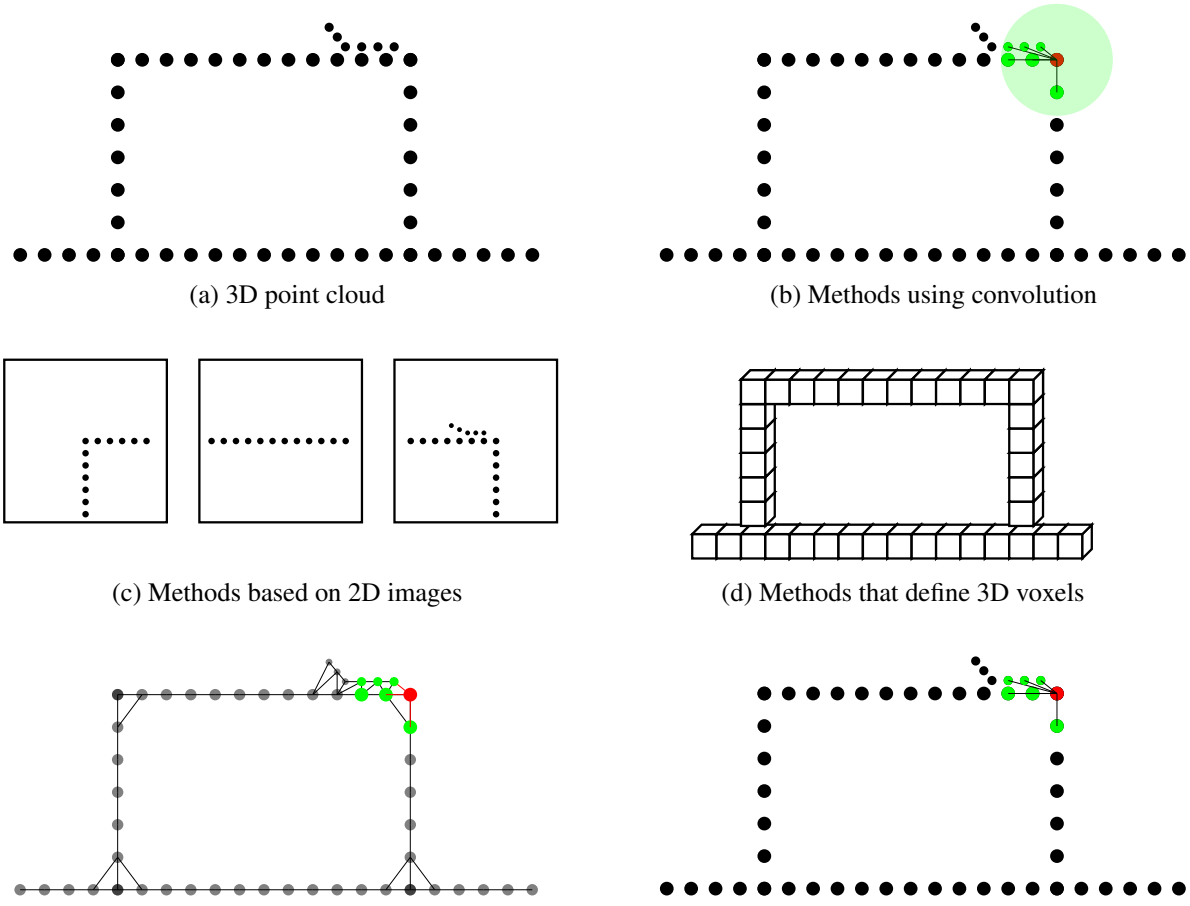
However, traditional approaches either used handcrafted features or relied on prior knowledge of the scene. This complicated the segmentation and degraded the results [65].

2.4 3D deep learning segmentation

In order to address irregularity, lack of structure and lack of order, researchers have adopted various representations to describe 3D point clouds. As shown in Fig. 2.5, five groups of methods are commonly used for segmenting 3D point clouds using deep learning. These methods can be extended to include hybrid methods and those based on superpoints [66, 23]. Methods working with 2D images project 3D coordinates to 2D planes. It is also possible to directly use 2D images from the point cloud acquisition. Another group of methods are convolutional models, which are directly inspired by the 2D image field. Other methods are derived from 3D point cloud products, *e.g.* graphs, voxels and superpoints. Since the pioneering PointNet [67], some deep models now deal directly with raw 3D data as input and develop architectures to handle raw data. On a specific side, some rare hybrid methods attempt to benefit from all the previous groups of methods.

2.4.1 Methods based on convolution

Methods in this section are inspired by 2D convolutional kernels [68, 69]. Designing convolution kernels for point clouds is a challenging task. Convolutional kernels can be split into two categories: continuous and discrete convolution. KPConv [70] and ConvPoint [71] use convolutions in an Euclidean space. As KPConv uses continuous convolutions, they can be learned by the network to create deformable convolutions. Of late, KPConv-X [72] enhance KPConv with an attention kernel. Unlike other methods that rely on dense convolution, PointCNN [68] employs sparse convolution kernels. Finally, Halperin et al. [73] points out that applying convolutional operations directly in 3D space can lead to increased inference times due to the increased computational complexity. Sparse convolutions may offer some benefits in terms of memory efficiency, but they introduce dilation problems (*e.g.* convolution spreads values, activating nearby cells that should stay off).



(a) 3D point cloud

(b) Methods using convolution

(c) Methods based on 2D images

(d) Methods that define 3D voxels

(e) Methods using graphs

(f) Methods operating directly on point clouds

Figure 2.5: Illustration of five different groups of methods working with point clouds, using a point cloud of a laptop on a desk as example. Some approaches adapt 2D convolution to in 3D fields (Fig. 2.5b) or use images to segment 3D point clouds (Fig. 2.5c). Other methods address the irregularity of point clouds using 3D grids (Fig. 2.5d) or graphs (Fig. 2.5e). The most recent techniques work directly on raw point clouds (Fig. 2.5f).

2.4.2 Methods using 2D images

Image-based methods project a 3D point cloud into multiple views [23]. We can consider two subcategories. The first uses raw images, acquired directly from 3D scanners or photogrammetric approaches. The second uses novel views, *e.g.* creating viewpoints within the scene, to project 3D coordinates onto a 2D plane in order to reconstruct images. Both aim to take advantage of image deep models to extract features. Then they either perform classification on images or project features into 3D space for classification. MVCNN [74] was a pioneering work that first applied this practice to 3D shape model classification. This approach was later extended to 3D object detection [75] and 3D segmentation of scenes [76]. Despite their time efficiency, these methods rely on the choice of a viewpoint [23], and they are extremely sensitive to occlusions and image selection. This issue has been widely studied in the field of surface reconstruction [77], and images can be selected using geometric priors or, more recently, the attention mechanism. For instance, MVPNet [78] introduces a view selection using a 2D-3D feature aggregator module and multi-view feature fusion using PointNet++. DeepViewAgg [77] introduced a method to select the most relevant 2D features based on an attention mechanism. Of late, DITR [79] argues that the potential of visual foundation models (*e.g.*, models trained on large-scale 2D datasets) for 3D tasks is largely underestimated. The authors propose an approach for extracting features from visual foundation models and using them in 3D deep learning architectures. They also introduce a pre-training task, when 2D images are missing, to distill 2D foundation models onto 3D models. Recently, with the introduction of SAM [19], many studies have attempted to perform zero-shot semantic and instance segmentation [80, 81].

The methods presented in this section are highly efficient in terms of computational time, as they rely on few parameters and can employ efficient 2D architectures [73]. However, all image-based methods suffer from loss of information [73] and well-known challenges such as occlusion and the difficulty of aggregating 2D features [23].

2.4.3 Methods based on 3D voxels

The methods grouped in this section, also known as "volumetric-based", tend to address one of the main challenges of 3D point clouds: their irregularity. These methods project a 3D point cloud onto a 3D grid (Fig. 2.5d) before applying a 3D Convolutional Neural Network (CNN). VoxNet [82] introduces an occupancy grid representation for real-time object detection. OctNet [83] proposes a hybrid grid-octree to create hierarchical partitions and focus computation only on relevant regions to reduce memory footprint. Following previous

works [84, 85] use sparse voxels to restrict CNN computation to non-empty voxels. Minkowski [86] designs a library for sparse tensors that enables the processing of large 3D point clouds. Always with the aim of reducing memory consumption, PointGrid [87] proposes a network that works with a fixed number of points within each grid. DRINet++ [88] proposes to treat a voxel as a point. By treating each point within a voxel as a "super point", the authors reduce the number of operations while implicitly preserving the geometry information. More recently, PVT [89] proposes to use Transformers on voxels and introduce a sparse window attention to avoid wasting time to structure the data. OctFormer [90] proposes to compute an octree attention. This ensures an equal number of points in each window, reducing computational cost while maintaining high accuracy.

Despite being one of the simplest methods to apply deep models to point clouds, voxels (Fig. 2.5d) can lead to information loss and artifacts at the boundaries of the voxels. In addition, creating voxels is time-consuming, and [23] notes that choosing the size of voxels can be challenging. Indeed, high resolution is expensive in terms of computational resources, but low resolution increases loss of information.

2.4.4 Methods based on graphs

Some authors have focused on graph-based deep networks (Fig. 2.5e) to segment 3D point clouds. Graph-based networks consider each point of a 3D point cloud as a graph vertex, and then generate edges based on the point neighborhood. Feature learning can be done in spectral [91] or spatial [92] domains. In the spectral domain [93, 94], the convolutions are defined by multiplying graph signals by the eigenvectors of the Laplacian matrix [91, 95]. RGCNN [93] constructs a graph based on coordinates and normal vectors, and considers point features as signals. RGCNN utilizes convolutions in spectral domains, and the graph is updated during the feature learning process. PointGCN [94] employs a Gaussian kernel to weight each edge of the graph and captures global and local structures using global and multi-resolution pooling. In the spatial domain [92, 96], the vertex features of a point cloud consist of its XYZ coordinates and any additional LiDAR information, such as intensity or color, while the edge features are derived from the geometric attributes between two connected points. As a precursor work, ECC [92] employs convolutions on a graph, where each point is considered as a vertex and is connected to its neighbors by edges. DGCNN [96] constructs a graph in feature space and updates it at each stage of the network. Then, the authors introduce a novel operation called EdgeConv, which generates edge features that describe the relationship between a query point and its neighbors. Finally, AF-GNN [97] uses an attention mechanism to improve semantic segmentation with graphs. Classical graph convolution is used in the encoder stage, while the attention mechanism is used in the decoder stage to suppress irrelevant information

from graph neighbors. Of late, SuperCluster [98] considers panoptic segmentation as graph clustering and uses graphs to enhance panoptic segmentation performance.

However, these methods are mainly limited by their small receptive fields [99] and can face scalability issues when dealing with large and dense point clouds. Moreover, operations in spectral domains are computationally expensive when calculating the eigenvectors of the Laplacian matrix.

2.4.5 Methods operating directly on raw point clouds

Previous methods do not work on 3D point clouds they transform 3D points into a different structure in order to apply deep models. As a pioneering work, PointNet [67] was the first to propose a neural architecture to work directly on 3D point clouds. PointNet takes as input the XYZ coordinates of the 3D point clouds, as well as colorimetric information and is able to classify each point. PointNet uses a simple architecture, invariant to permutation, consisting of a MultiLayer Perceptron (MLP) to learn global features for the point cloud. However, this simple architecture has limitations. The model is not able to capture the point hierarchy, has difficulty to capture local structure and struggles to generalize to complex scenes. Following PointNet, PointNet++ [100] attempts to capture the hierarchy of 3D point clouds, as well as neighborhood features at different scales. To achieve this, PointNet++ introduces a Set Abstraction layer, which consists of three steps: 1) sampling the point cloud to determine the centroids; 2) selecting neighbors to determine local regions; and 3) applying a mini-PointNet to create feature vectors. Since the emergence of PointNet and PointNet++, a plethora of architectures have been developed to work directly on 3D point clouds. For example, RandLANet [101] addresses the challenges of segmenting large 3D point clouds, using random sampling to improve time efficiency and introducing a local feature aggregation to overcome the risk of deleting key features during sampling. PointNeXt [100] revises PointNet++ to improve performance by introducing data augmentation, optimization techniques, and model scaling. PointTransformer [102] and Stratified Transformer [103] employ Transformer architectures to improve accuracy. PointVector [104] introduces 3D vectors in the Set Abstraction of PointNeXt. PointVector advances the idea that scalars can be replaced by 3D vectors, which are more expressive for representing features. More recently, PPT [105] tackles the existing domain gap between 3D datasets and introduces a multi-dataset training strategy. PPT relies on prompt normalization for model adaptation and on text encoder to address the transfer issue. PPT demonstrates improvements when using multi-dataset training with several backbones. PointTransformerV3 [106] also benefits from this multi-dataset pretraining [105]. PointTransformerV3 focuses its attention on time saving, and in particular replaces the

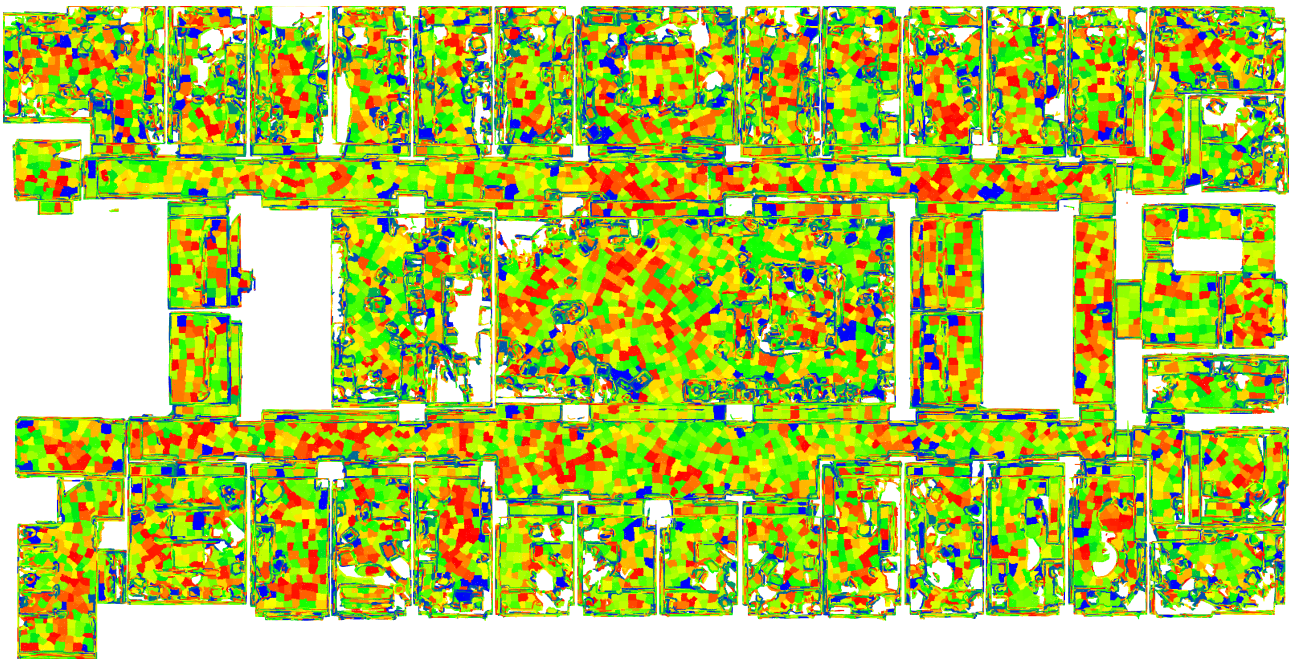


Figure 2.6: Clustering 3D points into regions, such as in the first area of S3DIS, can be achieved using efficient methods [107].

time-consuming K-Nearest Neighbors (KNN) algorithm with a serialized neighborhood mapping.

In summary, a major challenge for point-based methods is to capture local structure without significantly increasing computational time, making them less suitable for real-world applications [73].

2.4.6 Methods based on superpoints

Previous methods that work directly on 3D point clouds have a major constraint on large-scale point clouds: their time-consuming processing. Some studies attempt to reduce this constraint by partitioning the point cloud. To address the challenge of semantic segmentation of large point clouds, SuperPointGraph [108] first partitions the 3D point cloud into homogeneous regions called "superpoints". It then constructs a graph, called a "superpoint graph", and segments 3D point clouds using graph convolutions. SSTNet [109] and SPFormer [110] extend the prior work of SuperPointGraph, employing "superpoint" techniques to improve inference speed during instance segmentation. While the inference of SuperPointGraph is quick, SuperPointGraph suffers from a long pre-processing time to prepare the data for training. Based on hierarchical partitioning of 2D images [111, 112], SuperPoint Transformer (SPT) [107] proposes a method based on multi-scale partitioning. SPT shorten the processing time compared to SuperPointGraph. SPT benefits directly from the fields of 2D image processing

and superpixel ideas [113], and use the recent Transformer architecture to improve point cloud segmentation.

However, superpoint based methods can lead to the grouping of points belonging to different classes and require some resources to create partitions, which offsets their training time efficiency.

2.4.7 Hybrid methods

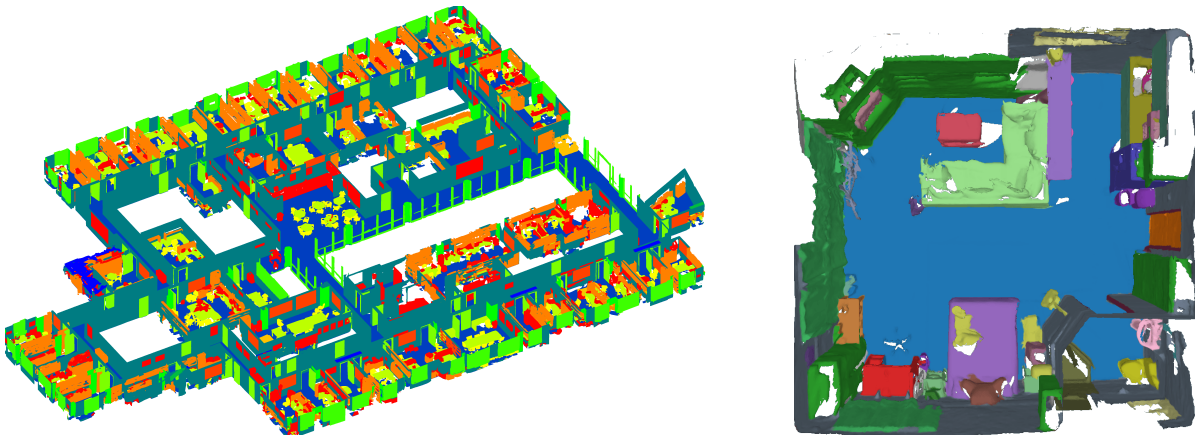
The previous methods have their own advantages and disadvantages. Methods detailed in Section 2.4.2 (such as DeepViewAgg [77] or DITR [79]) already combine point clouds and images, since they project 2D features onto a network that work with 3D point clouds as input. Based on these considerations, recent works have attempted to use multiple representations as inputs to deep models, such as 3DMV [114]. The particularity of 3DMV is that 2D features learned by 2D networks are projected onto voxels. Segmentation is then performed using the 3D CNNs presented in Section 2.4.3. The methods grouped in this section aim to use all the information available when scanning an indoor environment, such as XYZ coordinates, images... to identify objects within a 3D point cloud. RPV-Net [115] presents a three-branch architecture for the segmentation of 3D point clouds. This architecture takes voxels, XYZ coordinates, and a range image (*i.e.*, panoramic view) as inputs. The authors then designed a module to merge features extracted from all these modalities to improve point cloud segmentation. OmniVec [116] proposes a multi-task learning architecture (such as image recognition, point cloud segmentation, text summarization, etc.) with different modalities: images, depth maps, video, audio, text, and XYZ coordinates. They showed that using a joint training on different modalities helps the network to share information and improves segmentation.

All the methods grouped in this section aim to exploit data information sharing. However, these methods can lead to an increased processing time and may require the use of multiple data sources, even if they are unavailable at the time of acquisition.

2.5 3D networks: the lack of annotated data

In this section, we will discuss about the difficulty labeling 3D. Then, we present some strategies that reduce reliance on human labels.

2.5.1 Collecting 3D datasets



(a) Top view of area 5 of S3DIS colorized by class.

(b) Top view of a scan from ScanNet, colorized by class.

Figure 2.8: Top view of 3D point clouds from two commonly used datasets for indoor point cloud segmentation: S3DIS (Fig. 2.8a) and ScanNet (Fig. 2.8b). In these two datasets, the classification does not accurately represent the diversity of objects in the real world, and some objects are grouped into larger structures (*e.g.* switches are included in walls).

Figure 2.7: Different objects shapes from the ShapeNet dataset [3].

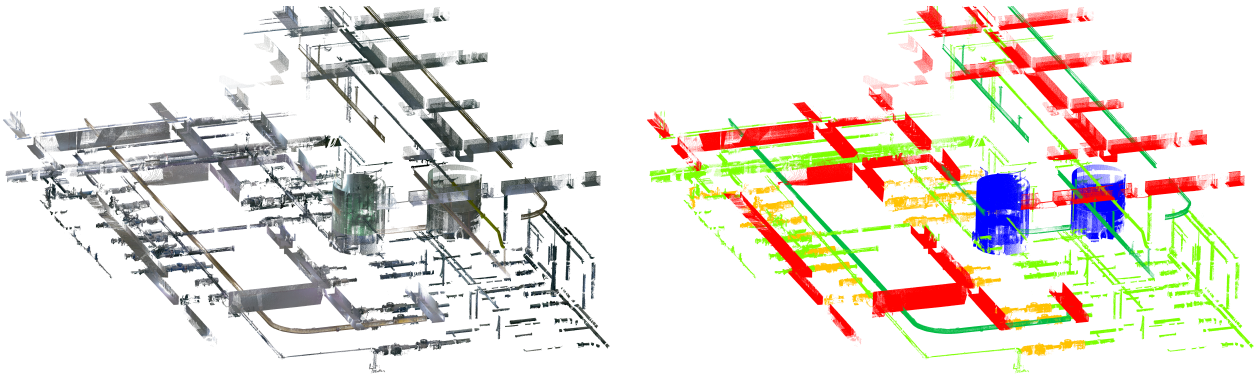
Since this thesis tackles the segmentation of objects within large-scale indoor point clouds, this subsection only presents datasets for indoor semantic segmentation. We do not focus on indoor datasets containing objects (Fig. 2.7), although such datasets are often used as benchmarks for pre-trained deep models [3, 117, 118]. A comprehensive review of such datasets can be found in Guo et al. [23]. The methods presented so far are supervised methods; *i.e.* they require labels. Supervised learning requires a large amount of labeled data to achieve high semantic segmentation accuracy. Despite advances in deep learning segmentation and the accessibility of 3D acquisition, many methods still rely on large annotated datasets (Fig. 2.8). However collecting and especially labeling 3D data is a complex task. As discussed in Section 2.4, 3D point clouds present several challenges characteristics compared to images, including sparsity, occlusion, and high size. Compared to image labeling, 3D point cloud labeling is time-consuming and reserved for expert users [119, 66]. This labor-intensive work [66] has been carried out due to the class limitation in existing datasets and the limited size of these indoor datasets. Consequently, 3D datasets do not reflect the real-world distribution of objects. As we can see in Table 2.1, popular 3D point cloud segmentation datasets are limited in terms of classes, except for ScanNet++ [39]. The two most used datasets for indoor point cloud segmentation, S3DIS [120] and ScanNet [121], the number of classes is limited

Table 2.1: Comparison of the characteristics of point cloud datasets from the literature. *Note that 3DSES [1] is a contribution of this thesis and is presented in Chapter 4.*

Name	Environment	Classes	Extent	Points (M)	Intensity	3D model	Source
S3DIS [120]	Indoor	13	6020 m^2	273	❌	❌	Camera
Matterport3D [131]	Indoor	20	219 399 m^2	-	❌	❌	Camera
ScanNet [121]	Indoor	20	78 595 m^2	242	❌	❌	Camera
ScanNet200 [132]	Indoor	200	78 595 m^2	242	❌	❌	Camera
ScanNet++ [39]	Indoor	1733	15 000 m^2	446	❌	❌	TLS
LiDAR-Net [133]	Indoor	24	30 000 m^2	3619	❌	❌	MLS
InLUT3D [134]	Indoor	18	-	3500	❌	❌	TLS
3DSES Gold 🥇 [1]	Indoor	18	101 m^2	65	✅	✅	TLS
3DSES Silver 🥈 [1]	Indoor	12	304 m^2	216	✅	✅	TLS
3DSES Bronze 🥉 [1]	Indoor	12	427 m^2	413	✅	✅	TLS
ScanNet++v2	Indoor	3470	-	-	❌	❌	TLS

to 13 and 20 classes, respectively, and does not represent the real diversity of objects in 3D point clouds. Meanwhile, precursor image segmentation datasets, such as PascalVOC [122] (20 categories) and COCO [123] (80 categories) surpass S3DIS and ScanNet. ImageNet1K [17] with its 1000 categories surpasses all datasets except ScanNet++. Datasets introduced in recent years, such as OpenImages (over 350 categories) [124], LVIS (1,200 categories) [125] for instance segmentation, and ImageNet21K [126] with its over 21,000 categories, far surpass all indoor point cloud segmentation datasets, even the largest ones. As we can see in Table 2.1, the order of magnitude of 2D classes is greater than that of the majority of 3D dataset classes. Most existing 3D datasets contain a small number of classes, even tough some papers attempt to tackle this human effort in creating point cloud segmentation datasets. The ScanNet authors introduce a pipeline based on crowdsourcing to annotate point clouds. They develop a user-friendly interface to enable non-familiar persons to label point clouds. They also rely on reduced mesh and over-segmentation [127] of this mesh, to produce a lightweight product for the annotations. ScanNet++ reuses this pipeline to annotate more clouds to more precise categories. Another strategy to reduce annotation effort is to overlay 3D scans [38]. This allows annotators to label multiple scans in a single pass. On the other hand, [128] exploits virtual reality to design a tool that simplify human labeling. The authors segment persons and objects, while more recent papers adapt this process for large point clouds [129, 130]. Even though this method simplifies annotation and makes it more user-friendly, it still requires significant resources to use virtual reality.

The low number of classes is not the only characteristic of indoor point cloud segmentation. Another concern



(a) Overview of PSNet5 Area 2 with color.

(b) Overview of PSNet5 area 2 colorized by classes.

Figure 2.9: Overview of PSNet5 area 2 dataset with color information (Fig. 2.9a) and class information (Fig. 2.9b). Legend: I-beam in dark green, rectangular beam red, pipe in light green, pump orange, tank in dark blue.

is the area. We can see in Table 2.1 that larger datasets are correlated with photogrammetric acquisitions [120, 131, 121, 132]. More recently, some datasets tend to use TLS, *e.g.* ScanNet++ [39], InLUT3D [134]. However, radiometric information is only provided for one dataset using TLS: 3DSES [1]. Note that the 3DSES [1] dataset was developed during the course of this thesis and will be presented in detail in Chapter 4.

The previous conclusions also hold valid for industrial point cloud segmentation datasets. Fewer point-labeled point clouds are open source for this type of environment. For example, CLOI [135] contains oil and gas point clouds, but it is not open source. The EIF [136] dataset contains industrial objects such as valves, tees and pipes, but does not contain an entire scene. As we can see in Fig. 2.9, PSNet5 [137] only contains partial scenes of 4000 m^2 of an industrial site. It consists only of beams, pipes, pumps, and tanks. This dataset does not contain any other industrial elements and is missing structural elements such as floors and walls. It also contains a small number of points, around 80 million.

Annotation Time According to the authors, annotation of SemanticKITTI [38], including corrections, took a total of 1700 hours. For indoor scenes, ScanNet [121] reported an average of 22.3 minutes per scene (*i.e.*, more than 500 hours for the entire dataset) using a process on Amazon Mechanical Turk with only twenty classes. For a larger dataset in terms of area and classes, ScanNet++ [39] required one hour per scene to produce more than 1,000 categories. This is similar to the time required to annotate high-density TLS point clouds from 3DSES [1], as explained in Chapter 4. LiDAR-Net authors required 550 hours to annotate their MLS point cloud dataset, focusing on twenty-four classes. This highlights the difficulty of annotating 3D point clouds, which explains why point cloud datasets have a limited number of categories and cover small areas. As shown in the previous

paragraph, the annotation time does not favor the development of indoor point cloud datasets. To overcome the lack of labeled data, it is possible to create synthetic data on which to train deep models. Researchers have also explored alternative approaches that do not rely on extensive labeling. This includes weakly supervised learning methods (Section 2.5.2), which use a small number of labels, and self-supervised learning methods (Section 2.5.3), which can operate with no labels at all.

2.5.2 Weakly supervised methods

Weakly supervised learning can be divided into three categories [138]: incomplete supervision (*i.e.* with a small number of labels), inexact supervision (*i.e.* labels not as exact as desired) and inaccurate supervision (*i.e.* some labels suffer from errors).

Guinard et al. [139] employ a conditional random field classifier for weakly supervised tasks, however, this approach struggles to capture context. Xu et al. [140] observe that the gradient of incomplete supervision is close to fully supervised gradient (*i.e.*, it is a good approximation). Then, they design a complex architecture with several branches to learn segmentation from a few labeled points. They showed promising results on point cloud datasets, and that 10 percent of points is enough to approximate the full gradient with their model. However, their method is not designed to deal with large-scale point clouds and is limited to instances of small-scale point clouds (due to the lack of a topological relationship and the computational complexity [141]). In the context of weakly supervised learning, some studies [140, 142] highlight that it is better to have one or more labels per object rather than fully annotating a part of the dataset. For instance, Hu et al. [142] show that annotating only 0.1% of a room in S3DIS takes only two minutes instead of twenty minutes for full labeling. SQN [142] employs RandLA-Net [101] to perform large-scale weakly supervised segmentation. SQN uses the RandLA-Net encoder to extract features from the entire point cloud, as well as a query network that selects relevant features for a query point. While presenting a simple approach, SQN underperforms older RandLA-Net on six-fold validation on S3DIS. PSD [143] introduces a self-distillation framework with two branches: one for the original point clouds and one for perturbed point clouds. Despite achieving close IoU to supervised methods, PSD may be limited by its affinity matrix when processing large point clouds. HybridCR [144] designs a contrastive network and mainly relies on data augmentation. Of late, WSPCSS [145] addresses label sparsity using 2D images and SAM [19]. It uses SAM to create artificial labels and augment the initial sparse labels of 3D point clouds, and then applies active learning. However, using images induces projection errors, and SAM may introduce incomplete 2D masks. Moreover, 2D images are not available alongside all point cloud acquisitions. Despite promising research in this

field, these methods perform worse than supervised methods and are bounded by the information provided by sparse labels [145]. Furthermore, most existing methods rely on complex training frameworks [144, 142], with multi-stage processes that are not suitable for practical applications.

2.5.3 Self supervised methods

The deep models presented in this section are largely driven by advances in image self-supervised learning [146]. To cope with the lack of large annotated datasets, some studies have focused on self-supervised learning. This learning is used to pre-train networks before fine-tuning them for downstream tasks. However, these pre-trained features can be leveraged to reduce the number of labels in a dataset, and thus the labeling effort required to create a dataset.

Object level pre-training The first works to apply self-supervised learning to 3D point clouds focused on objects rather than scenes. For example, FoldingNet [147] uses an autoencoder architecture for unsupervised learning on ShapeNet [148]. Following PointBert [149], PointMAE [150] applies the principle of masked autoencoding to 3D point clouds. PointMAE randomly masks a portion of the 3D cloud and then uses an autoencoder to retrieve the masked XYZ coordinates of the object. I2P-MAE [151] uses image knowledge to guide the 3D mask autoencoding task. Recently, 3D-JEPA [152] applied the concept of Joint Embedding Predictive Architecture (JEPA) introduced by Lecun [153]. While previous methods attempted to reconstruct the raw XYZ coordinate data, 3D-JEPA objective is the abstract semantic representation. Additionally, 3D-JEPA creates "target" blocks from a point cloud at a smaller scale, while also randomly selecting a "context" block at a larger scale that provides global information.

Scene level pre-training PointContrast [154] pretrained a model on ScanNet [121] then fine-tuning it on several datasets. PointContrast selects point-cloud pairs (*i.e.* with an overlapping superior to 30%) on ScanNet and trains a registration task, based on FCGF [155]. The authors demonstrate that pre-training on large datasets helps improve final model accuracy on downstream tasks. Unlike PointContrast, which is limited to existing RGB-D scans, MSC [156] works directly on the 3D point cloud and generates pairwise frames. Based on SimCLR [157], MSC introduces a wide range of augmentations (such as random brightness, saturation and rotation). GroupContrast [158] highlights that previous methods discard semantic correlation when generating distinct views, and proposes a semantic guidance during the learning. Contrastive Scene Contexts [159] revisits PointContrast and divides the point clouds into several regions, before applying contrastive learning. Then

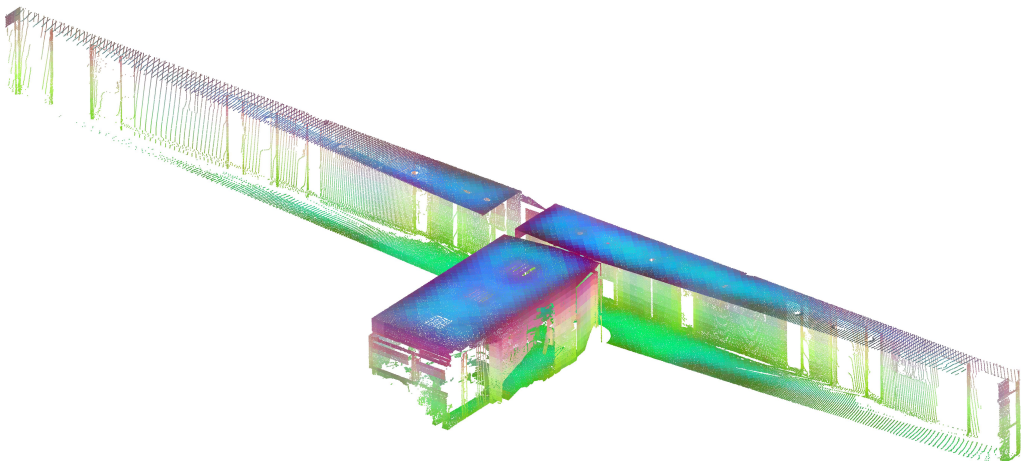


Figure 2.10: The Sonata model demonstrates its ability to extract semantic features from PCA (Principal Component Analysis) representations of 3D point clouds in 3DSES. A quick glance at the figure reveals distinct groups of points, such as ceiling and slab.

it uses an active labeling strategy to performs instance segmentation. Recently, Sonata [160] highlights that previous methods completely fail when using linear probing (*i.e.*, freeze the entire backbone and train only a new head), compared to training from scratch. Sonata advances the idea that previous models primarily learn spatial features (*e.g.* height and normal) rather than semantic features. Sonata proposes a self-distillation approach, where a teacher model works on global views and a student model learns from local views. Furthermore, Sonata expands the multi-dataset strategy introduced in PPT [105] to new datasets for a total of 140,000 point clouds. Then Sonata is trained on these 140,000 point clouds, enabling metrics close to state of the art when using linear probing (<0.2 million parameters). Sonata demonstrates zero-shot ability to express the semantic meaning of a scene (Fig. 2.10). Despite this promising approach, Sonata relies on color and was trained using 32 GPUs.

The lack of annotated data, especially regarding small objects, plays a crucial role in our work. Despite recent advances in weakly and self supervised learning, these methods still present limitations. As we prefer to rely on supervised methods, the following chapters present our approaches to deal with the scarcity of 3D labeled data in order to improve deep segmentation accuracy.

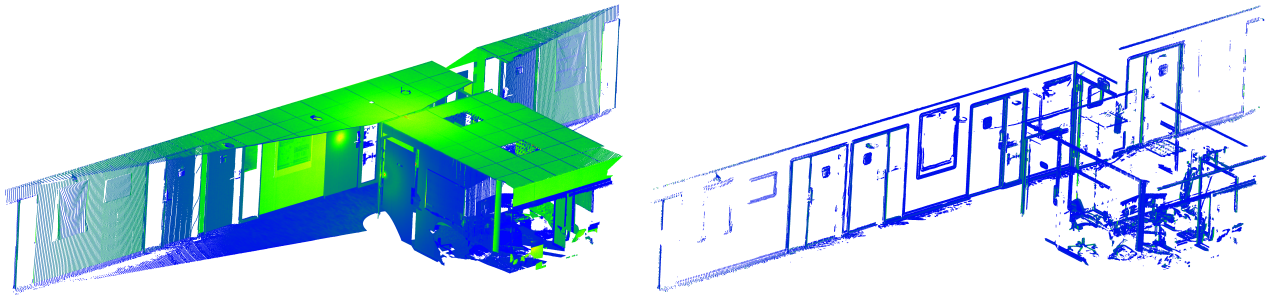
3

Improving segmentation with handcrafted features

Content

Abstract

Prior to deep learning, handcrafted features and methods based on rules were the only tools available to enrich 3D point clouds. However, new deep models tend to overlook classical geometric features used in photogrammetry, despite the fact that they could benefit model training and improve segmentation accuracy. We show that incorporating geometric features, such as linearity, omnivariance, and curvature, can enhance deep semantic segmentation of 3D point clouds on several indoor datasets. To improve the robustness of such models to scenarios where these features are unavailable, we introduce "DropFeatures", a regularization strategy that randomly removes geometric features during training. We show that this not only prevents catastrophic performance drops when features are missing at inference but can also improve performance by acting as data augmentation. In order to benefit from previous surveyor acquisitions, which include XYZ coordinates and intensity, we conduct experiments on radiometric features, the intensity.



(a) Point clouds colorized by intensity

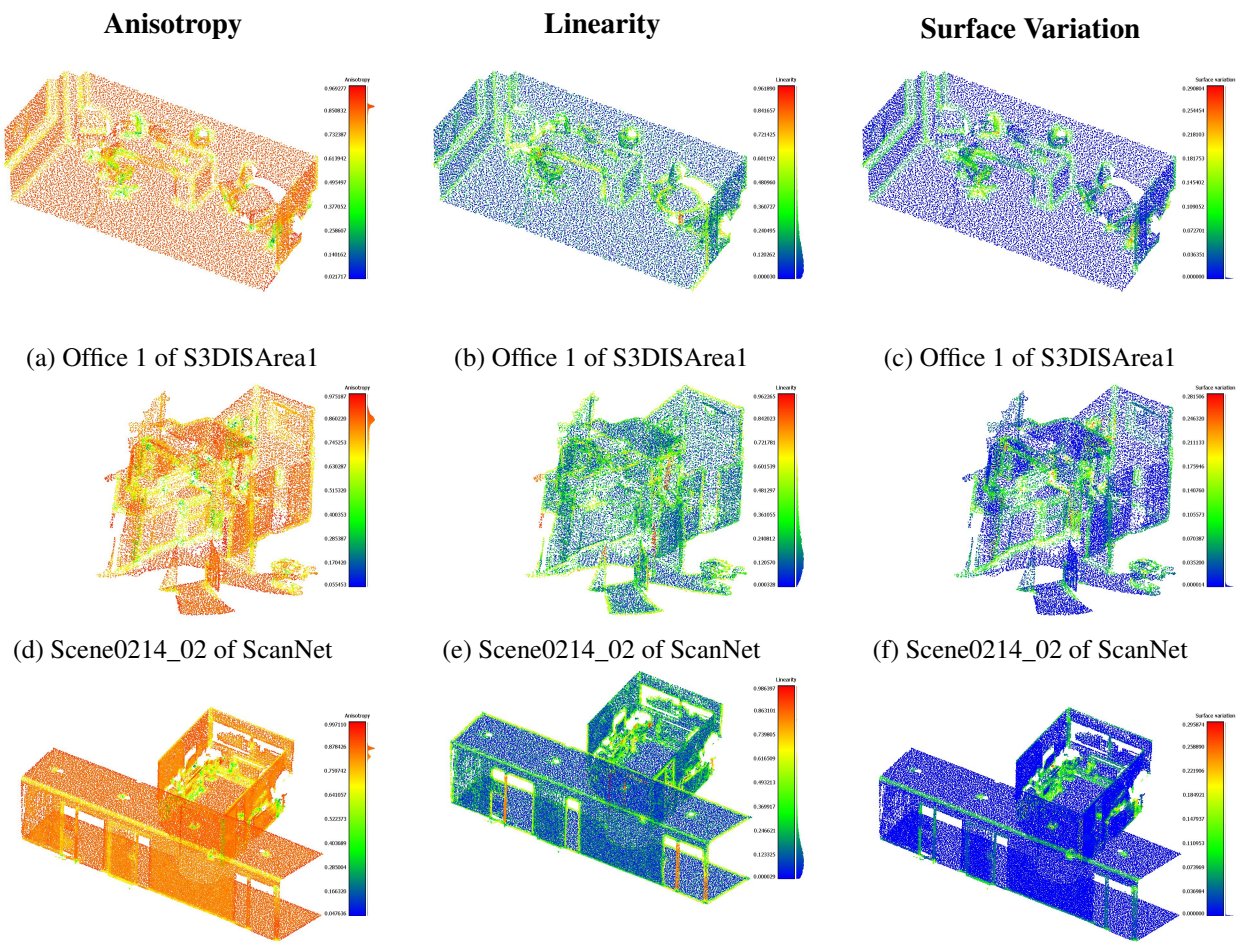
(b) Point filtered based on local curvature

Figure 3.1: Handcrafted features can help to identify interesting points, such as those on corners. Specific to LiDAR acquisition, another features can be available the intensity. Example from 3DSES [1].

3.1 Handcrafted features in the era of deep learning

Over the years, advancements in acquisition techniques have democratized the use of three-dimensional (3D) data. Older methods for creating a 3D point clouds often relied on photogrammetric approaches. However, recent improvements of 3D sensing methodologies have rendered the acquisition of three-dimensional data more intuitive and straightforward. The advent of 3D reconstruction devices, such as LiDAR and time-of-flight cameras, has significantly facilitated the acquisition of 3D point clouds. Nevertheless, during their acquisition, land surveyors may not have access to colorimetric information, due to the additional time often required to record images. As discussed in Section 2.2, TLS measures are based on laser emission, which enables the record of an intensity. Therefore, the majority of scans only have XYZ coordinates and intensity (cf. Fig. 3.1a). Yet, the intensity has not been well studied in deep models, due to unclear preprocessing by manufacturers [43], difficulties in normalizing [42, 43], and its absence from most indoor datasets (cf. Section 2.2 for further details). Furthermore, if we want to benefit from all previous point cloud data from land surveyor companies, we are restricted to XYZ coordinates. Therefore, a value correlated with the object's reflectance would be very useful to improve the accuracy of deep models. Another alternative is to rely solely on XYZ coordinates. Handcrafted features allows for the effective use of local structures in a neighborhood (Fig. 3.1b). Geometric features are contingent on 3D coordinates, while other information, such as color or intensity, is acquired by the scanner and may not be available due to the acquisition process. Furthermore, these geometric features confer the benefit of not requiring a substantial volume of data for the training of deep models [23], which can be beneficial for land surveyor companies.

As introduced in Chapter 2, first semantic segmentation methods relied upon local neighborhoods, with handcrafted rules being used for feature extraction [161]. Subsequently, machine learning was used to classify 3D point clouds such as Support Vector Machine (SVM) [61], Random Forest [12], and Markov Random Field [64]. Nevertheless, the process of features extraction by hand has limited capacity for feature expression, a constraint that is especially pronounced in complex scenes. This limit is due to its heavy reliance on expert knowledge [161]. Advancements in computing and data processing have made deep learning more applicable to point cloud analysis. Presently, the majority of deep learning approaches focus developing new architectures by integrating of modules that extract more relevant local and global features. This tendency has mostly resulted in an augmentation of the number of network parameters and a concomitant increase in computational complexity. This may be unreasonable for land surveyor companies, which may not have high computational power, and do



(a) Office 1 of S3DISArea1 (b) Office 1 of S3DISArea1 (c) Office 1 of S3DISArea1

(d) Scene0214_02 of ScanNet (e) Scene0214_02 of ScanNet (f) Scene0214_02 of ScanNet

(g) S168 of 3DSES (h) S168 of 3DSES (i) S168 of 3DSES

Figure 3.2: Overview of three geometric features: “Anisotropy”, “Linearity” and “Surface variation” on three point clouds from three datasets: S3DIS, ScanNet and 3DSES.

not want to rely on external providers to carry out their point cloud processing.

Since deep networks automatically learn representations, classical feature engineering derived from signal processing has largely faded from the relevant literature. Nevertheless, some state-of-the-art methods have tried to shift their focus from developing new architectures to addressing other factors, such as integrating color information from RGB images, improving training strategies [100], or incorporating prior knowledge about local geometry [107, 162]. In DC3DCD [163], the authors trained a deep model to detect and classify changes in 3D point clouds. They also demonstrated that adding handcrafted features to deep models improves performance. For these reasons, we investigate whether geometric features are still relevant in the age of deep learning for semantic segmentation of indoor 3D point clouds. Previous works [164, 165] introduced local descriptors such

as “linearity”, “planarity”, “anisotropy”, and “omnivariance”. Fig. 3.2 illustrates some of these handcrafted features, and provides a visual representation of how these features can help distinguish different elements. For example, “linearity” helps to identify linear structures in point clouds. This chapter explores different sets of features based on the local neighborhood of point clouds in a variety of datasets. However, the computation of geometric features during inference can be time consuming, especially for large and dense point clouds. In practice, all desired geometric features may not always be available during testing. To address this practical challenge, we introduce a random reset strategy (see Section 3.3), similar to Dropout [166], that randomly resets the geometric features to zero during the training. This enhances the robustness of the models, leading to deep networks that can benefit from geometric features when they are available, while maintaining accuracy in their absence. Finally, in Section 3.5, we attempt to calibrate the intensity from several scanners.

3.2 Previous work

Geometric features are numerical descriptors that describe the local geometric properties of 3D point clouds. In most cases, they focus on describing the spatial neighborhood around a point, producing a feature vector that can be used in statistical models for classification, reconstruction, etc.

3.2.1 3D neighborhood

One of the core problems in extracting geometric features from 3D point clouds consists in determining what is an optimal neighborhood around a point. A neighborhood that is too large will misrepresent the geometry of the object surrounding the point of interest, while a neighborhood that is too small will contain too few points to be statistically representative. Various strategies have been introduced over the years to select an appropriate neighborhood for 3D point cloud classification. This selection step is crucial for geometric features, given its impact on their accuracy, but also their compute and memory costs [165]. The neighborhood of a point can be defined as its k-nearest neighbors using the Euclidean distance in 3D space [167]. This results in a fixed number of neighbors, also can lead to defining as neighbors points that are quite far away if k is large or the point cloud is sparse. Alternatively, the neighborhood of a given point can be defined as all the points inside some geometrical shape. A classical approach consists in defining the neighborhood as the points inside a sphere of a fixed radius r [168]. It’s also possible to define a cylindrical neighborhood [169], projecting 3D points onto a 2D plane and keeping points inside the 2D disk of radius r. These methods have the main drawback of defining neighborhoods using a unique scale parameter, either expressed as a fixed radius r or a

number of neighbors k. Consequently, setting this parameter requires prior knowledge of the scene. Furthermore, because this parameter is uniform across the entire scene, it does not account for the varying density of the point cloud, *e.g.* mixing sparse and dense regions. In practice, this has led to dataset-specific tuning of the neighborhoods to deal with the characteristics of each point cloud. Recent works [164, 170] have shown that classification of different objects could benefit from different neighborhood sizes, emphasising the need for data-driven approaches to determine optimal neighborhoods. Some works have addressed this issue by moving to multi-scale neighborhood methods, extracting multi-scale features at different neighborhood sizes [27, 164]. Multi-scale methods allow the classifier to select the most suitable scale for each individual point in 3D point clouds, avoiding to rely on a single suboptimal neighborhood definition. In practice, multi-scale approaches rely on the usual k-nearest neighbors definition, although considering multiple values for k. Other works have chosen a different path. Rather than focusing on the search of the optimal value of k, some methods try to define the optimal neighborhood based on geometric features themselves [29]. Typically, this consists in computing the geometric features at different scales and then choosing which neighborhood has produced the "best" features, based on some criterion: Shannon entropy [164, 171], surface variation [172], local curvature [173], local density [174], etc. For instance, the omnivariance criterion (defined below in Table 3.1) can be used [29], *i.e.*, choosing the number of neighbors that minimizes the omnivariance of the neighborhood.

3.2.2 Geometric Features

Geometric features refer to point cloud descriptors based on local neighborhood structures. These handcrafted features were previously the standard for point cloud analysis. Since estimating the surface curvature and identifying the points of maximum curvature help to detect edge points [175]. Moreover edges on segmented surfaces of 3D objects can be found using curvature [176] while local information variance [177] helps to classify TLS point clouds. Techniques, such as feature extraction and sampling, were presented to build an indoor mapping system [178]. These geometric descriptors are fed into statistical models for various tasks, *e.g.* point cloud classification [179, 180]. In this section we will describe some of the most commonly used geometric features.

Covariance-based features Most geometric features are derived from the covariance matrix of the points inside the neighborhood [27], although other handcrafted features have been introduced in the literature [164]. More precisely, geometric features first build upon the 3×3 covariance matrix computed on all the points belonging to the neighborhood of interest. The purpose of the covariance matrix is to evaluate how each

Table 3.1: Commonly used handcrafted geometric features and their interpretation.

Features	**Expression**	**Interpretation**
Linearity (L)	$(\lambda_1 - \lambda_2)/(\lambda_1)$	Increases around linear objects. *Example: lamp posts, electric cables.*
Planarity (P)	$(\lambda_2 - \lambda_3)/(\lambda_1)$	Increases near 2D planar structures. *Example: walls, floors, roofs.*
Scattering (S_c)	(λ_3/λ_1)	Increases around spherical structures. *Example: helps detect rain gutters.*
Omnivariance (O)	$\sqrt[3]{\lambda_1 \cdot \lambda_2 \cdot \lambda_3}$	Describes the distribution of points in the neighbourhood. *Example: trees are not uniformly distributed in 3D neighbourhood, contrarily to smooth surfaces such as walls and floors.*
Anisotropy (A)	$(\lambda_1 - \lambda_3)/(\lambda_1)$	Increases around edges and corners. Represents how a surface behaves differently in different directions.
Sum of eigenvalues (S_e)	$\lambda_1 + \lambda_2 + \lambda_3$	Represents the total variance of the neighbourhood.
Surface variation (S_v)	$\lambda_3/(\lambda_1 + \lambda_2 + \lambda_3)$	Increases around curvature changes in the local surface.
Verticality (V)	$1 - \left\lvert \langle \overrightarrow{[0,0,1]}, \overrightarrow{e_3} \rangle \right\rvert$	Increases when the local structure has a vertical trend.
Normals (N)	$\overrightarrow{e_3}$	Defines the local surface plane and allows to distinguish surface orientation.

dimension of the local neighborhood deviates from the mean with respect to each other [181]. Let i denote the index of the query point p_i in a point cloud P and let N denote its number of neighbors. The covariance matrix Σ_i of the neighborhood around point p_i is given by:

$$\Sigma_i = \frac{1}{N} \cdot \sum_{j=0}^{N} (p_j - \overline{p}) \cdot (p_j - \overline{p})^T \quad \text{with} \quad \overline{p} = \frac{1}{N} \cdot \sum_{j=0}^{N} p_j \tag{3.1}$$

Let (e_1, e_2, e_3) denote the eigenvectors of the covariance matrix Σ_i and $\left(\lambda_1 > \lambda_2 > \lambda_3\right)$ denote its eigenvalues. From there, we can define some of the most common geometric features used for 3D point cloud classification, *e.g.* linearity, planarity, scattering... These descriptors and their formulas are detailed in Table 3.1. Another commonly used set of handcrafted features are the normals to the point cloud. For each point p_i, its normal n_i represents the direction of the perpendicular vector to the best-fitting plan of its neighborhood [29]. A classical way to compute the normals is to compute the covariance matrix, extract the eigenvectors and use $n_i = e_3$, *i.e.* define the normal as the third eigenvector [181, 182]. Normal estimation has then grown into its own topic of interest. For example, the Randomized Hough Transform is employed to estimate normals [183]. Later works use deep learning to compute normals, *e.g.* projected a discretized Hough space (representing the direction of the normals) onto a structure suitable for deep learning [184], and used deep models and multi-scale edge refinement

to improve accuracy [185]. These methods offer a compromise between precision and speed, allowing accurate estimation of normals at the cost of time. Although there are several other methods to obtain normals, their specific computation is outside the scope of this work.

Handcrafted features and 3D deep learning A few works have explored the incorporation of handcrafted features into deep models for 3D point cloud classification and segmentation. 3D deep models not only use RGB color information from images in addition to XYZ coordinates, most of them also include normals [67, 100, 186] to improve accuracy. However, some models also include additional handcrafted features. For example, PointNeXt [100] augments the XYZ coordinates and RGB information with an ad hoc "height" information, defined as the difference between the z-coordinate of the point and the minimum value of the z-coordinates for the entire point cloud. This feature significantly improves object classification accuracy by allowing the model to distinguish object sizes. In S3DIS, a widely used indoor semantic segmentation dataset [120], appending height information increases mIoU. RepSurf [162] introduces its own novel representation for encoding point clouds as input to deep models. It focuses particularly on the representation of local structures: Triangular RepSurf and Umbrella RepSurf are based on triangular meshes and umbrella curvature, respectively. They showed that the triangular representation has a limited receptive field and can lead to the misrepresentation of local structures, while the umbrella curvature actually expands the perceptive field of the models. SuperPointTransformer [107] is another notable work that leverages geometric features to represent point clouds. It clusters points into hierarchical partitions based on XYZ coordinates and handcrafted features. In addition, these features are also used to train semantic segmentation deep models. The authors report that the removal of these handcrafted features has a detrimental effect on model performance. Overall, it is noteworthy that geometric features are still relevant for point cloud segmentation in the era of deep learning. In this work, we investigate the impact of geometric features on the performance of recent deep models and how to make the most of them in practice.

3.2.3 Resetting strategies

Dropout and variants Randomly resetting values to zero is a classic tool used in deep learning as regularization. Srivastava et al. [187] originally showed that the overfitting problem that arises during the training of deep models can be mitigated by implementing a technique known as Dropout. This technique consists in randomly "removing" neurons during the training of a deep model by setting their activations to zero, thereby preventing the co-adaptation of activations. Dropout has become a staple of deep learning and is an effective regularization as demonstrated through improvements in the performance of various tasks, including speech

recognition and vision. Multiple variants have been designed upon the foundations laid out by Dropout. For example, DropConnect [188] implements a similar idea, although it randomly sets *weights* to zero instead of activations. DropPath [189] randomly sets to zero activations for entire layers to prevent co-adaptation of parallel paths on very deep models and was introduced as an alternative to residual networks.

Dropping inputs Some works have also investigated resetting the *inputs* of the model to improve its robustness. In computer vision, some techniques such as CutOut randomly erase pixels inside square or rectangular selections by setting them to a constant value [190, 191]. Similarly, DropBlock [192] randomly sets entire regions of the activation map in convolutional neural networks (CNN) to zero, erasing not only the inputs, but also the higher-level features. In the context of 3D point cloud segmentation, PointNeXt [100] was the first to implement a similar technique. It introduced "color drop", a regularization that randomly resets the RGB colorimetric information to zero with a fixed probability. The authors proposed color dropping as a data augmentation strategy, and was one of the major sources of improvement in S3DIS. This strategy has now been incorporated into novel deep models that have demonstrated superiority over state-of-the-art methods in point clouds semantic segmentation, as exemplified by Swin3D [186]. However, what is missing point is that this random color drop makes deep models more robust to missing RGB data. Not all 3D point clouds are colored, and using a dropping strategy for color input makes pretrained models more versatile. In this work, we extend this idea to DropFeatures, a reset strategy that covers the full set of geometric features.

3.3 Local features for deep models

Our goal is to provide a general framework for adding handcrafted geometric features to 3D deep models for point cloud segmentation. This section will answer two questions:

1. How to compute and integrate geometric features as inputs to the model on the fly?
2. How to train models that can also be used at inference time without the geometric features?

3.3.1 Batch sampling

Let P denote a color point cloud containing N points:

$$P = \begin{vmatrix} x_1 & y_1 & z_1 & r_1 & g_1 & b_1 \\ x_2 & y_2 & z_2 & r_2 & g_2 & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N & y_N & z_N & r_N & g_N & b_N \end{vmatrix} \tag{3.2}$$

Because N is large, the entire point cloud generally does not fit in memory when training deep models. The 3D point clouds are therefore downsampled, often by voxelization. The 3D space is discretized into voxels of a given size $(\Delta x, \Delta y, \Delta z)$, generally around a few centimeters. Each point $p_i = (x_i, y_i, z_i)$ is mapped to voxel coordinates $v_i = (v_{x_i}, v_{y_i}, v_{z_i})$ where:

$$v_{x_i} = \left\lfloor \frac{x_i}{\Delta x} \right\rfloor, v_{y_i} = \left\lfloor \frac{y_i}{\Delta y} \right\rfloor, v_{z_i} = \left\lfloor \frac{z_i}{\Delta z} \right\rfloor \tag{3.3}$$

where $\lfloor * \rfloor$ denotes the floor function. In practice, only m points are fed as input to the deep network, typically $m = 24,000$. This is achieved by selecting uniformly at random a voxel v_j and its $m-1$ closest voxels. Let $\mathscr{P}_j$ be the set of points that map to the j-th voxel v_j. To preserve the diversity of the original point cloud, the final point p_j that is included in the batch is sampled uniformly amongst $\mathscr{P}_j$, *i.e.* amongst points inside the voxel v_j. In practice, this means that the inputs to the deep model are the coordinates of these m points and their RGB color information, *i.e.*:

$$X_k = \begin{vmatrix} x_{i_1} & y_{i_1} & z_{i_1} \\ x_{i_2} & y_{i_2} & z_{i_2} \\ \vdots & \vdots & \vdots \\ x_{i_m} & y_{i_m} & z_{i_m} \end{vmatrix} \quad C_k = \begin{vmatrix} r_{i_1} & g_{i_1} & b_{i_1} \\ r_{i_2} & g_{i_2} & b_{i_2} \\ \vdots & \vdots & \vdots \\ r_{i_m} & g_{i_m} & b_{i_m} \end{vmatrix} \tag{3.4}$$

It is common to normalize the coordinates by subtracting the minimum coordinate for each column. The final batch B_k is obtained by concatenation:

$$B_k = X_k \oplus C_k \tag{3.5}$$

Additional features F_k, *i.e.* geometric features such as normals, can be propagated in the same way.

3.3.2 Data augmentation

Most deep architectures leverage data augmentation to improve model performance [100, 107, 106]. Data augmentation involves applying transformations to the input of a deep model to increase the diversity of the dataset, with the aim of improving performance. In the case of 3D point clouds, data augmentation techniques often include translation, rotation, jittering and scaling. As a result, the coordinates of 3D point clouds undergo various transformations. For example, PointNeXt [100] applies random translations and rotations to the 3D point clouds when training the deep model. These transformations affect the XYZ coordinates X_k, but leave the additional RGB color information C_k untouched. However, the geometric features F_k we are interested in are neither invariant nor equivariant to these augmentations. Therefore, we must be cautious about integrating the geometric features into deep models that depend on data augmentation. In fact, since geometric features are derived from the local neighborhood of the points, applying transformations to the 3D coordinates will change the neighborhood, the covariance and thus the features. As a result, geometric features may not remain consistent with the original 3D coordinates. Consequently, we have two options for integrating geometric features into the augmentation pipeline:

1. Determine how data augmentation affects XYZ coordinates and neighborhood, and augment the features along with the points;
2. Perform data augmentation on XYZ coordinates first, and then compute geometric features on-the-fly on the augmented batch.

Option 1 is unrealistic, as some data augmentation (*e.g.* jittering) will affect the neighborhoods and therefore change the covariance matrices. For this reason, we turn to option 2: on-the-fly computation of geometric features.

3.3.3 Online geometric feature extraction

Traditionally, geometric features are computed *offline*, *i.e.* the features are computed on the entire point cloud as a preprocessing step. Previous work first determine the neighborhoods for each point, then estimate the covariance matrix and finally compute local geometric features and store them alongside the XYZ coordinates.

However, 3D point clouds are heavy data, so this approach is time and memory consuming. In addition, there are acquisition scenarios where point clouds are acquired incrementally, *e.g.* using mobile Lidar in autonomous driving [193] or indoor navigation [133]. It also means that if the point cloud dataset changes, the preprocessing has to be done again, as any change in the 3D coordinates alters the whole structure of the point cloud, requiring the computation of neighborhoods and covariance matrices. To deal with this, we used the nearest neighbors and the previous Eq. (3.1) to compute the covariance matrix directly on the batch of point clouds X_k.

Let B_k be the k-th batch of points in the dataset, with m being the batch size. Let $(x_i, y_i, z_i)_{1 \leq i \leq m}$ denote the 3D coordinates of the point selected in the batch, (r_i, g_i, b_i) denote their color information, and $f_i = (f_i^1, f_i^2, \ldots, f_i^l)$ represent the vector of l additional geometric features (*e.g.* linearity, planarity, normals, etc.). The batch B_k is obtained by concatenating the three matrices X_k, C_k, F_k as follows:

$$B_k = \underbrace{\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_m & y_m & z_m \end{vmatrix}}_{X_k} \oplus \underbrace{\begin{vmatrix} r_1 & g_1 & b_1 \\ r_2 & g_2 & b_2 \\ \vdots & \vdots & \vdots \\ r_m & g_m & b_m \end{vmatrix}}_{C_k} \oplus \underbrace{\begin{vmatrix} f_1^1 & f_1^2 & \cdots & f_1^l \\ f_2^1 & f_2^2 & \cdots & f_2^l \\ \vdots & \vdots & \vdots & \vdots \\ f_m^1 & f_m^2 & \cdots & f_m^l \end{vmatrix}}_{F_k} \tag{3.6}$$

where $\oplus$ denotes the matrix column-wise concatenation.

Instead of relying on a precomputed online set of features F_k, we compute the geometric features on-the-fly directly from the 3D coordinates of the batch X_k. This means that, instead of finding the nearest neighbors in the original (full-density) point clouds, we find the k-nearest neighbors in the subsampled point clouds for each point of the batch. We then follow the mathematical expression described in Table 3.1 to determine the geometric feature vector $F_{k,i}$ for each point p_i, where the neighborhood is restricted to points belonging to the batch. As the points correspond to a subsampled point cloud, we have found that a neighborhood of $k = 50$ points is sufficient in practice. This allows us to integrate geometric features in combination with any data augmentation. For example, the "jitter" augmentation, which adds random noise to the XYZ point coordinates, can be used with our online features, because we compute the features *after* the transformation. We implement this on-GPU using PyTorch3D [194], which adds a negligible overhead both in train and inference.[1]

Normal exception As the normal vectors are prevalent features in the 3D deep learning literature, many datasets already integrate them natively. Most data augmentations have a natural effect on the normals, *i.e.*

[1]Thanks to asynchronous batch loading, features are computed even before the previous batch has finished being processed by our models. Overhead for LPS_cV is inferior to 2 seconds when testing over a full point cloud takes around 1 minutes with Swin3D [186].

rotating points also rotate the normals in the same way. For this reason, and for the sake of comparison with existing methods, we either use the provided normals or recompute them offline on the full point cloud for greater accuracy.

3.3.4 Features reset strategy

DropFeatures is a regularization technique for 3D deep models that relies on randomly resetting features in the inputs. As stated previously, geometric features can be time-consuming and computationally expensive. Therefore, they may not always be available at test time, when we want to use a trained model to predict labels for new point clouds. However, if we train a deep network on XYZ coordinates and features, and then remove the features at test time, the performance drops significantly. Therefore, our objective is to develop a reset method that ensures that the model performs well even when it is used without access to the geometric features.

DropFeatures We define DropFeatures as the reset operation that randomly replaces the row values of the matrix F_k with zeros. The reset operation is applied to the features f_k of the batch B_k as follows:

$$F'_k = \text{reset}(F_k, p) = \delta \cdot F_k \tag{3.7}$$

where δ is a random variable distributed according to Bernoulli's law with parameter p, *i.e.* $\delta = 0$ with probability p and 1 otherwise. Compared to other "drop" reset techniques, DropFeatures resets the features for *the entire batch*. This is because the features of neighboring points are correlated. So, the model might otherwise be able to reconstruct the missing features based on the other points in the batch. If $p = 0$, DropFeatures has no effect and is the same as training the model using XYZ, RGB and the features. If $p = 1$, it is the same as training the model with XYZ and RGB only. p is the only hyperparameter of the method, and its tuning is discussed in the experiments section (Section 3.4.3).

3.4 Increase deep accuracy with handcrafted features

3.4.1 Experimental setup

In this section, we will describe the datasets and models used to see how well handcrafted geometric features can improve the accuracy of 3D point cloud semantic segmentation when performed with deep models. Despite the willingness to use only XYZ coordinates, to be as close as possible to older land surveyor point clouds, we include RGB in our experiments for comparison with literature baselines. Note that color dropout occurs with

Table 3.2: Indoor datasets used to evaluate the contributions of geometric features on point clouds segmentation. *Note that 3DSES [1] is composed of three dataset variants, each with a different number of points (see Chapter 4 for more details).*

Name	Environment	Source	Colors	Points
S3DIS [120]	Indoor	Camera	✅	273
ScanNet [121]	Indoor	Camera	✅	242
3DSES [1]	Indoor	TLS	✅	[65,216,413]

all deep models tested in this chapter.

We focus on three datasets: S3DIS [120], ScanNet [121], and 3DSES [1]. These three datasets all contain indoor scenes, but they cover different types of areas, have been acquired with different sensors and have different levels of point density (see Table 3.2).

S3DIS [120] covers six indoor areas for a total of 273 million of points, 6000 m^2 and 13 classes. The 3D data and RGB information were collected simultaneously using structured-light cameras. This is a staple of 3D deep learning, but the evaluation focuses mostly on area 5, which is not necessarily representative of generalization. Instead, we follow [107] and instead perform an exhaustive 6-fold cross-validation over all the areas.

ScanNet [121] is a dataset consisting of 1513 RGB-D scans acquired within 707 indoor rooms. It contains both 2D and 3D data with a high point density. It has been annotated at the point level for 20 classes. ScanNet is divided into three parts: one for training, one for validation, and one for testing. As the test labels for ScanNet are not public and are used for scoring on a private leaderboard, all the results presented are from the validation set.

3DSES [1] is an indoor Lidar dataset designed for two point cloud tasks: 3D semantic segmentation and 3D reconstruction. Further details on the 3DSES dataset are provided in Chapter 4. There are three versions of 3DSES, each with different areas and semantic diversities. 3DSES has a dual annotation format: semantic labels and semantic pseudo-labels, obtained using a model-to-cloud alignment algorithm. For the Gold 🥇 and Silver 🥈 versions, the deep models were trained on real labels and then tested on hidden real labels from the 3DSES test set. Bronze 🥉 contains only pseudo-labels to train the models, and evaluations were always performed on real labels from the test set.

Geometric features computation We compute the geometric features online using a pipeline based on

PyTorch3D [194] to compute the features at the batch level. As PointNeXt and Swin3D sample batches after voxelization, we defined neighborhoods using k-nearest neighbors with $k = 50$. This allows us to have enough points to compute a robust covariance matrix, while having a moderate computational cost. For ScanNet and S3DIS, we used precomputed normals from CloudCompare [4] with a spherical neighborhood radius of 7 cm.

Deep models To assess the impact of geometric features on deep models, we considered different architectures covering the main classes of models and explored different parameter counts. We chose PointNeXt-S [100] as the smallest deep model for our experiments, with a parameter count close to 1 million. It is a popular model with a simple design, while being also modest in computational requirements. We compared it with its larger counterpart PointNeXt-XL, which has 44 million parameters. To compare these models with a state-of-the-art high-performance architecture, we also included Swin3D [186] in our benchmark. Swin3D is a Transformer-based architecture with 60 million parameters.

Hyperparameters For the hyperparameters, we kept the original setup as specified by the authors. We trained Swin3D with an AdamW optimizer and 100 epochs, using the original batch size of 3 for S3DIS and ScanNet, and 6 for 3DSES, with a learning rate of $l_r = 0.001$ for S3DIS and 3DSES, and 0.006 for ScanNet. We trained PointNeXt (S and XL versions) using the AdamW optimizer for 100 epochs, a batch size of 32 and a learning rate of $l_r = 0.01$ for S3DIS, $l_r = 0.05$ for 3DSES, and $l_r = 0.001$ for ScanNet. For both models, we used a cosine learning rate schedule and a standard cross-entropy loss. All models were trained on an NVIDIA V100 (32GB) for S3DIS and ScanNet. For 3DSES, we used an NVIDIA RTX A6000 (49G) for Swin3D and an NVIDIA RTX 6000 (23G) for PointNeXt-S. Training the larger Swin3D models on the biggest dataset (ScanNet) take less than 100 hours. Smaller models and/or datasets take significantly less time *e.g.* 20 hours on S3DIS and PointNeXt-S. We reported model performance based on the last epoch. We observe that for these three types of architectures, the Intersection over Union (IoU) metric converges to a fixed value by the end of training, thereby reducing variance.

In the following section, we present the results of the semantic segmentation using different deep network architectures on the S3DIS, ScanNet and 3DSES datasets. We report the commonly used segmentation metrics: Intersection over Union (IoU), Overall Accuracy (OA) and Average Accuracy (AA). These metrics are detailed in Appendix A.

Table 3.3: Segmentation metrics obtained with a 6-fold cross-validation on S3DIS. Metrics obtained using different geometric features and different models are compared to the RGB baseline. **Green** indicates an improvement compared to the baseline, while **red** indicates a decrease. Mean intersection over union (mIoU), overall accuracy (OA), average accuracy (AA).

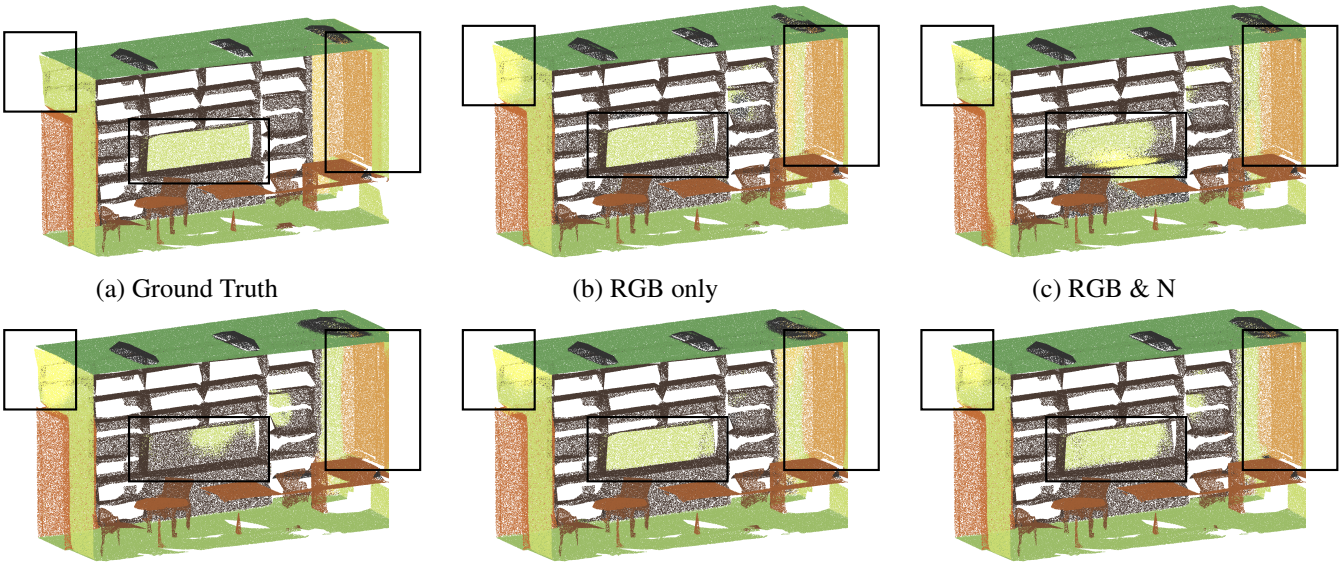
				PointNeXt-S			PointNeXt-XL			Swin3D-L		
RGB	N	LPS_cV	OAS_eS_v	**OA**	**AA**	**mIoU**	**OA**	**AA**	**mIoU**	**OA**	**AA**	**mIoU**
✅	❌	❌	❌	87.71	78.88	66.72	89.77	83.82	73.24	90.55	82.28	74.08
✅	✅	❌	❌	$87.56_{-0.15}$	$79.33_{+0.46}$	$66.84_{+0.12}$	$89.85_{+0.08}$	$83.39_{-0.43}$	$73.12_{-0.12}$	$91.07_{+0.52}$	$\mathbf{84.07}_{+1.79}$	$75.17_{+1.09}$
✅	❌	✅	❌	$87.96_{+0.25}$	$\mathbf{80.22}_{+1.35}$	$\mathbf{67.81}_{+1.09}$	$90.43_{+0.66}$	$83.24_{-0.58}$	$73.47_{+0.23}$	$\mathbf{91.27}_{+0.72}$	$83.36_{+1.07}$	$\mathbf{75.45}_{+1.36}$
✅	❌	❌	✅	$\mathbf{87.98}_{+0.27}$	$79.22_{+0.34}$	$67.01_{+0.29}$	$90.34_{+0.57}$	$83.46_{-0.36}$	$73.68_{+0.44}$	$90.66_{+0.11}$	$83.13_{+0.85}$	$74.50_{+0.41}$
✅	✅	✅	✅	$87.89_{+0.18}$	$79.86_{+0.98}$	$67.69_{+0.97}$	$\mathbf{90.69}_{+0.93}$	$\mathbf{84.02}_{+0.20}$	$\mathbf{74.32}_{+1.08}$	$90.89_{+0.34}$	$83.90_{+1.62}$	$75.30_{+1.22}$

3.4.2 Contribution of geometric features

S3DIS Table 3.3 shows that geometric features are helpful for the three models on S3DIS. For PointNeXt-S, the LPS_cV combination achieved the highest mIoU score of 67.81%. The OAS_eS_v combination achieved the highest OA (87.98%). Overall, the combination of all features consistently improves the segmentation metrics. While the model trained with all features performs better than the RGB baseline (*e.g.* +0.97% mIoU), the addition of all features does not always perform best. This is the case for PoinNeXt-XL but not for PointNeXt-S, although the gap to the second best combination is small. As with PointNeXt-S, adding the LPS_cV features helps Swin3D to get the best mIoU. We show some qualitative results in Fig. 3.3. Fig. 3.4 shows that handcrafted features benefits are limited on large objects (such as "ceiling", "floor" and "wall"). However, these features can be helpful with less represented objects in the dataset such as "bookcase", "table", "window".

ScanNet The results on ScanNet presented in Table 3.4 are contrasted. For PointNeXt-S, features do not help the model, and the inclusion of normals actually results in the lowest segmentation accuracy. PointNeXt-XL does not benefit from geometric features either, with all combinations being outperformed by RGB alone. Swin3D performs slightly better with color information and normals, with +0.5% in mIoU and almost identical OA and AA. Overall, the inclusion of features has minimal effect on Swin3D's performance. It is possible that this large Transformer-based model has enough parameters – and the dataset enough samples, as ScanNet is significantly larger than 3DSES and S3DIS – to learn effective representations without relying on handcrafted features.

3DSES Results on 3DSES are shown in Table 3.5. For PointNeXt-S, the geometric features improve almost



(a) Ground Truth (b) RGB only (c) RGB & N

(d) RGB & LPS_cV (e) RGB & OAS_eS_v (f) RGB & N & LPS_cV & OAS_eS_v

Figure 3.3: Qualitative segmentation results for PointNeXt-S on Area 5 office 4 from S3DIS using various sets of inputs. This shows that including geometrical features help segment more accurately planar and linear objects (here, column and window). Legend: Covering in dark green, floor in green, wall in light green, beam in light yellow, column in light orange, window in light brown, door in brown, table in brown, chair in brown, bookcase in light black, clutter in dark black

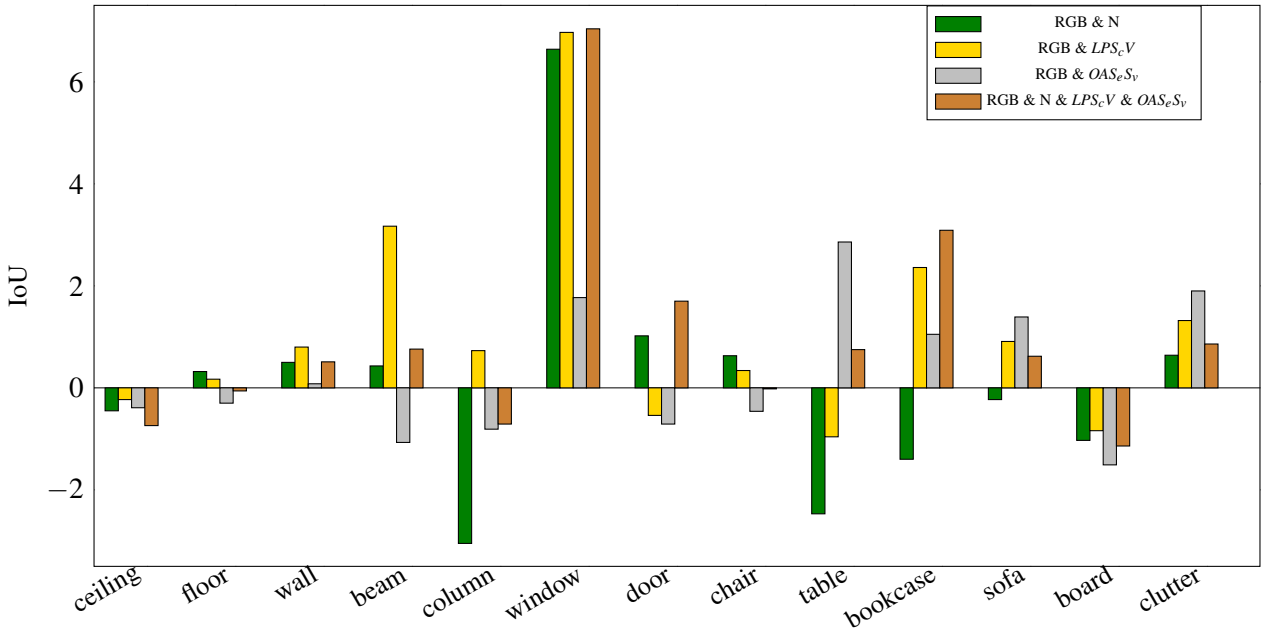


Figure 3.4: Histogram representing the Intersection over Union by class for a 6-fold validation on S3DIS with PointNeXt-S and several handcrafted features. RGB IoU is subtracted from all handcrafted results for better visualization.

Table 3.4: Segmentation metrics obtained on the ScanNet validation set. Metrics obtained using different geometric features and different models are compared to the RGB baseline. **Green** indicates an improvement compared to the baseline, while **red** indicates a decrease. Mean intersection over union (mIoU), overall accuracy (OA), average accuracy (AA).

				PointNeXt-S			PointNeXt-XL			Swin3D-L		
RGB	N	LPS_cV	$OASS_v$	OA	AA	mIoU	OA	AA	mIoU	OA	AA	mIoU
✅	❌	❌	❌	**86.36**	**73.12**	**64.34**	**89.46**	**79.45**	**71.17**	**90.76**	**82.91**	74.76
✅	✅	❌	❌	85.62$_{-0.74}$	72.05$_{-1.07}$	63.10$_{-1.24}$	89.11$_{-0.35}$	79.27$_{-0.18}$	71.13$_{-0.04}$	90.74$_{-0.02}$	82.90$_{-0.01}$	**75.28**$_{+0.52}$
✅	❌	✅	❌	85.94$_{-0.42}$	72.75$_{-0.37}$	63.64$_{-0.70}$	88.92$_{-0.54}$	78.84$_{-0.61}$	70.22$_{-0.95}$	90.68$_{-0.08}$	82.56$_{-0.35}$	74.42$_{-0.34}$
✅	❌	❌	✅	86.03$_{-0.33}$	72.38$_{-0.74}$	63.71$_{-0.63}$	89.36$_{-0.10}$	79.04$_{-0.41}$	70.62$_{-0.55}$	90.25$_{-0.51}$	80.98$_{-1.93}$	72.53$_{-2.23}$
✅	✅	✅	✅	85.98$_{-0.38}$	72.72$_{-0.40}$	63.80$_{-0.54}$	89.07$_{-0.39}$	79.11$_{-0.34}$	70.74$_{-0.43}$	90.74$_{-0.02}$	82.84$_{-0.07}$	74.81$_{+0.05}$

Table 3.5: Segmentation metrics obtained on the 3DSES test set. Metrics obtained using different geometric features and two models are compared with the RGB baseline. *Note that results from PointNeXt-S are the results of 3 runs and standard deviation is printed in Table B.1, Table B.2 Table B.3.* **Green** indicates an improvement compared to the baseline, while **red** indicates a decrease. Mean intersection over union (mIoU), overall accuracy (OA), average accuracy (AA).

				PointNeXt-S			Swin3D		
	RGB	LPSV	OASS	OA	AA	mIoU	OA	AA	mIoU
Gold	✅	❌	❌	82.18	36.17	30.14	**89.76**	**78.30**	**48.99**
	✅	✅	❌	**86.49**$_{+4.31}$	**37.58**$_{+1.41}$	**32.55**$_{+2.42}$	82.41$_{-7.35}$	73.04$_{-5.26}$	48.32$_{-0.67}$
	✅	❌	✅	85.82$_{+3.64}$	35.93$_{-0.24}$	31.22$_{+1.08}$	86.54$_{-3.22}$	70.05$_{-8.25}$	45.44$_{-3.55}$
Silver	✅	❌	❌	92.29	63.30	57.34	91.44	85.16	59.61
	✅	✅	❌	**93.89**$_{+1.60}$	**65.30**$_{+2.00}$	**60.73**$_{+3.35}$	91.09$_{-0.35}$	85.20$_{+0.04}$	58.86$_{-0.75}$
	✅	❌	✅	92.73$_{+0.44}$	61.94$_{-1.37}$	56.50$_{-0.84}$	**94.06**$_{+2.62}$	**85.91**$_{+0.75}$	**62.47**$_{+2.86}$
Bronze	✅	❌	❌	**94.47**	67.94	62.48	94.32	**93.78**	68.19
	✅	✅	❌	93.98$_{-0.49}$	**71.49**$_{+3.55}$	**65.24**$_{+2.76}$	95.01$_{+0.69}$	91.81$_{-1.97}$	69.54$_{+1.35}$
	✅	❌	✅	94.00$_{-0.47}$	70.08$_{+2.14}$	63.74$_{+1.26}$	**95.55**$_{+1.23}$	93.36$_{-0.42}$	**71.19**$_{+3.00}$

all metrics. The combination of color information and LPS_cV gives the best mIoU (with $> 2\%$ increase) for all versions of 3DSES, as well as the best AA for all versions. The OAS_eS_v features help the model for Gold and Bronze variants, but slightly reduce the mIoU performance mIoU for Silver. On the more challenging Gold variant, Swin3D trained with features performs below the RGB baseline, possibly due to overfitting this large model to a smaller dataset. In comparison, on the larger Bronze model, the combination of *RGB* and LPS_vV, and the combination of *RGB* and OAS_eS_v, significantly improves mIoU for both the small PointNeXt-S and the large Swin3D models. The intermediate Silver variant is more contrasted as the best combination of features depends on the model, although an increase in mIoU of around 3% is achieved for both models. We hypothesize than the features are mostly beneficial on smaller datasets, *i.e.* 3DSES and S3DIS as they introduce "expert knowledge" in the models that cannot be otherwise learnt. This is especially true for small models: on 3DSES, the small PointNext-S benefits from geometric features in all data regimes, but Swin3D benefits from the same features only on the larger Silver and Bronze variants. The results in this section show that features can help deep models enhance their performance. The question then arises as to what happens if these features are not available during the testing phase.

3.4.3 Reset percentage of DropFeatures

In this section, we evaluate the impact of removing features during the testing phase. We conduct an ablation study to determine what percentage of resetting results in a better mIoU, and then compare models trained with and without our DropFeatures approach.

Optimal reset percentage Fig. 3.5 shows the evolution of mIoU on S3DIS when training PointNeXt-S for different reset percentages using the widely used LPS_cV feature combination [108, 107]. The results are the mean of a 6-fold cross-validation, that we ran three times to account for training variability. The blue curve represents the results for PointNeXt-S trained on RGB+ LPS_vV and tested with the same features, while the red curve shows test results with LPS_vV features set to 0 during inference. Note that 0% represents the baseline model trained with colorimetric information (RGB) and LPS_cV features without DropFeatures, while 100% represents the baseline model trained with RGB only. As expected, lower reset percentages result in a sharp drop in performance when the model is used for inference without LPS_vV features. In fact, this configuration achieves the worst mIoU of all setups: 13.38%, representing a gap of -54.22% mIoU compared to the model tested with LPS_vV. Small reset percentages (5%, 15% and 25%) also result in performance losses, albeit smaller (respectively -1.92%, -0.62% and -0.39% mIoU compared to the models with features). For higher reset

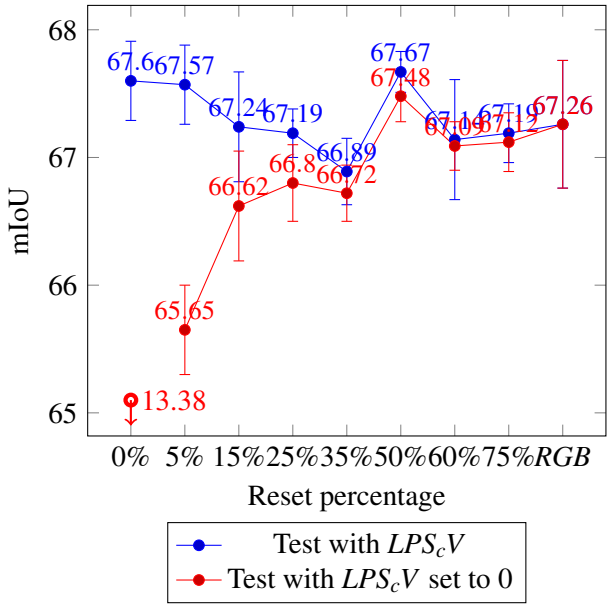


Figure 3.5: **Ablation study.** Contribution of the reset percentage for our DropFeatures strategy. Mean Intersection over Union (mIoU) obtained with PointNeXt-S, evaluated through *three rounds of 6-fold cross-validation* on the S3DIS dataset. On blue: PointNeXt-S was tested with *RGB* and LPS_cV. The red curve shows the results obtained by the models with LPS_cV set to 0.

Table 3.6: Contribution of the reset percentage on S3DIS with PointNeXt-S (left side : test with *RGB* and LPS_cV and right side : test with *RGB* and LPS_cV set to 0). Segmentation metrics obtained *from performing three rounds of 6-fold cross-validation*. Standard deviation is presented in Fig. 3.5. Mean intersection over union (mIoU), overall accuracy (OA), average accuracy (AA). "RGB 0000" indicates that features are set to 0 during inference.

			RGB LPS_cV			**RGB 0000**		
RGB	**LPS_cV**	**DropFeatures**	**OA**	**AA**	**mIoU**	**OA**	**AA**	**mIoU**
✅	❌	❌	87.94	79.31	67.26	❌	❌	❌
✅	✅	❌	87.89	**79.85**	67.60	36.62	23.30	13.38
✅	✅	5%	**88.14**$_{+0.25}$	79.72$_{-0.13}$	67.57$_{-0.03}$	87.19$_{+52.12}$	78.66$_{+57.87}$	65.65$_{+53.94}$
✅	✅	15%	87.98$_{+0.10}$	79.60$_{-0.25}$	67.24$_{-0.37}$	87.67$_{+52.61}$	79.17$_{+58.36}$	66.62$_{+54.91}$
✅	✅	25%	87.88$_{-0.01}$	79.76$_{-0.09}$	67.19$_{-0.42}$	87.67$_{+52.61}$	79.39$_{+58.58}$	66.80$_{+55.09}$
✅	✅	35%	87.68$_{-0.21}$	79.17$_{-0.68}$	66.89$_{-0.71}$	87.59$_{+52.53}$	79.09$_{+58.28}$	66.72$_{+55.01}$
✅	✅	50%	88.10$_{+0.21}$	79.74$_{-0.11}$	**67.68**$_{+0.08}$	**88.01**$_{+52.94}$	**79.68**$_{+58.87}$	**67.48**$_{+55.78}$
✅	✅	60%	87.93$_{+0.06}$	79.52$_{-0.54}$	67.14$_{-0.34}$	87.90$_{+52.84}$	79.54$_{+58.73}$	67.09$_{+55.39}$
✅	✅	75%	87.95$_{+0.06}$	79.40$_{-0.45}$	67.19$_{-0.41}$	87.91$_{+52.94}$	79.38$_{+58.57}$	67.12$_{+55.41}$

Table 3.7: Segmentation metrics obtained on a *single 6 fold on S3DIS*. Metrics obtained using various geometric features and PointNeXt-S are compared to a training without DropFeatures. **Green** indicates an improvement compared to the baseline, while **red** indicates a decrease. Mean intersection over union (mIoU), overall accuracy (OA), average accuracy (AA). "RGB Features set to 0" indicates that features are set to 0 during inference.

					RGB + Features			**RGB + Features set to 0**		
RGB	N	LPS_cV	OAS_eS_v	DropFeatures	**OA**	**AA**	**mIoU**	**OA**	**AA**	**mIoU**
✅	❌	❌	❌	❌	87.71	78.88	66.72	❌	❌	❌
✅	✅	❌	❌	❌	87.56	**79.33**	66.84	33.63	20.23	11.00
✅	✅	❌	❌	50%	**87.92**$_{+0.36}$	79.26$_{-0.07}$	**66.90**$_{+0.06}$	**87.93**$_{+54.30}$	**79.28**$_{+59.05}$	**66.90**$_{+55.90}$
✅	❌	❌	✅	❌	87.98	79.22	67.01	36.14	18.26	10.51
✅	❌	❌	✅	50%	**88.14**$_{+0.16}$	**79.81**$_{+0.59}$	**67.16**$_{+0.15}$	**88.12**$_{+51.98}$	**79.79**$_{+61.52}$	**67.07**$_{+56.57}$
✅	✅	✅	✅	❌	**87.89**	**79.86**	**67.69**	27.10	15.89	7.79
✅	✅	✅	✅	50%	87.82$_{-0.07}$	78.95$_{-0.91}$	66.81$_{-0.89}$	**87.76**$_{+60.66}$	**78.90**$_{+63.00}$	**66.67**$_{+58.89}$

percentages, the gap decreases as the model becomes more robust to missing features. The reset percentage that gives the highest mIoU (with a small standard deviation) is 50%, with 67.48% mIoU. We observe that this reset probability results in a superior model compared to *RGB* alone and, indicating that our random reset strategy helps to improve model performance even when features are not available during testing. DropFeatures with 50% is also better or on-par with the baseline for the other metrics, as reported in Table 3.6. In practice, note that models trained without DropFeatures exhibit no robustness to missing features. DropFeatures perform an "interpolation" between the RGB and the RGB+ LPS_cV baselines. When tested in the missing features scenario, all models trained with our DropFeatures strategy dramatically improve the performance of PointNeXt-S by about +50% mIoU compared to the extremely low mIoU of the baseline (13.38% mIoU), showing that it achieves the desired robustness. Based on these results, we use a reset probability of 50% for the next experiments.

3.4.4 Contribution of DropFeatures

S3DIS

First, we report in Table 3.7 the results of several PointNeXt-S models trained on S3DIS using different combinations of features – *e.g.* normals, OAS_eS_c, etc. – with and without DropFeatures ($p = 0.5$). As before, the models trained without DropFeatures dramatically underperform when tested in the missing feature scenario. In practice, the deep net collapses to always predicting predict the same class. This is expected because the model is "out-of-distribution" as it was never trained with all feature inputs set to zero. In the missing features scenario,

Table 3.8: The contribution of our "DropFeatures" strategy is evaluated on LPS_cV features using segmentation metrics from a 6-fold validation on S3DIS. **Green** indicates an improvement compared to the baseline, while **red** indicates a decrease. Mean intersection over union (mIoU), overall accuracy (OA), average accuracy (AA). "RGB 0000" indicates that features are set to 0 during inference.

				RGB LPS_cV			RGB 0000		
	RGB	**LPS_cV**	**DropFeatures**	**OA**	**AA**	**mIoU**	**OA**	**AA**	**mIoU**
PointNeXt-XL	✅	❌	❌	89.77	**83.82**	73.24	❌	❌	❌
	✅	✅	❌	**90.43**$_{+0.66}$	83.24$_{-0.58}$	**73.47**$_{+0.23}$	20.82	11.44	3.24
	✅	✅	50%	89.58$_{-0.19}$	83.64$_{-0.18}$	73.05$_{-0.19}$	**89.45**$_{+68.63}$	**83.55**$_{+72.11}$	**72.89**$_{+69.65}$
Swin3D	✅	❌	❌	90.55	82.28	74.08	❌	❌	❌
	✅	✅	❌	**91.27**$_{+0.72}$	**83.36**$_{+1.08}$	**75.45**$_{+1.37}$	75.62	48.04	39.31
	✅	✅	50%	91.12$_{+0.57}$	83.13$_{+0.85}$	75.02$_{+0.94}$	**91.07**$_{+15.46}$	**83.01**$_{+34.97}$	**74.89**$_{+35.58}$

models trained with DropFeatures significantly outperform their respective baselines, showing that DropFeatures helps model robustness at test time (right side of Table 3.7). In addition, models trained with a combination of RGB and geometric features perform better than the RGB baseline (+0.06% mIoU with "normals" and +0.15% mIoU with OAS_eS_v). Overall, training with DropFeatures offers the best trade-off, with on-par or better performance in the feature-present scenario and significantly better performance in the feature-absent scenario.

These observations still partially hold for larger models, such as PointNeXt-XL and Swin3D, as reported in Table 3.8. For these two models, while DropFeatures does not increase mIoU in the available features scenario, it still significantly boosts performance in the missing features scenario with a +69.65% mIoU increase for PointNeXt-XL. We observe similar trends for Swin3D, for which our DropFeatures strategy increases mIoU by +35.58% when testing without features.

ScanNet We report in Table 3.9. DropFeatures results on the ScanNet dataset.

As observed before (see Section 3.4.2), the benefit of geometric features is contrasted on this dataset. For the two versions of PointNeXt (S and XL v), geometric features do not improve mIoU, even when combined with DropFeatures. The only improvement occurs in OA with the XL version. On the other hand, the combination of DropFeatures with Swin3D performs a slightly helpful regularization, resulting in a small but consistent performance boost across all metrics, *e.g.* +0.40% increase in mIoU in the features available scenario. Once again, DropFeatures shines in the missing features scenario. While the performance drop is smaller than for S3DIS, DropFeatures remains essential for the models to be robust to missing features, especially for the large PointNeXt-XL where it improves mIoU by 48.38%.

Table 3.9: The contribution of our "DropFeatures" strategy is evaluated on LPS_cV features using segmentation metrics on the ScanNet validation set. **Green** indicates an improvement compared to the baseline, while **red** indicates a decrease. Mean intersection over union (mIoU), overall accuracy (OA), average accuracy (AA). "RGB 0000" indicates that features are set to 0 during inference.

				RGB LPS_cV			RGB 0000		
	RGB	LPS_cV	DropFeatures	OA	AA	mIoU	OA	AA	mIoU
PointNeXt-S	✅	❌	❌	**86.36**	**73.12**	**64.34**	❌	❌	❌
	✅	✅	❌	85.94$_{-0.42}$	72.75$_{-0.37}$	63.64$_{-0.70}$	74.10	45.85	38.95
	✅	✅	50%	85.82$_{-0.54}$	72.38$_{-0.74}$	63.22$_{-1.12}$	**83.69**$_{+9.59}$	**67.95**$_{+22.10}$	**58.10**$_{+19.15}$
PointNeXt-XL	✅	❌	❌	86.46	**79.45**	**71.17**	❌	❌	❌
	✅	✅	❌	88.92$_{+2.46}$	78.84$_{-0.61}$	70.22$_{-0.95}$	59.48	16.85	12.07
	✅	✅	50%	**88.98**$_{+2.52}$	78.51$_{-0.94}$	70.08$_{-1.09}$	**84.94**$_{+25.46}$	**68.29**$_{+51.44}$	**60.45**$_{+48.38}$
Swin3D	✅	❌	❌	90.76	82.91	74.76	❌	❌	❌
	✅	✅	❌	90.68$_{-0.08}$	82.56$_{-0.35}$	74.42$_{-0.34}$	85.99	72.07	62.84
	✅	✅	50%	**90.92**$_{+0.16}$	**83.02**$_{+0.11}$	**75.16**$_{+0.40}$	**90.93**$_{+4.94}$	**82.96**$_{+10.89}$	**75.11**$_{+12.27}$

3DSES We report in Table 3.10 DropFeatures results on the 3DSES dataset, for its three variants. The results are similar to the two previous datasets. In the missing features scenario, our DropFeatures strategy helps both PointNeXt-S and Swin3D to reduce the performance gap with their counterparts tested with the geometric features. In particular, Swin3D achieves on-par performance without the features compared to the baseline, but tends to overfit due to its size.

It is worth noting that, on the smaller 3DSES Gold dataset(Table 3.10), Swin3D forgets many classes when tested without LPS_cV, achieving only 36.75% OA. However, when using our DropFeatures strategy, Swin3D forgets fewer classes and is able to achieve 87.05% OA. We can observe a similar gap for some other classes (such as "Covering", "Railing", "Window"), as shown in Fig. 3.6. Focusing on "Wall", we see that the model trained with LPS_cV and tested without achieves a poor IoU, while the model trained with the DropFeatures strategy achieves an IoU close to the model trained and tested with LPS_cV. For PointNeXt-S, DropFeatures also reduces the gap and in missing scenario our strategy enable PointNeXt to RGB baseline for Bronze variant. Fig. 3.7 presents the results for Swin3D on 3DSES-Bronze and highlight that geometric feature can reduce segmentation error on some objects such as heater, door or wall.

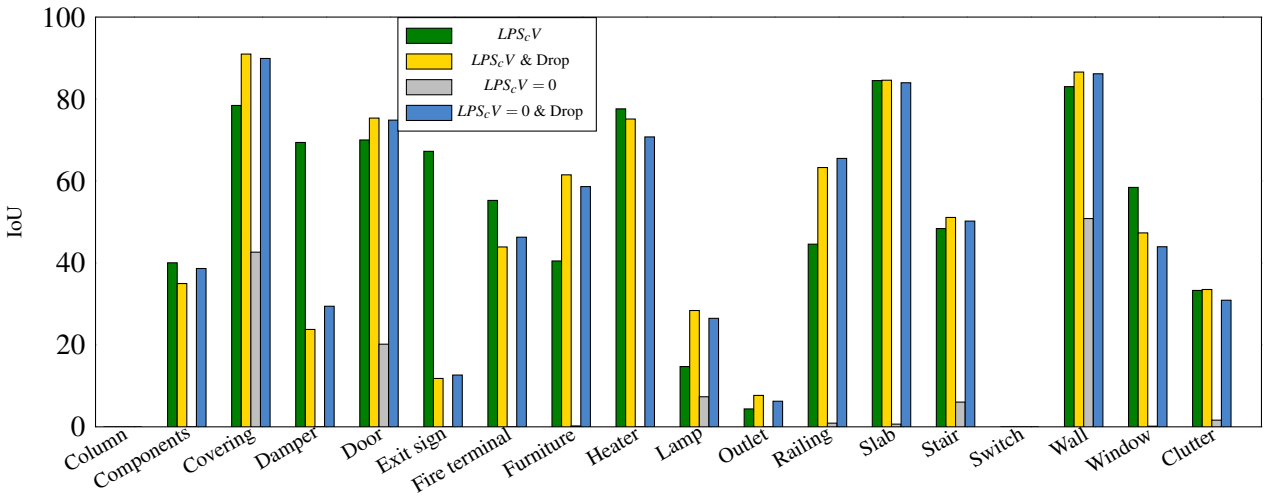
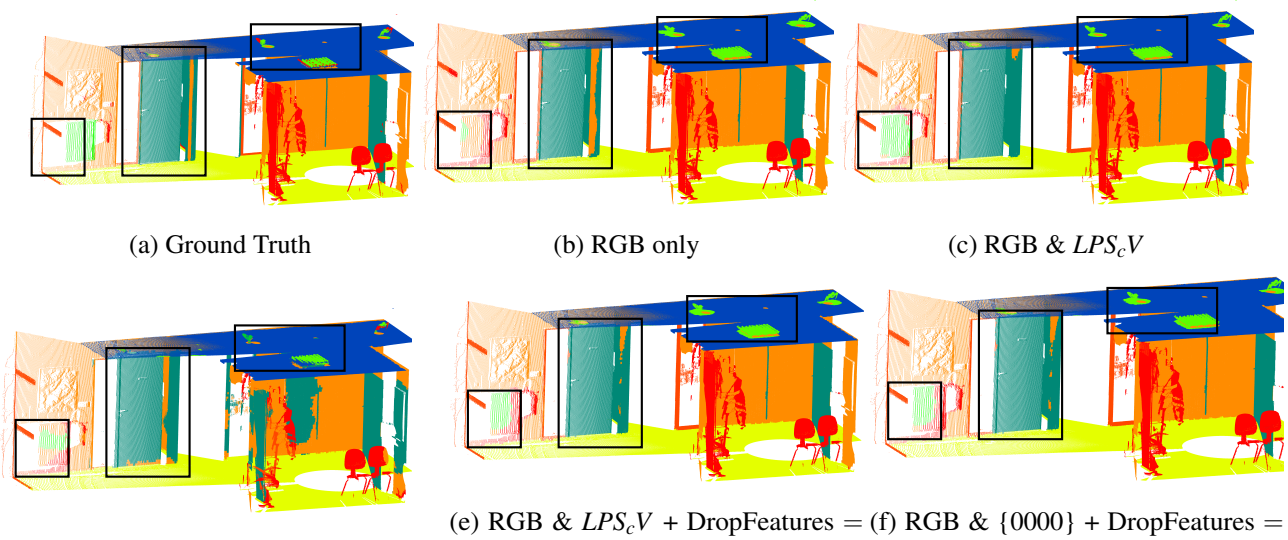


Figure 3.6: Intersection over union (IoU) by classes obtained with Swin3D trained under various configurations on the 3DSES Gold dataset. **Green** is the $RGB + LPS_cV$ model, **yellow** is the $RGB + LPS_cV$ model trained with "DropFeatures" ($p = 0.5$), **gray** refers to the $RGB + LPS_cV$ model with $LPSc_V = 0$ at test time, and **blue** indicates the $RGB + LPS_cV$ model trained with "DropFeatures" and tested with $LPScV = 0$.



(a) Ground Truth (b) RGB only (c) RGB & LPS_cV

(d) RGB & 0000 (e) RGB & LPS_cV + DropFeatures = 50% (f) RGB & {0000} + DropFeatures = 50%

Figure 3.7: Qualitative segmentation results on 3DSES - Bronze 🥉 using Swin3D. We observe in Fig. 3.7d that setting the features to zero results in many errors for the doors, heaters, walls, etc. In comparison, in Fig. 3.7f we see that DropFeatures makes the model more robust, and the segmentation is closer to the ground truth. Legend: Coverings in dark blue, doors in dark green, heaters in light green, lamps in green, ground in yellow, walls in orange, windows in light red, clutter in red.

Table 3.10: The contribution of our "DropFeatures" strategy is evaluated on LPS_cV features using segmentation metrics on the 3DSES test set. *Note that results from PointNeXt-S are the results of three runs and the standard deviations are printed in Table B.1, Table B.2 Table B.3.* **Green** indicates an improvement compared to the baseline, while **red** indicates a decrease. Mean intersection over union (mIoU), overall accuracy (OA), average accuracy (AA). "RGB 0000" indicates that features are set to 0 during inference.

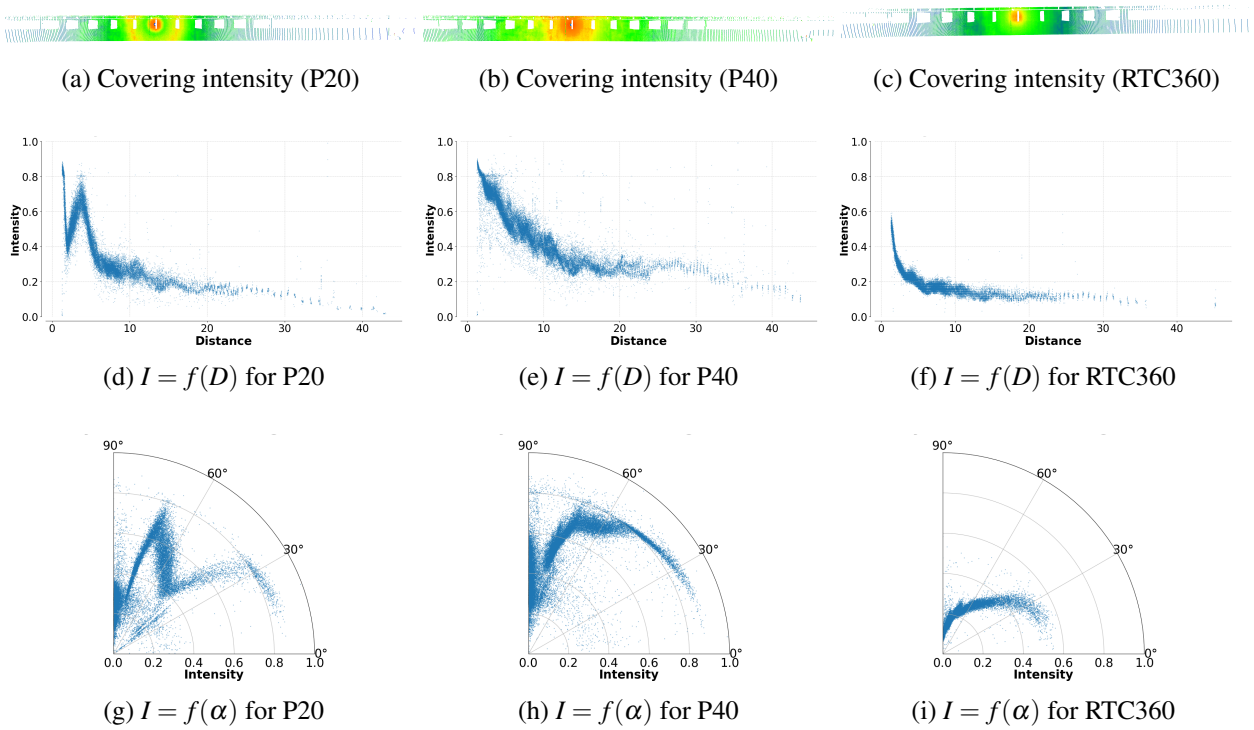
					RGB LPS_cV			RGB 0000		
		RGB	LPS_cV	DropFeatures	OA	AA	mIoU	OA	AA	mIoU
Gold	PointNext-S	✅	❌	❌	82.18	36.17	30.14	❌	❌	❌
		✅	✅	❌	**86.49**$_{+4.31}$	**37.58**$_{+1.41}$	**32.55**$_{+2.42}$	32.66	22.65	8.64
		✅	✅	50%	83.43$_{+1.25}$	36.34$_{+0.17}$	30.65$_{+0.51}$	**81.24**$_{+48.58}$	**35.09**$_{+12.44}$	**29.20**$_{+20.56}$
	Swin3D	✅	❌	❌	**89.76**	78.30	**48.99**	❌	❌	❌
		✅	✅	❌	82.41$_{-7.35}$	73.04$_{-5.26}$	48.32$_{-0.67}$	36.75	18.82	7.25
		✅	✅	50%	87.82$_{-1.94}$	**78.66**$_{+0.36}$	45.55$_{-3.44}$	**87.05**$_{+50.30}$	**77.72**$_{+58.90}$	**45.25**$_{+38.00}$
Silver	PointNext-S	✅	❌	❌	92.29	63.31	57.34	❌	❌	❌
		✅	✅	❌	**93.89**$_{+1.60}$	**65.30**$_{+2.00}$	**60.73**$_{+3.39}$	28.17	19.67	7.95
		✅	✅	50%	92.57$_{+0.28}$	63.52$_{+0.21}$	57.62$_{+0.28}$	**91.92**$_{+63.75}$	**62.68**$_{+43.01}$	**56.45**$_{+48.50}$
	Swin3D	✅	❌	❌	91.44	85.16	**59.61**	❌	❌	❌
		✅	✅	❌	91.09$_{-0.35}$	85.20$_{-0.04}$	58.86$_{-0.75}$	78.79	33.51	26.40
		✅	✅	50%	**91.74**$_{+0.3}$	**85.41**$_{+0.25}$	59.56$_{-0.05}$	**91.09**$_{+12.30}$	**84.81**$_{+51.30}$	**58.42**$_{+32.02}$
Bronze	PointNext-S	✅	❌	❌	**94.47**	67.94	62.48	❌	❌	❌
		✅	✅	❌	93.98$_{-0.49}$	71.49$_{+3.55}$	65.24$_{+2.76}$	37.83	25.05	16.33
		✅	✅	50%	94.31$_{-0.01}$	**71.92**$_{+3.98}$	**65.67**$_{+3.19}$	**94.28**$_{+56.45}$	**72.09**$_{+47.04}$	**65.77**$_{+49.44}$
	Swin3D	✅	❌	❌	94.32	**93.78**	68.19	❌	❌	❌
		✅	✅	❌	**95.01**$_{+0.69}$	91.81$_{-1.97}$	**69.54**$_{+1.35}$	76.82	53.67	43.45
		✅	✅	50%	93.90$_{-0.42}$	93.57$_{-0.21}$	67.33$_{-0.86}$	**93.66**$_{+16.84}$	**93.62**$_{+39.95}$	**66.63**$_{+23.18}$

3.5 A radiometric feature: the intensity

As seen previously (Section 2.2), the LiDAR intensity is always recorded in TLS scans. Since we aim to benefit from older surveyor acquisitions (*i.e.* XYZ coordinates and intensity), we attempt to observe if the intensity is link to object reflectance and test its contribution for segmentation. The experiments presented in this section are a preliminary test of intensity calibration. Integration of the intensity in deep models is realized in Chapter 4.

We conducted some TLS acquisitions in the ESGT (*École Supérieure d'ingénieurs Géomètres et Topographes*) metrology room (Fig. 3.8) with the objective of determining whether calibration is necessary for our 3D point cloud intensity. We acquired several scans for each TLS available at QUARTA to examine the intensity distribution of different devices from the same constructor. Specifically, Fig. 3.8 presents point clouds from the Leica Geosystems family. During this initial investigation, our objective was to identify any discrepancies in intensity distribution across different TLS devices. As shown in Fig. 3.8, the distribution of intensity can vary even if the 3D point clouds originate from the same constructor. Notably, we observed that for the P40 (Fig. 3.8b), the intensity of points near the sensor is close to 0. In contrast, for the P20 (Fig. 3.8a) and RTC360 (Fig. 3.8c), the intensity at the same distance and on the same objects is close to 1. Moreover in Fig. 3.8, we can observe a variation of intensity according to the distance and incidence angle. This suggests that intensity may not represent the reflectance of the objects. Fig. 3.10 illustrates the intensity distribution for an RTC360 and two additional elements (floor and wall) acquired in the ESGT metrology room, as a function of the distance and incidence angle. On the covering, we can see a $1/D^2$ variation for the distance (Fig. 3.8f) and a $\cos(\alpha)$ variation for the incidence angle (Fig. 3.8). We retrieve this trend for the wall (Fig. 3.9a). However the floor intensity seem to be corrected (Fig. 3.9c).

With our restricted protocol, it seems unclear whether intensity varies solely as a function of object reflectance or if it is also affected by distance and incidence angle. In Fig. 3.9, we decided to correct the intensity using a model-driven approach (cf. Eq. (2.3)). To test some corrections, we used the point clouds acquired by the RTC360 and used the distance to the scanner (Fig. 3.10b) and the angle of incidence (Fig. 3.10c) to correct the constructor's intensity (Fig. 3.10a). Fig. 3.10d corresponds to a 3D point cloud with an intensity corrected by previous parameters. We observe that the corrected intensity does not help in distinguishing objects. As it is challenging to calibrate the intensity (*i.e.* without a measure on Spectralon), for our following experiments we will use the intensity from the constructor.

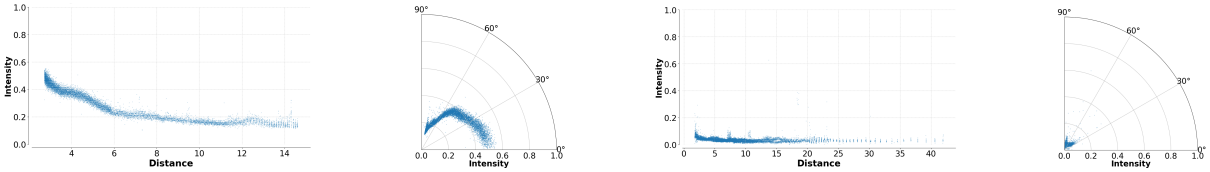


(a) Covering intensity (P20) (b) Covering intensity (P40) (c) Covering intensity (RTC360)

(d) $I = f(D)$ for P20 (e) $I = f(D)$ for P40 (f) $I = f(D)$ for RTC360

(g) $I = f(\alpha)$ for P20 (h) $I = f(\alpha)$ for P40 (i) $I = f(\alpha)$ for RTC360

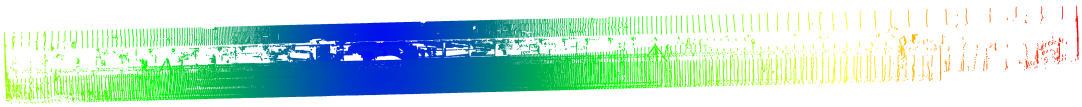
Figure 3.8: Experiment on the "ESGT metrology room" to acquire the same room using three different TLS (P20, P40, RTC360). Figs. 3.8a to 3.8c presents the ceiling of "ESGT metrology room" acquired with three distinct TLS from the same manufacturer. This figure shows that the constructor's intensity is not balanced within the same intensity interval. Intensity is balanced between **zeros** and **one**. We also plot the variation of intensity according to the sensor distance (D) and incidence angle (α).



(a) $I = f(D)$ (b) $I = f(\alpha)$ (c) $I = f(D)$ (d) $I = f(\alpha)$

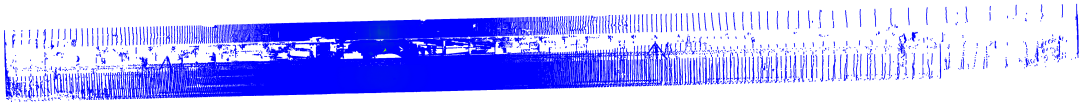
Figure 3.9: Plots of intensity as a function of the distance to the sensor and incidence angle. These plots, correlate with Fig. 3.8 exhibit dissimilar trends: while the intensity for wall appears to be influenced by distance and incidence angle, the floor intensity does not seem to vary. This plot suggests that the TLS manufacturer has already applied some preprocessing to the intensity.

(a) Constructor intensity for the RTC360



(b) Visualization of the D^2

(c) Visualization of the incidence angle



(d) Visualization of the calibrated intensity

Figure 3.10: Attempts at intensity correction (Fig. 3.10a). We do not design a calibration process and try to calibrate the intensity with the Eq. (2.3) (*i.e.* with distance and incidence angle).

3.6 Conclusion

In this chapter, we evaluated the contribution of geometric features in different deep models for semantic segmentation of 3D point clouds. We showed that even in the age of 3D deep learning, handcrafted features can still be helpful and lead to performance improvements, especially for small models and small datasets. However, the contribution of handcrafted features on small objects is ambiguous and further experiments need to be carried out. These promising results is an encouraging way to benefit from older surveyor's point clouds which only contain XYZ coordinates and intensity. Moreover, the results presented in this chapter hint that features can be more helpful for small architectures. If confirmed, this could be an alternative to large models for surveyor companies. Since geometric features can be expensive to compute, and are sometimes unavailable at test time, we introduced DropFeatures to regularize deep models and make them robust to missing features by randomly resetting them to zero during training. We showed that this strategy effectively improves model robustness in the missing features scenario (even for small objects), and that it can even improve model performance through regularization. In S3DIS, our strategy helps maintain performance and avoids a drop of +35.58%mIoU for Swin3D and +69.65%mIoU for PointNeXt-XL. In practice, this allows us to obtain models that outperform the baselines when tested with features, while still maintaining on-par segmentation performance when used without the features. Regarding another features, the intensity, we are experiencing difficulties in calibrating it. In the following chapter, we introduce a new dataset with point clouds that includes TLS intensity. We hope this dataset will help the community to calibrate indoor LiDAR intensity and use them in recent deep models.

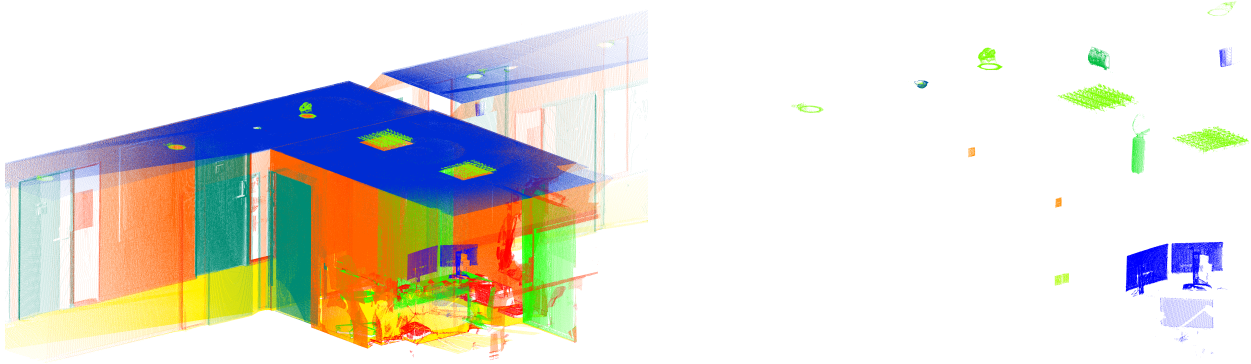
4

A realistic indoor point cloud segmentation dataset

Content

Abstract

As outlined in Chapter 2, annotating a 3D point clouds is time consuming, especially when focusing on small and complex objects. In the preceding chapter, we proposed to rely on handcrafted features to improve segmentation accuracy. However, handcrafted features do not solve the lack of small objects in available datasets. In this chapter, we propose a method to benefit from "frozen" data of land-surveyor companies. We present *3DSES (3D Segmentation of ESGT point clouds)*, a new dataset of indoor dense TLS colorized point clouds covering 427 m^2 of an engineering school. 3DSES has a unique double annotation format: semantic labels annotated at the point level and a full 3D CAD model of the building. We introduce a model-to-cloud alignment algorithm to automate the labeling of indoor point clouds using an existing 3D CAD model. This alignment allows us to train deep models with significant time savings compared to manual labeling. First baselines on 3DSES show the difficulties encountered by existing deep models when segmenting small objects (*e.g.* lights and safety utilities). All data are available on Zenodo.



(a) Point clouds colorized by classes

(b) Small objects from the previous point clouds

Figure 4.1: Examples of small objects of 3DSES [1]. In Fig. 4.1a, the point clouds are segmented in several classes. While Fig. 4.1b illustrates several small objects included in 3DSES such as exit sign, monitor, switch.

4.1 From surveys to dataset

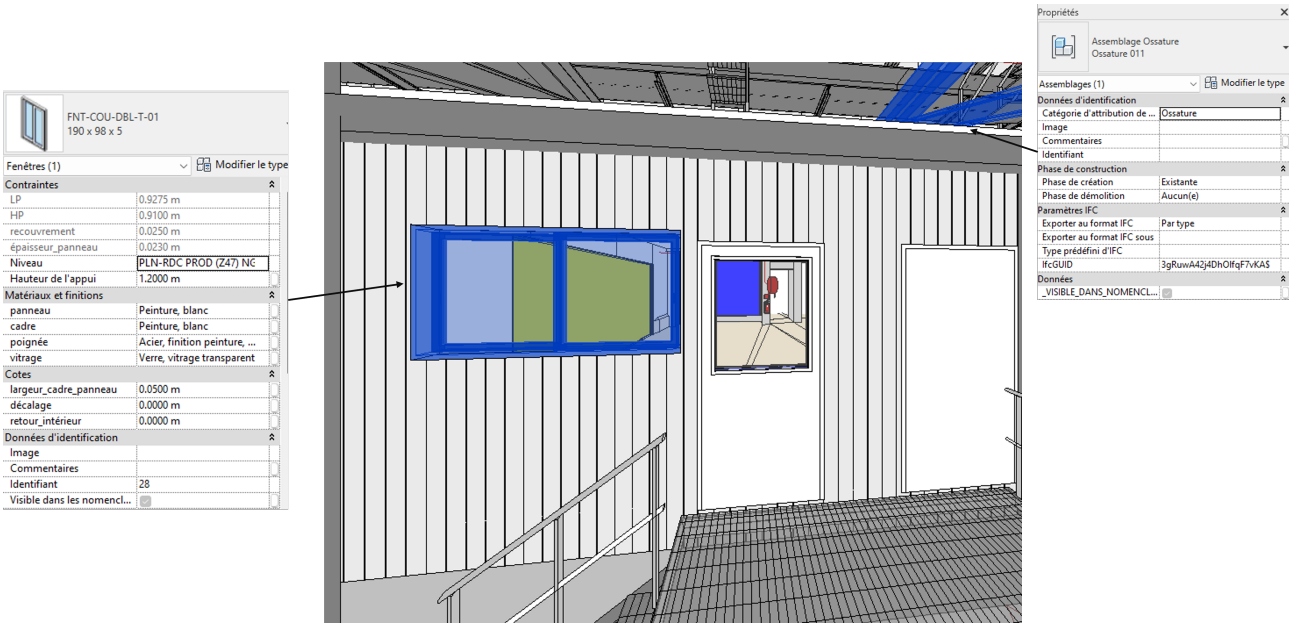
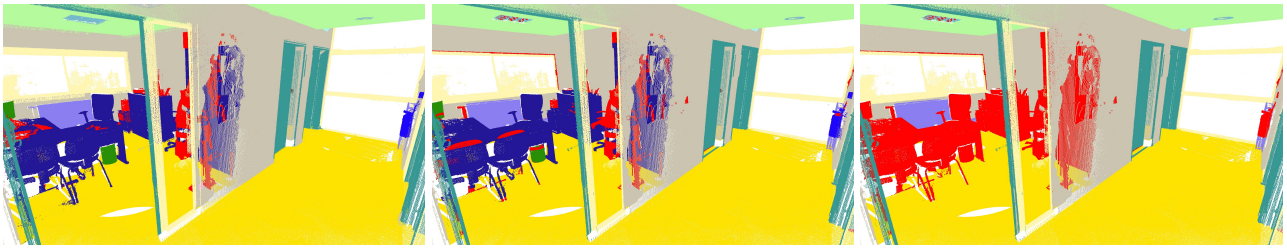


Figure 4.2: A 3D CAD model is an approximate representation of reality. In most surveyor companies, it is based on point cloud scans. Objects within a CAD model can be tagged with additional information (such as material, year of maintenance, . . .).

As introduced in Section 1.3, QUARTA is a land surveyor company that carries out a lot of point cloud acquisitions (see Fig. 4.1). However the main finality is not the point clouds, but rather a 2D plan or a Building Information Model (BIM). In this chapter, we focus on the benefits of BIM for the enrichment of point clouds. BIM is a comprehensive tool for managing buildings throughout their entire life cycle, from construction to demolition. It consists in creating a digital representation of a building, called a “digital twin” (Fig. 4.2). 3D models help reduce construction and maintenance costs by facilitating planning and simulation on the virtual assets [195] and preserve heritage structures [8]. BIM allows for monitoring buildings over time and managing equipment by recording details such as installation date and maintenance schedules. The creation of digital twins involves *in situ* acquisitions to reconstruct the building’s 3D structure, often using point clouds [6, 5, 196]. In recent years, 3D data acquisition technologies have not only significantly improved in accuracy, but also diversified their sensing apparatus. In most cases, sensors create point clouds based either on photogrammetry, *e.g.* using stereo photography or structure-from-motion, or on laser-based Lidar systems. Acquisition has been made increasingly intuitive and easy with the improvements of 3D scanners, including real-time positioning and very high acquisition speed. Terrestrial Laser Scanning (TLS) has become the standard for surveyors to create

(a) RGB (b) 3D model (c) Intensity



(d) Gold - Real labels (e) Gold - Pseudo labels (f) Silver - Pseudo labels

Figure 4.3: Modalities and annotation variants of 3DSES. Gold real labels are manual annotations across 18 classes, including small objects such as light switches and electrical outlets. Pseudo-labels are obtained by automatically aligning the 3D model on the point cloud, introducing some noise in the annotation (see *e.g.* the top of the chairs). Silver labels use a simplified classification of only 12 categories (*e.g.* the wastebin is now simply "clutter"). Legend: Column in dark purple, components in dark green, coverings in light green, doors in green, emergency signs in light blue, fire terminals in dark blue, heaters in light purple, lamps in blue, ground in yellow, walls in grey, windows in light yellow, clutter in red.

large point clouds of building interiors in a few hours.

Meanwhile, the enrichment of point clouds has not met the same progresses. 3D CAD modeling of buildings based on point clouds remains a manual and time-consuming task. Creation of 3D CAD models is minimally automated and still requires the intervention of qualified experts. Semantic segmentation of point clouds is a promising avenue to automatically label point clouds, and could accelerate the modeling by helping surveyors to identify structural primitives (walls, ground, doors) and even furniture types (chairs, tables, etc.). However, as introduced in Chapter 2, few datasets exist for semantic segmentation of indoor TLS point clouds.

Moreover, surveying companies (such as QUARTA) have access to large databases of existing 3D CAD models and associated point clouds, but the latter are mostly unlabeled. Indeed, in Chapter 2, we saw that annotating a 3D point cloud is time-consuming, and in the case of surveying companies, a lot of time has already been dedicated to the creation of the 3D model. For these reasons, we introduce 3DSES (Fig. 4.3), a dataset of indoor TLS acquisitions with manually annotated point clouds and a BIM-like 3D CAD model. In addition

Table 4.1: Comparison of the characteristics of various point cloud datasets from the literature. Note that 3DSES is the only indoor TLS dataset that includes intensity, point level annotations and a 3D CAD model. Despite its size, it also has more points than most existing datasets, demonstrating a very high point density.

Name	Environment	Classes	Extent[1]	Points (M)	Intensity	3D model	Source
Oakland [197]	Outdoor	44	-	1.6	❌	❌	MLS
Paris-rue-Madame [198]	Outdoor	17	160 m	20	✅	❌	MLS
IQmulus [28]	Outdoor	8	10 000 m	12	✅	❌	MLS
Semantic 3D [199]	Outdoor	8	-	4000	✅	❌	TLS
Paris-Lille-3D [37]	Outdoor	9	1940 m	143.1	✅	❌	MLS
SemanticKITTI [38]	Outdoor	25	39 200 m	4500	✅	❌	MLS
Toronto-3D [200]	Outdoor	8	1000 m	78.3	✅	❌	TLS
S3DIS [120]	Indoor	13	6020 m^2	273	❌	❌	Camera
Matterport3D [131]	Indoor	20	219 399 m^2	-	❌	❌	Camera
ScanNet [121]	Indoor	20	78 595 m^2	242	❌	❌	Camera
ScanNet200 [132]	Indoor	200	78 595 m^2	242	❌	❌	Camera
ScanNet++ [39]	Indoor	1000+	15 000 m^2	446	❌	❌	TLS
LiDAR-Net [133]	Indoor	24	30 000 m^2	3619	❌	❌	MLS
3DSES Gold 🥇	Indoor	18	101 m^2	65	✅	✅	TLS
3DSES Silver 🥈	Indoor	12	304 m^2	216	✅	✅	TLS
3DSES Bronze 🥉	Indoor	12	427 m^2	413	✅	✅	TLS
Indoor Modelling [201]	Indoor	❌	2824 m^2	127	❌	✅	5 sensor
Craslab [202]	Indoor	❌	417 m^2	584	✅	✅	TLS

[1] Surface for indoor datasets, linear extent for outdoor datasets.

to the overall structure and furniture, we label several types of common BIM elements, such as extinguishers, alarms and lights, that are challenging to detect in point clouds. To evaluate the feasibility of automatically annotating point clouds based on existing BIM models, we introduce in Section 4.4.1 a 3D model-to-cloud alignment algorithm to label point clouds. In Section 4.4.2, we show that these pseudo-labels are nearly as effective as manual point cloud annotation for most classes. However, in Section 4.6 we show that small objects remain extremely challenging for existing point cloud segmentation models.

4.2 Previous work

Numerous datasets exist for semantic segmentation of point clouds with several sizes of scenes, different types of objects of interest and acquired using various sensors, each with their own characteristics. Previously,

in Section 2.5, we described some indoor datasets. Here, we review in Table 4.1 some point cloud segmentation datasets available and their modalities at the time of 3DSES's creation.

4.2.1 Outdoor datasets

The first popular datasets for semantic segmentation of point clouds focused on outdoors. Mobile laser scanning is popular for outdoor scenes as moving platforms cover more ground. Since the laser is moving, the point clouds tend to be sparse, *e.g.* the seminal Oakland dataset [197] has less than 2M points. Later datasets such as IQmulus [28] or Paris-rue Madame [198] are also relatively small, with less than 20M points. Bigger datasets have been consolidated by covering larger scenes, such as Paris-Lille-3D [37] and SemanticKITTI [38]. While MLS makes sense for autonomous driving, segmentation performance on these point clouds is not representative of indoor scenes which are much denser with lots of small objects. Concurrently, point clouds acquired by aerial Lidar have been used to create datasets on very large scenes, such as the ISPRS 3D Vaihingen [34], DublinCity [35], LASDU [203], DALES [204], Campus3D [205], Hessigheim [206], SensatUrban [207] and FRACTAL [36]. These datasets use Aerial Laser Scanning (ALS), with a top-down view that makes them effective for digital surface models but unsuitable for BIM.

However, some outdoor datasets have a density and geometry close to those found in indoor cases. For example, Semantic 3D [199] and Toronto-3D [200] both use TLS with high point density. These outdoor scenes do not contain many small objects, though, as they rarely consider classes smaller than outdoor furniture, *e.g.* benches or trashbins.

4.2.2 Indoor datasets

Few new indoors datasets have been published in the last five years. The two most widely used datasets S3DIS [120] and ScanNet [121] – were published in 2017. The lesser known Matterport3D [131] was published in the same year with similar characteristics. ScanNet was updated with more classes in ScanNet200 [132], yet using the same point clouds. All these datasets are acquired by RGB-D cameras. The resulting point clouds are sparser and more sensitive to occlusions than TLS data. For example, S3DIS contains 273 million points, which corresponds to approximately ten stations in a medium-resolution TLS system. Yet, these datasets are the most common benchmarks to evaluate deep point cloud segmentation, meaning that new approaches are tested on partially obsolete technology. While indoor TLS datasets exist, *e.g.* Indoor Modeling [201] and Craslab [202], they do not contain semantic labels and only release a simplified CAD model. LiDAR-Net [133] uses a mobile

laser scanner (MLS) to create an indoor dataset more suitable for autonomous navigation, resulting in a point cloud that contains scan holes, scan lines and various anomalies that are not shared with TLS scans for building surveys. To the best of our knowledge, the only dataset using labeled TLS point clouds is ScanNet++ [39]. However, ScanNet++ used a complex three devices acquisition setup. DSLR images were acquired separately from the scans, and then backprojected to colorize point clouds. This setup is not representative of usual surveys practices. For 3DSES, we use a simpler acquisition workflow, as the RGB information comes directly from the TLS.

Point clouds with intensity As explained in Section 2.2, LiDAR intensity measures the strength of the laser impulse returned by a scanned point. It is a feature commonly used in outdoor point cloud datasets, especially because infrared is helpful to identify vegetation. However, intensity is notably absent from indoor datasets. Yet, we presented in Chapter 2 that intensity is often recorded with LiDAR apparatus. In theory, different materials reflect light differently and these variations impact the measured intensity of the laser echo. This information might help deep models to discriminate between objects that have similar geometry, but different natures. For this reason, we include the intensity information in our 3DSES dataset.

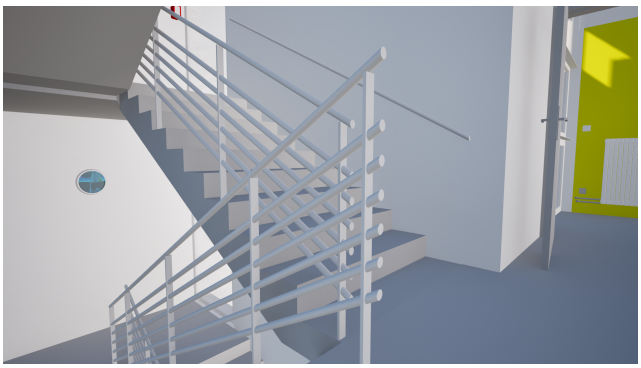
Uniqueness of 3DSES While covering a smaller surface than other datasets, 3DSES is extremely dense, with a sub-centimeter resolution. It is also the only TLS dataset with Lidar intensity, an information often removed in publicly available datasets, despite theoretically being a discriminative property of materials. 3DSES is also a *labeled* dataset, suitable to train or evaluate semantic segmentation algorithms. Finally, 3DSES comes with a 3D CAD model designed for BIM. This combination is unique across existing datasets, and makes 3DSES suitable to investigate 3D point clouds for indoor building surveys and modeling.

4.3 A data collection close to the land surveyor process

We present in this section the data acquisition, the 3D modeling and then the labeling process. To demonstrate the benefits of our method for land surveyor companies, we need to follow their data acquisition practices. Similar to the land surveyor process, we first need to perform 3D point clouds acquisition, followed by 3D modeling. Additionally, to compare our results to a ground truth, we must conduct manual point cloud annotations.

Point cloud acquisition Data acquisition was carried out at ESGT (*École Supérieure d'ingénieurs Géomètres et Topographes*) using two Terrestrial Laser Scanners (TLS): a Leica RTC360 and a Trimble X7. High-resolution

(a) Example of modeled 3D systems: fire alarm, fire extinguisher, heater, outlet, light switch.



(b) Structural objects: stairs, railings, doors, walls, floors.

(c) 3D point cloud of a room

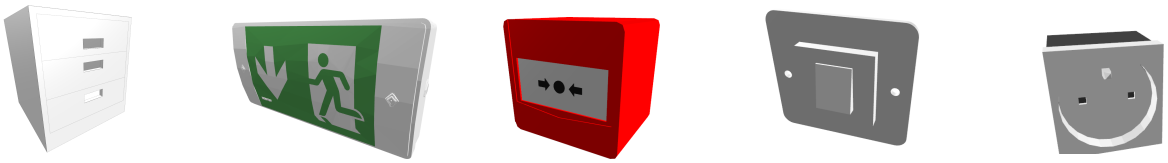
(d) 3D model of a room

(e) Overlay of clouds and objects

Figure 4.4: View of a test area room. For 3D modeling, we follow the practices of land surveyor companies. The generic 3D models are close, but not perfect matches for the actual scans.

pictures were taken for each scan (15MP for RTC360 and 10MP for Trimble X7). Scans were preregistered during the survey, using software from each corresponding manufacturer. We performed and bundled multiple scans inside every room to capture as many objects as possible. Scans were then merged for registration, and any missing link was manually corrected. Point clouds are georeferenced using coordinates from total stations and GNSS. We release both colorized Fig. 4.3a) and intensity Fig. 4.3c) clouds. Note that 3DSES concerns the first floor of the ESGT building and contains only scans from RTC360.

3D CAD model In surveyor companies, a 3D CAD model may be built using several proprietary softwares, such as Autodesk Revit [208]. However, to facilitate exchanges the final 3D model is standardized to Industry Foundation Classes (IFC). Each type of object is tagged as a member of the corresponding IFC family. The geometry of structural elements (walls, floors, roofs, etc.) is accurately modeled, *i.e.* shapes and dimensions are modeled as precisely as possible. Furniture, such as tables and chairs, and utilities, such as fire extinguishers and emergency exit signs, use standard models, *e.g.* all chairs use the same mesh (see Fig. 4.4d). This is a common practice in BIM, as defining a separate "chair" family for each instance would be too time-consuming. Fig. 4.4e



(a) Cabinet file. (b) Exit sign (c) Manual trigger. (d) Switch (e) Outlet

Figure 4.5: Examples of CAD objects used in 3DSES. We use generic shapes for each element, and model some small objects present in indoor scenes.

illustrates how these generic 3D CAD models create slight geometrical discrepancies between the point cloud and the model. Fig. 4.5 illustrates other generic uses during the 3D modeling. Finally, a special care is given to doors, that can appear either open or closed in scans. We model each door in its correct state depending on its true position in the point cloud. Complete modeling took slightly less than 30 hours in Autodesk Revit.

Manual labeling In order to prove the interest of our method, we need a ground truth. Therefore we manually annotated the point clouds to create a ground truth denoted as the *real labels*, shown in Fig. 4.3d. A taxonomy of 3DSES variants is presented in more detail in Appendix C. As introduced in Chapter 2, this labeling process is time-consuming. Therefore, we annotated only 10 point clouds in 18 fine-grained classes: “Column”, “Component”, “Covering”, “Damper”, “Door”, “Exit sign”, “Fire terminal”, “Furniture”, “Heater”, “Lamp”, “Outlet”, “Railing”, “Slab”, “Stair”, “Switch”, “Wall”, “Window” and a “Clutter” class that encompasses all points not belonging to another class. Labels were annotated in two passes: 1) labeling by a single annotator (30 to 40 minutes per scan, depending on the complexity of the point cloud, the number of points and the diversity of represented objects); 2) verification pass by an experienced annotator (20 to 30 minutes per scan). We then annotated 20 additional point clouds with a simpler taxonomy of only 12 classes, shown in Fig. 4.3f. These labels were annotated in a single pass, as the target objects are less ambiguous with simpler geometries. During this process, the point clouds were partially cleaned of outliers and far away points.

4.4 An alignment method

Our method is detailed in a pseudo-code (Algorithm 1) and visually explained in (Fig. 4.6). Additional details concerning computation time and confusion matrix are detailed in Appendix C.

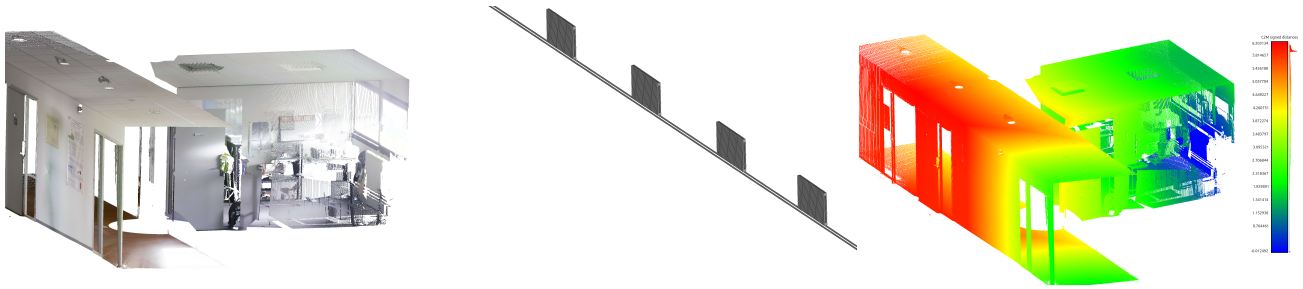
4.4.1 Pseudo-labeling from the 3D model

One of our goals is to evaluate the feasibility of using existing 3D CAD models to label automatically point clouds for semantic segmentation. Pseudo-labels (*i.e.* labels from CAD model) could help leverage existing databases of surveyed buildings that have been scanned and modeled, but not annotated at the point level. To this end, we design an alignment algorithm to map the 3D model on a point cloud (Algorithm 1).

Algorithm 1: Pseudo-labeling of the point cloud using model-to-cloud alignment.

Data: point cloud $\leftarrow [p_1, p_2, \ldots, p_n]$; /* list of (x,y,z) points */
Data: model $\leftarrow [\text{mesh}_1, \text{mesh}_2, \ldots, \text{mesh}_k]$; /* list of 3D objects */
Data: $\tau \geq 0$; /* threshold */
classes $\leftarrow$ initialize list of size n with "Clutter";
for *p in point cloud* **do**
  $d_{\text{min}} = +\infty$;
  for *mesh in model* **do**
    $d \leftarrow$ compute signed distance between $p = (x, y, z)$ and mesh;
    if $d \leq 0$; /* point inside the object */
    then
      classes[p] $\leftarrow$ class(mesh);
    else if $d \leq \tau$ ***and*** $d < d_{min}$; /* p outside but near the closest object */
    then
      classes[p] $\leftarrow$ class(mesh);
      $d_{\text{min}} = d$;
  end
end
Result: classes ; /* classified point cloud */

First, we divide our 3D CAD model into objects. This allows us to separate individual instances of walls, heaters (Fig. 4.6b), light switches and so on. For each object, we produce the corresponding 3D mesh. Since the 3D CAD model and the point cloud are georeferenced, we can compute a mesh-to-cloud distance for every point in the point cloud. For each CAD object, we first compute its georeferenced bounding box. Then, we compute the distance for each point inside the bounding box to the mesh of the object using the Metro algorithm [209], implemented in CloudCompare [4]. All points that are inside the mesh are labeled the same class as the IFC family of the object the mesh is derived from. To alleviate for geometrical discrepancies between the mesh and the point cloud, points outside the mesh are assigned to their closest mesh as long as the distance is lower than a predefined threshold. We then repeat this process for all objects. Remaining points that have not been labeled are classified as "clutter". This covers objects that are present in the scan, but have not been modeled, *e.g.* jackets



(a) Point cloud of 3DSES (b) CAD of some heaters (c) Point cloud colorized by distance

Figure 4.6: Overview of our process for a point cloud of 3DSES. Our model-to-cloud alignment method treats each scan separately (Fig. 4.6a), and the algorithm (Algorithm 1) utilizes a CAD model Fig. 4.6b to assign a class label to each point (Fig. 4.6c).

on chairs, books and papers on tables, etc.. For instance, Fig. 4.6c illustrates the resulting from the distance between "Heaters" and the XYZ coordinates of a 3DSES's point cloud. The algorithm runs in around 9 hours on CPU to align the full dataset (Bronze). This means the pseudo-labeling process (3D model + alignment) takes $\approx$ 40 hours. In comparison, manual point cloud annotation takes 1 hour per scan on average, *i.e.* would have taken 42 hours for 3DSES Bronze, including quality check. While these times are comparable, point clouds are intermediate products in indoor surveys, the end goal of which is almost always the production of a 3D CAD model. This is why we assess whether pointwise labels can be obtained as a "free" byproduct, without any additional time dedicated to point annotation.

4.4.2 Evaluation of the pseudo-labels

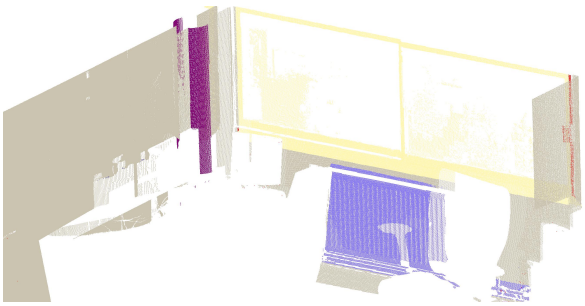
Since 3DSES also includes real labels, we can evaluate how well the pseudo-labels match the ground truth. To do so, we computed some standard segmentation metrics, *i.e.* Intersection over Union (IoU), mean Accuracy (mAcc) and Overall Accuracy (OA) (see Appendix A). We used different confidence thresholds depending on the object class:

- Gold: 4 cm for all classes, except for "Door", "Furniture", "Window", for which we used 10 cm, due to larger uncertainties when modeling;
- Silver and Bronze: 4 cm for all classes, except for "Door" (10 cm) and "Window" (15 cm).

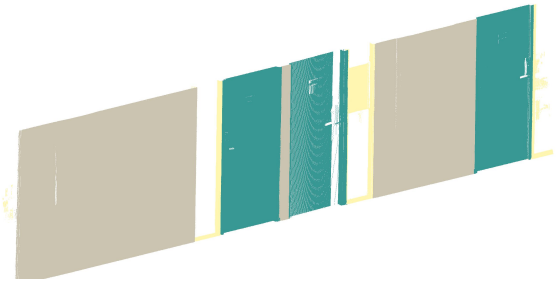
Metrics are computed between pseudo-labels and the manual ground truth over the full dataset. We report the alignment metrics in Table 4.2. We obtain high-quality pseudo-labels on Gold version with $\approx$ 70% mIoU

Table 4.2: Evaluation of the accuracy of the pseudo-labels obtained using our alignment algorithm on 3DSES. Intersection over Union (IoU) per class, mean IoU (mIoU), overall accuracy (OA) and average accuracy (AA).

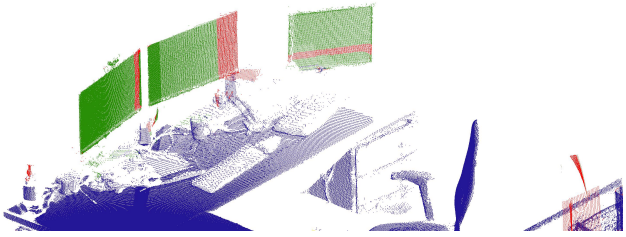
Variant	Column	Components Covering	Damper	Door	Exit sign	Fire terminal	Furniture	Heater	Lamp	Outlet	Railing	Slab	Stair	Switch	Wall	Window	Clutter	**OA**	**AA**	**mIoU**
Gold	21.00	80.96	95.95	77.29	91.95	73.16	86.57	79.48	91.08	66.71	37.59	58.52	95.05	59.07	45.66	93.64	64.55	36.44	94.66	83.09	69.70
Silver	25.02	✗	97.99	✗	93.97	72.27	✗	✗	82.22	73.88	✗	58.52	96.20	59.07	✗	91.52	56.67	88.88	96.37	83.40	74.68



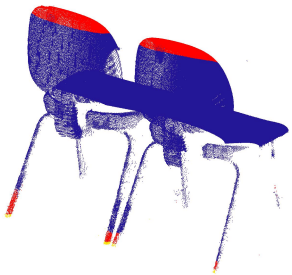
(a) A room with “Column”, “Heater”, “Window”, “Wall” well segmented by Algorithm 1.



(b) A corridor with “Door”, ‘Wall”, “Window” well segmented by Algorithm 1.



(c) Three screens. The red color (*i.e.*,“Clutter”) indicates a segmentation error in our alignment methods.



(d) Two chairs. The red color (*i.e.*,“Clutter”) indicates a segmentation error in our alignment methods.

Figure 4.7: Presentation of pseudo-labels for several point clouds. Our method performs well on structural elements. However, smaller elements, which are usually less well modeled, are susceptible to more mis-annotations.

Figure 4.8: Top view of 3DSES and the split of points across the Gold, Silver, and Bronze variants. Note that these variants are inclusive, *i.e.* Silver includes Gold and Bronze includes Silver.

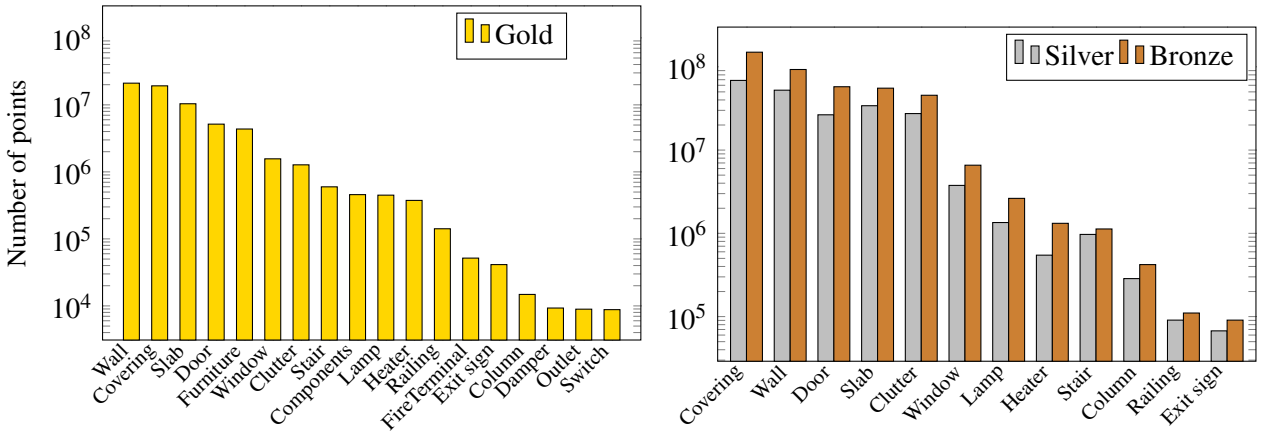
and 95% accuracy. Structural classes ("Covering", "Slab", "Wall") are very well annotated, with a >90 % score. This is expected as these entities have regular shapes with a fine alignment between the 3D model and the point cloud. The lowest scores are on the "Outlet" and "Switch" classes, below 50 % mIoU. Alignment on the Silver variant is also satisfactory with $\approx$ 75% mIoU and $>$ 96% accuracy. Metrics are higher on Silver since it focuses on structural classes that are generally easier to align. The IoU for "Column" is also the lowest due to the use of a slightly too small column diameter in the CAD model. The second worst score is for "Window" with 69 %, as Silver contains more window types, including frames that deviate from the CAD model. Finally, metrics on "Railing" and "Stair" are identical on Gold and Silver, since stairs cover the same area in both datasets. Fig. 4.7 shows pseudo-labels resulting from our methods for several elements of 3DSES. Fig. 4.7a and Fig. 4.7b reveal that our method performs well on structuring elements. However, using generic 3D shapes in our method generates some errors. More precisely, 3D point clouds objects may be labeled as "Clutter" for objects model using generic shapes. Fig. 4.7c highlights difficulties in correctly labeling screens using generic shapes, due to their size and orientation. Furthermore, using generic chair models induces mislabeling of some chair (Fig. 4.7d).

4.5 3DSES: Dataset variants

Based on the TLS scans and the manual annotations, we built three versions of the 3DSES dataset (cf. Table 4.3). The Gold version is composed of the 10 scans annotated in 18 classes. We consider it to be the "gold standard", using fine-grained high quality real labels. We then extended it into a Silver version that contains all the Gold data and an additional 20 scans. Silver labels use a simplified taxonomy of only 12 classes, that are less

Table 4.3: Characteristics of the three variants of the 3DSES dataset.

Variant	Scans	Points	Ground truth	Pseudo-labels	Features	Classes
Gold 🥇	10	65 214 193	✅	✅	RGB & I	18
Silver 🥈	30	216 181 580	✅	✅	RGB & I	12
Bronze 🥉	42	413 486 927	❌	✅	RGB & I	12



(a) Real labels distribution (Gold 🥇).

(b) Pseudo-labels (Silver 🥈 /Bronze 🥉).

Figure 4.9: Distribution of the real and pseudo labels in the variants of the 3DSES dataset.

time consuming to produce. Both Gold and Silver variants of 3DSES are high quality, using a real ground truth and cleaned up point clouds. Finally, we deliver a Bronze version that includes 12 more scans. Bronze contains the raw point clouds and not the processed and cleaned clouds. These full point clouds are denser and noisier, but also more representative of actual field scans. Since the additional point clouds have not been manually labeled, the Bronze dataset uses the automatically generated pseudo-labels based on the 3D model using the procedure detailed in Section 4.4.1.

Note that all variants suffer from class imbalance, as shown in Figs. 4.9a and 4.9b. Structural elements are over represented compared to other classes, especially furniture and utilities, that are comprised of smaller objects. This is a well-known issue in indoor datasets, such as S3DIS [120], which has 10× more wall points than window points, and ScanNet200 [132], which contains 51 million wall points and only 50 000 fire extinguisher points.

Train/test split We define a set train/val/test split with a common test area to all variants, based on three scans located in the Gold section (scans S170, S171 and S180). It contains ≈ 20.7 million points with real

ground truth. This allows us to evaluate models on real labels only, whether they have been trained on real or pseudo-labels. Ground truth labels on the test set are kept hidden for an use in a Codabench challenge. However, as detailed in Appendix C, coordinates, color and intensity for this three point clouds are available on Zenodo.

4.6 Limits of deep models for small objects segmentation

To assess the difficulty of 3DSES, we evaluate initial baselines for the three variants: Gold, Silver and Bronze. We opt for PointNeXt [100] and Swin3D [186], since they are some of the highest performing models for semantic segmentation on S3DIS [120], and their code is available. We compare PointNeXt-S (800 000 parameters) to Swin3D-L (68M parameters).

Note that these models both perform voxelization (see Section 3.3 for more details) and therefore do not benefit from the extremely high point density of 3DSES. In particular, PointNeXt is not designed to process dense point clouds in optimum time ($\approx$ 4 hours per scan). To reduce inference times, we subsample our test point clouds to 1 cm. We expect that future models evaluated on 3DSES will better take into account the fine resolution of indoor TLS scans.

Hyperparameters We train Swin3D-L with AdamW, a cosine learning rate for 100 epochs, a batch size of 6, and an inverse class frequency weighted cross-entropy to deal with class imbalance. PointNext-S is trained with the original S3DIS hyperparameters: epochs $= 100$, batch size $= 32$, AdamW optimizer, a CosineScheduler and a non-reweighted CrossEntropyLoss. We only tune the learning rate to $l_r = 0.05$ (instead of 0.01 in original setup). Following standard practices [210, 211, 186], we use test-time augmentation and aggregate segmentation predictions with a majority vote over 12 rotations. Models are trained on an NVIDIA RTX A6000.

4.6.1 Baseline for 3DSES

Results on 3DSES Gold 🥇 We train both Swin3D-L and PointNeXt-S models on 3DSES Gold: one on the real labels and the other on the pseudo-labels. All models are evaluated on the ground truth over the test area. Results are reported in Table 4.4. We observe that 3DSES is a challenging dataset: mean IoU is heavily penalized by low performance on small objects. Classes comprised of small objects with few points ($< 10^5$ points) are difficult to learn and the model either never predicts them, or makes significant errors. Note that despite its high intraclass variance, "Clutter" is mostly well segmented with a $> 50\%$ IoU, showing that the model is able to automatically identify most irrelevant objects from the point clouds. Interestingly, the results

Table 4.4: Segmentation metrics on the test set for 3DSES Gold, either with real or pseudo labels (and intensity features or not). Intersection over union (IoU) per class, mean IoU (mIoU), overall accuracy (OA), average accuracy (AA).

	Real labels	Intensity	Column	Components	Covering	Damper	Door	Exit sign	Fire terminal	Furniture	Heater	Lamp	Outlet	Railing	Slab	Stair	Switch	Wall	Window	Clutter	**OA**	**AA**	**mIoU**
Swin3D	✅	❌	0.00	31.16	90.12	14.63	75.95	12.19	56.67	71.57	76.18	26.76	9.53	71.75	87.63	70.59	0.00	88.40	47.26	52.03	89.74	78.30	49.02
	✅	✅	0.00	49.76	94.62	18.23	81.87	27.37	67.10	73.13	83.61	47.73	0.00	57.31	85.29	56.67	0.00	89.68	53.54	50.46	91.64	74.45	52.02
	❌	❌	17.52	34.81	88.90	31.71	75.84	16.31	48.28	68.87	71.04	24.50	12.85	45.53	86.84	58.64	0.93	87.09	50.59	40.31	88.54	76.80	47.81
	❌	✅	30.06	51.07	93.29	63.98	54.16	0.00	21.36	51.32	66.14	41.09	6.33	50.31	79.04	40.46	0.00	83.92	48.96	31.98	86.48	74.10	45.19
PointNeXt-S	✅	❌	0.00	0.00	96.27	0.00	35.43	0.00	0.00	32.84	0.00	69.12	0.00	0.00	90.87	60.40	0.00	74.58	38.05	24.80	82.58	35.04	29.02
	✅	✅	0.00	56.16	96.73	0.00	65.80	0.00	0.00	52.57	26.59	72.78	0.00	60.75	94.28	85.93	0.00	86.76	59.78	39.47	91.19	49.25	44.31
	❌	❌	0.00	0.01	96.01	0.00	37.57	0.00	0.00	45.11	0.00	39.76	0.00	0.00	89.73	60.33	0.00	77.57	1.18	20.33	84.19	30.48	25.98
	❌	✅	0.00	50.10	96.68	0.00	67.86	0.00	0.00	49.83	43.32	65.51	0.00	7.51	93.79	81.23	0.00	86.27	55.81	21.35	90.08	44.86	39.96

also show that Swin3D only slightly underperforms when trained on the pseudo-labels, with a 1.2% decrease in mIoU (47.8% vs. 49.0%) compared to the model trained on the real labels. Segmentation errors when using pseudo-labels are concentrated on classes for which the alignment procedure showed weaknesses, such as "Stair" and "Railing". This demonstrates the potential of using CAD models to automatically label point clouds, as way of circumventing the lack of annotated datasets for specialized settings (*i.e.* factories, schools or administrative buildings...). PointNext struggles with 3DSES and achieves low mIoU scores. However, the same trends hold with better segmentation of structural elements and underperformance on minority classes.

Results on Silver 🥈 / Bronze 🥉 We report in Table 4.5 the segmentation metrics on the 3DSES test set when training Swin3D and PointNeXt-S on Silver, both with pseudo and real labels, and on Bronze with pseudo labels. We observe that metrics are consistently higher for all 12 classes on Silver with real label compared to training the Gold subset. This is expected, since the Silver classification is simpler and removes small objects that were heavily penalized. Yet, the larger training set (Silver is 3× as large as Gold) benefits the segmentation, with higher scores on the "Lamp", "Window" and "Clutter" classes that exhibit strong diversity. Training with pseudo-labels on Silver results in a significant performance drop, correlated with the lower class alignment scores discussed in Section 4.4.2. However results on 3DSES Bronze show that the noise in the pseudo-labels can be alleviated by a larger dataset. Despite using raw point clouds and error-prone pseudo-labels, models trained on Bronze achieves similar (PointNeXt) or even better (Swin3D) segmentation accuracy than when trained on the clean Silver dataset. We assume that diversity partially compensates for label noise, allowing models to learn better invariances despite small errors in the labels. In addition, the raw point clouds are denser

Table 4.5: Segmentation metrics on the test set for 3DSES Silver and Bronze, either with real or pseudo labels (and intensity features or not). Intersection over union (IoU) per class, mean IoU (mIoU), overall accuracy (OA), average accuracy (AA).

		Real labels	Intensity	Column	Covering	Door	Exit sign	Heater	Lamp	Railing	Slab	Stair	Wall	Window	Clutter	**OA**	**AA**	**mIoU**
Swin3D	Silver	✅	❌	0.00	89.07	76.40	9.93	74.69	32.24	46.22	86.40	67.75	89.24	54.62	90.42	91.69	84.84	59.75
		✅	✅	5.40	94.35	83.06	9.30	75.27	44.04	37.63	84.08	38.69	85.34	54.99	72.83	90.47	83.39	57.08
		❌	❌	25.47	88.50	61.62	12.96	59.24	30.79	35.94	77.55	36.22	87.61	48.76	71.15	87.49	88.44	52.98
		❌	✅	52.31	95.82	89.01	11.79	65.29	55.28	64.17	82.06	34.32	92.44	54.00	91.92	93.46	89.44	65.70
	Bronze	❌	❌	51.76	95.90	89.37	12.45	65.80	52.25	82.14	86.80	43.15	93.33	60.53	93.59	94.59	93.67	68.92
		❌	✅	59.68	95.97	88.10	41.80	71.59	55.59	77.20	85.81	41.40	93.00	60.89	94.52	94.51	94.37	72.13
PointNeXt-S	Silver	✅	❌	0.00	96.77	67.11	0.00	16.45	69.95	61.75	94.88	83.87	89.26	62.54	80.25	93.30	66.27	60.24
		✅	✅	0.00	97.07	76.66	0.00	38.73	78.11	65.26	94.85	86.97	90.84	67.08	84.35	94.63	70.59	64.99
		❌	❌	0.00	96.53	73.07	0.00	20.33	66.71	2.79	93.50	76.90	90.32	40.60	71.12	92.68	57.60	52.66
		❌	✅	58.44	96.55	69.81	0.00	33.96	67.00	38.90	93.86	83.48	88.12	51.25	73.60	92.58	71.19	62.91
	Bronze	❌	❌	11.21	95.68	85.16	0.00	69.18	66.19	15.97	93.53	80.09	92.62	49.09	82.86	94.57	66.47	61.79
		❌	✅	56.45	96.44	81.39	0.00	79.71	77.40	42.25	93.35	78.33	91.57	56.47	80.94	94.47	77.06	69.53

that the clean versions used in Silver and Bronze and might provide more geometrical information that is more costly to process, but also more discriminative. These observations show the tradeoffs of the three variants of 3DSES, from training on small high-quality data, to larger but noisier point clouds.

4.6.2 Impact of LiDAR intensity

As described in Section 4.2, 3DSES is the only indoor TLS dataset that provides LiDAR intensity. Previously, in Section 3.5, we showed that intensity cannot be calibrated easily to retrieve the reflectance of objects. Since 3DSES contains only LiDAR intensity from one sensor (the RTC360) and are exported from a single software (*i.e.*, Register360), intensity variation across the dataset will be the same. In this section, we included the constructor's intensity as an additional feature in our models to evaluate its impact on semantic segmentation. As shown in Table 4.4 for Swin3D, we observe a 3.0% increase in mIoU when using intensity in addition to color on real labels on Gold. Nonetheless, we observe a decrease for Swin3D on pseudo-labels (2.6%). However, the drop is not consistent on all classes, *e.g.* few classes obtain better IoU. On the other hand, including the intensity for PointNeXt improves mIoU by 15% on Gold real-labels. This shows that intensity helps generalization of smaller models. In Table 4.5, intensity helps Swin3D and PointNeXt in most cases. In comparison, Swin3D trained on Silver variant with pseudo-labels and intensity obtains *better* scores (+12.7% IoU) than without

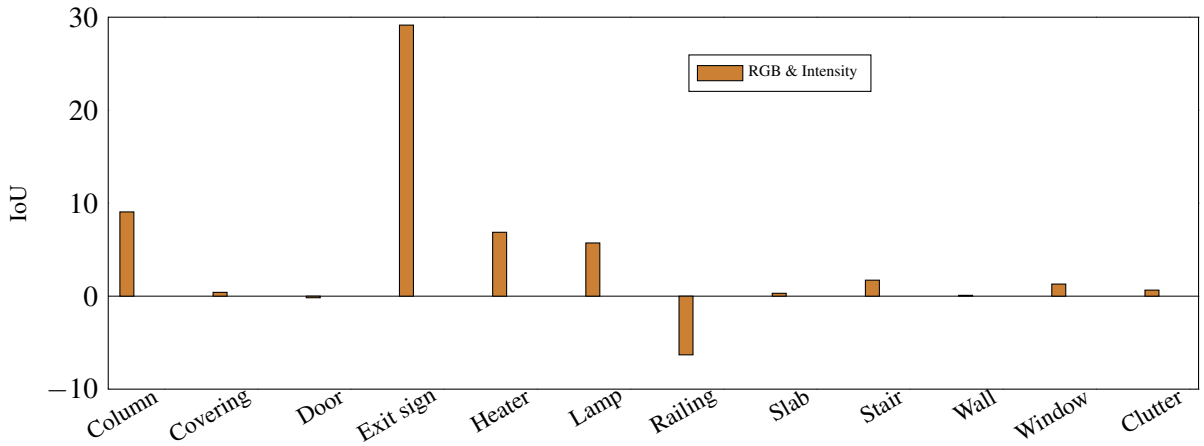


Figure 4.10: Intersection over Union by classes for Swin3D on 3DSES Bronze 🥉 pseudo labels. RGB IoU is subtracted from all RGB&Intensity IoU results for better visualization. Intensity shows promising improvements on some classes compared to the RGB baseline (*e.g.*, "Exit sign").

intensity. The histogram (Fig. 4.10), shows that adding to XYZ and RGB intensity help Swin3D on Bronze to better segment several classes such as "Column", "Exit sign", "Heater", "Lamp". This raises concerns about elements with high reflectivity and suggests that the intensity data from the RTC360 can be helpful.

Intensity & Handcrafted features In previous chapter, we showed handcrafted features can benefit recent deep models. Here we test the contribution of the intensity with previous handcrafted features (LPS_cV and OAS_eS_v) to determine if a combination can be helpful for recent deep models.

Table 4.6 shows that intensity is a discriminative feature for PointNeXt-S. PointNeXt-S benefits from the contribution of intensity and other features (such as LPS_cV and OAS_eS_v), and sees all metrics improve. With major improvements on Gold variant: $+13.5\%$ mIoU attained with intensity and color information. For Swin3D, the results are a bit more contrasted, due to its high number of parameter. The intensity alone did not help Swin3D to perform better on Gold and Silver variants; however, it helped to furnish $+4.0\%$ mIoU on Bronze variant. Overall, the preliminary results could indicate that LiDAR intensity can indeed be discriminative for some classes, especially for larger datasets. Further experiments are required to validate these observations.

Table 4.6: Segmentation metrics obtained with intensity & features on the 3DSES test set. Metrics obtained using different geometric features and two models are compared with the RGB baseline. *Note that results from PointNeXt-S are the results of three runs and the standard deviations are printed in Table C.4*. Contrary to previous results, here the presented results are obtained with multiple runs and the latest checkpoints (see Chapter 3). **Green** indicates an improvement compared to the baseline, while **red** indicates a decrease. Mean intersection over union (mIoU), overall accuracy (OA), average accuracy (AA).

					PointNeXt-S			Swin3D		
	RGB	I	LPS_cV	OAS_eS_v	OA	AA	mIoU	OA	AA	mIoU
Gold	✅	❌	❌	❌	82.18	36.17	30.14	**89.76**	**78.30**	**48.99**
	✅	✅	❌	❌	**90.66**$_{+8.48}$	**49.09**$_{+12.92}$	**43.65**$_{+13.51}$	88.61$_{-1.15}$	68.69$_{-9.61}$	44.80$_{-4.19}$
	✅	✅	✅	❌	89.88$_{+7.70}$	47.86$_{+11.69}$	41.99$_{+11.85}$	89.16$_{-0.60}$	69.51$_{-8.79}$	48.09$_{-0.90}$
	✅	✅	❌	✅	89.69$_{+7.51}$	47.85$_{+11.68}$	42.32$_{+12.18}$	85.96$_{-3.80}$	70.99$_{-7.31}$	43.57$_{-5.42}$
Silver	✅	❌	❌	❌	92.29	63.30	57.34	91.44	85.16	59.61
	✅	✅	❌	❌	93.30$_{+1.01}$	**69.74**$_{+6.44}$	63.44$_{+6.10}$	91.94$_{+0.50}$	76.92$_{-8.24}$	56.23$_{-3.38}$
	✅	✅	✅	❌	**93.90**$_{+1.61}$	68.73$_{+5.43}$	63.44$_{+6.00}$	92.11$_{+0.67}$	84.05$_{-1.11}$	59.81$_{+0.20}$
	✅	✅	❌	✅	93.47$_{+1.18}$	69.42$_{+6.12}$	**63.72**$_{+6.38}$	**93.83**$_{+2.39}$	**86.08**$_{+0.92}$	**61.98**$_{+2.37}$
Bronze	✅	❌	❌	❌	94.47	67.94	62.48	94.32	93.78	68.19
	✅	✅	❌	❌	93.76$_{-0.71}$	74.55$_{+6.61}$	66.89$_{+4.41}$	94.63$_{+0.31}$	**94.39**$_{+0.61}$	**72.17**$_{+3.98}$
	✅	✅	✅	❌	**94.87**$_{+0.40}$	75.22$_{+7.28}$	**68.21**$_{+5.73}$	93.24$_{-1.08}$	92.99$_{-0.79}$	67.69$_{-0.50}$
	✅	✅	❌	✅	94.13$_{+0.34}$	**75.75**$_{+7.81}$	67.91$_{+5.43}$	**95.07**$_{+0.75}$	93.32$_{-0.46}$	70.42$_{+2.23}$

4.7 Conclusion

In this chapter, we introduced 3DSES, a new dataset for semantic segmentation of dense indoor LiDAR point cloud. 3DSES fills the need for indoor TLS datasets designed for building survey and modeling. It contains a unique combination of point cloud labels for semantic segmentation, a georeferenced 3D CAD model with BIM oriented objects and LiDAR intensity, a radiometric feature not provided in existing datasets. We demonstrate that using 3D CAD models to automatically annotate point clouds is a time-efficient strategy that produces pseudo-labels with 95% accuracy compared to a manual ground truth. Moreover, we show that training on pseudo-labels achieves similar performance to training on real ones on 3DSES. We show that segmentation accuracy can benefit from LiDAR intensity in indoor settings, despite radiometry being often ignored in previous works. Segmentation results demonstrate that 3DSES is a challenging new dataset, especially for BIM-oriented classes, *e.g.* small building components such as electrical terminals and safety systems. 3DSES is a unique dataset that contains all the steps required for automated scan-to-BIM: dense point clouds, semantic segmentation labels and a full 3D CAD model. We hope that 3DSES will enable the creation and testing of deep models for multiple tasks, from point cloud segmentation to BIM generation through mesh to point cloud alignment. We will discuss this perspective in Chapter 6. In this chapter, we demonstrate that our methods can effectively emphasize "frozen" data from land surveyor companies and contribute to creating datasets with small objects. We hope this new dataset will stimulate research on indoor point clouds processing and motivate the community to investigate auto-modeling tasks in scan-to-BIM.

5

Synthetic 3D scenes for small objects segmentation

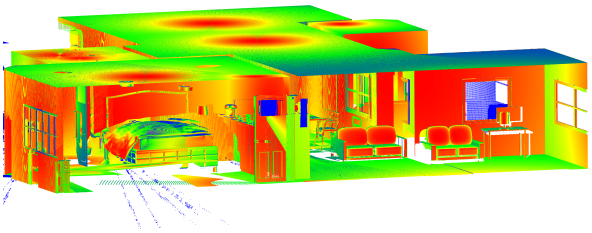
Content

Abstract

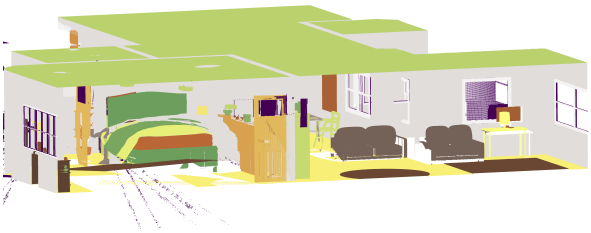
Recent deep learning methods improvements stem from supervised learning and, consequently necessitate labeled data. Meanwhile, point cloud labeling remains a tedious task, typically reserved for expert users which leads to a limited number of datasets. In the previous chapter, we introduced a method that uses existing CAD models from surveying companies to create pseudo-labels. But what if no point cloud data is available? Should we discard this 3D model? Our intuition says no. In this chapter, we present a method to generate fully synthetic 3D point clouds with small objects. First, we generate indoor 3D scenes using a procedural home generator. Our approach relies on infinite procedural generation, which allows us to generate a high amount of 3D scenes without human action. We then use virtual sensors to simulate high-density point clouds. Finally, we discuss perspectives on the use of the point clouds generated for semantic segmentation.

(a) View of a scene from IPCS.

(b) Bathroom of the previous scene



(c) Point cloud with intensity [212]



(d) Point cloud with semantic labels [212]

Figure 5.1: Overview of a 3D scene and its corresponding point clouds. IPCS is a unique synthetic point cloud dataset for indoor point cloud segmentation. It contains indoor 3D scenes and point clouds with intensity and semantic&instance labels.

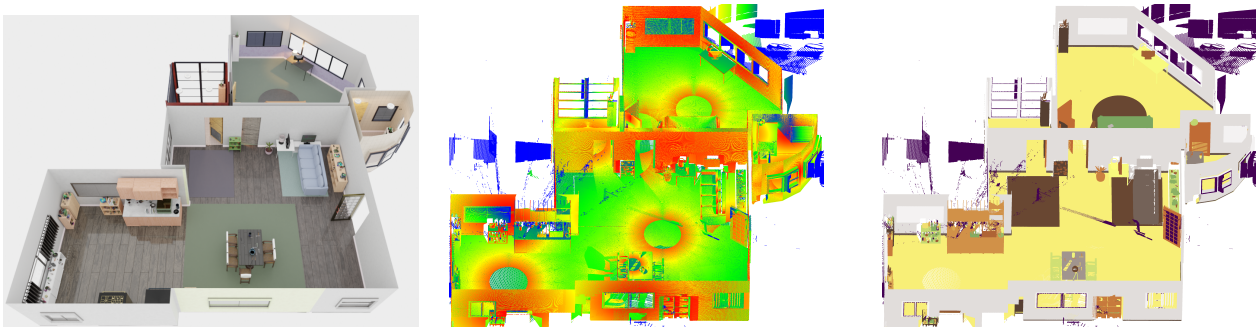
5.1 Benefiting from synthetic 3D scenes

Segmentation of 3D point clouds is an essential stage in a variety of applications, including Building Information Modeling (BIM). As has been observed in other research domains, conventional methods that rely on handcrafted features have been eclipsed by deep learning. However, recent progress in 3D point cloud segmentation using deep learning depends on supervised learning and access to labeled data. Yet, the labeling of 3D point clouds necessitates the involvement of experienced users [119], leading to a limited number of labeled datasets (see Chapter 2). For example, annotating the SemanticKITTI dataset [38] required 1,700 hours of labor for 43,552 scans, and annotating the ScanNet [121] took over 500 hours for 1513 scans. Furthermore, recent methods [105, 106] uses multi dataset training to improve final segmentation, making these approaches label hungry. In the previous chapter, we introduced a method for creating labels, without spending a lot of time on detouring objects in 3D point clouds. However, our method assumes on the availability of both 3D point clouds and 3D CAD models. But what happens when we do not have any point cloud? Do we discard the available 3D models? Thanks to virtual TLS, we can generate point clouds from this existing CAD model [119, 213]. But what if no CAD models are available? Can we extend this method to zero-existing indoor scenes? We are convinced that yes, and propose in this chapter a method to generate synthetic indoor scenes before raycasting 3D point clouds (cf. Fig. 5.1).

Using synthetic data to improve performance on real data is not a new idea. It has already been employed in text recognition [214], text localization [215], 2D object detection [216, 217] and segmentation [218, 219], and even in 3D semantic segmentation of outdoor scenes [119, 220]. A wealth of 3D environments is available [119] in domains such as video games and land surveying companies. In the era of indoor point cloud segmentation, synthetic image datasets can be employed to enable the creation of point clouds and pre-train deep models. However, point clouds generated by these approaches may face photogrammetric challenges such as occlusion and low point cloud density. Recently, some synthetic point cloud datasets have been proposed [119, 220], but these mainly concern outdoor settings (cf. Section 5.2.4). Point clouds collected outdoors using mobile laser scanning (MLS) differ from those acquired indoors using terrestrial laser scanning (TLS) in terms of point density and object distribution. This limits the use of outdoor datasets to pre-train deep models for indoor tasks.

In this chapter, we present a pipeline for creating synthetic TLS point clouds in indoor environments (cf. Fig. 5.2). Prior to considering the use of surveyor CAD models to improve the segmentation of small professional objects (*e.g.*, alarms, extinguishers and outlets, . . .), we opted to generate indoor scenes using a procedural

generator to validate our approach. We introduce IPCS (*"Indoor synthetic Point Cloud Segmentation"*): an indoor synthetic LiDAR dataset for indoor point clouds segmentation. IPCS consists of virtual TLS point clouds containing a large number of small objects (*e.g.*, books, forks, desk lamps and spoons). IPCS has been developed to enable the community to pre-train new deep models on TLS point clouds and improve the accuracy of small object segmentation.



(a) Blender scene generated by Infinigen [221].

(b) Point clouds with BLAINDER [212] colorized with the simulated intensity

(c) Point clouds with BLAINDER [212] colorized with the semantic labels.

Figure 5.2: Modalities of IPCS. Blender scenes are generated using Infinigen [221]. Then, a virtual TLS [212] moves inside each indoor environment to produce synthetic point clouds with intensity. This enables us to assign object names to each point. These objects are then grouped into different semantic classes.

5.2 Previous works

The idea of generating synthetic data to train deep models is not new [222, 223]. Many studies have looked into the lack of labeled data in 3D fields. These studies have developed methods to create synthetic point clouds (Section 5.2.1 and Section 5.2.2), generate synthetic environments (Section 5.2.3), and create 3D synthetic datasets (Section 5.2.4).

5.2.1 Virtual sensors

In 2007, Lohani et al. [224] developed a simulator dedicated to airborne acquisition. The authors created a simulator, with a graphical user interface that can generate scenes. A virtual sensor has also been used to model full-waveform signals and calculate beam divergence to improve tree height estimation [225]. However, previous simulated ALS can only work with "2.5D" environment and only measures from above [226]. This means that it is not possible to measure from the ground, and so the developed approach cannot be used to get TLS measurements. To enable TLS acquisition, the pipeline must enable the representation of complex

objects and be suitable for measurements at close distance and in any direction [226]. Hodge [227] is one of the prior studies dedicated to modeling beam divergence within the scope of TLS. However, the system was designed to work with small scenes ($< 1 \times 1m$) and supported only one scan position. One of the first approaches using 3D geometry to enable TLS acquisition is presented in [228]. Then, an approach especially designed for TLS measurements was proposed to simulate the interaction between LiDAR and a tree [229]. Blensor [230] is another virtual scanner simulation. It is fully integrated with Blender [231] and enables the simulation of multiple sensors, such as rotating LiDAR and time-of-flight (ToF) cameras. In addition, Blensor sensors support ray reflection. As a virtual sensor, Blensor stores the ground truth for each ray. It relies on a community software named Blender to model dynamic or static environments. This enables researchers to reuse existing 3D scenes to create synthetic point clouds without having to recreate environments in other software. HELIOS [232] is an alternative developed to support the simulation of multiple virtual sensors: TLS, MLS and ALS. The advantage of HELIOS is its flexibility: the XML format supports a broad range of scenarios and requirements. The platform uses triangle meshes to define scenes and supports various formats, such as ".obj", ".geotiff", and ".xyz" but not Blender scenes. Other works have focused on providing additional data for deep learning approaches. For example, SqueezeSeg [233] designed a LiDAR simulator [234] in Grand Theft Auto V. They position a LiDAR scanner and a camera on an autonomous vehicle in the game to acquire 3D point clouds and images. This setup enabled the authors to double the number of samples in their training set. Finally, to produce more realistic point clouds than those in SemanticKITTI, the authors added noise to the 3D point clouds. VRScan3D [235] introduces a virtual reality environment, designed for student applications. VRScan3D simulates brand TLS (such as RTC360, Faro), however, TLS needs to be moved manually and labels are not recorded. Similar to Blensor, BLAINDER [212] is an add-on for the Blender modeling software. Focusing on LiDAR and Sonar simulation, BLAINDER enables users to customize sensors and implement them in various environmental conditions (*e.g.* rain, fog). A more recent paper [220] focuses on simulated outdoor point clouds using the well-known CARLA [236] simulator. CARLA (Car Learning to Act) is an open-source simulator dedicated to autonomous driving research. It is used to simulate urban environments in order to evaluate deep models for tasks such as autonomous driving. CARLA simulates environments and provides an interface to interact with agents. These environments comprises 3D models of buildings, roads, traffic signs, etc., as well as dynamic objects such as cars and humans. HELIOS++ [226] is the direct successor of HELIOS. Compared with its predecessor, HELIOS++, reduced computation times in most scenarios. However, HELIOS++ supports the same files as HELIOS and does not directly support Blender scenes. Although an add-on has been developed

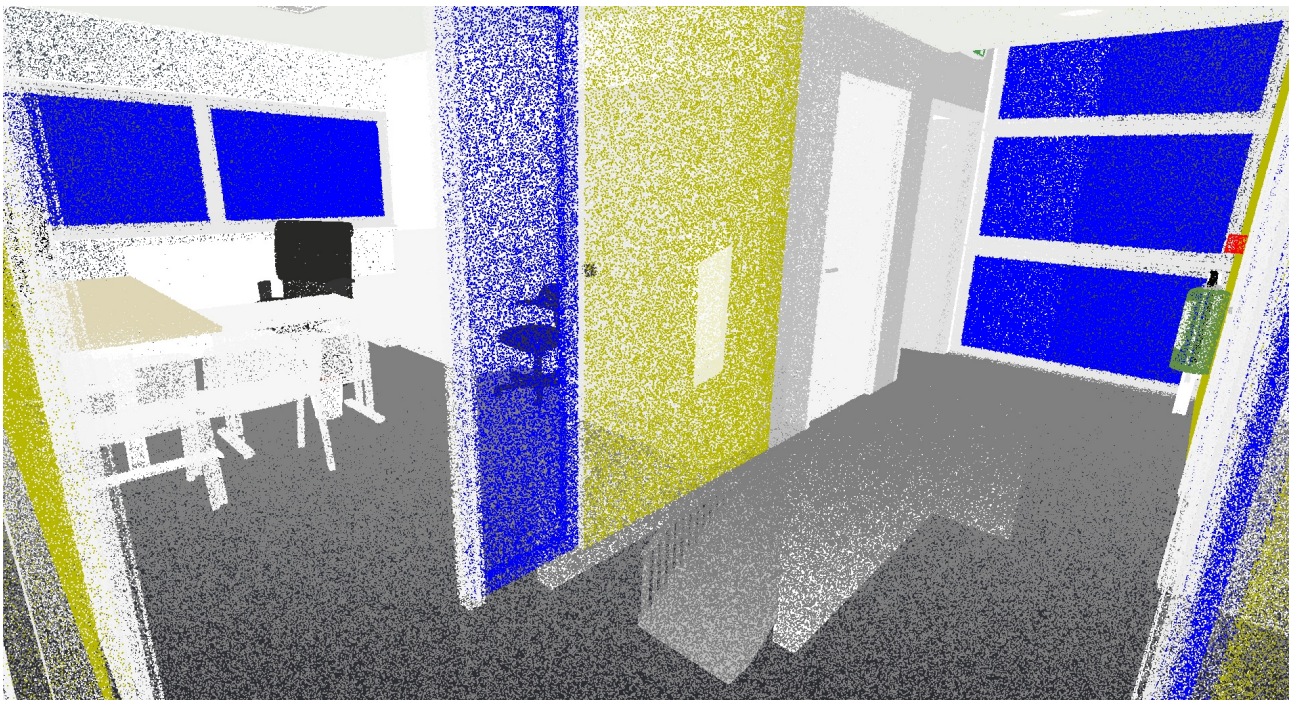


Figure 5.3: View of a 3D point clouds sampled on 3DSES's CAD model. The 3D points are colored using the color of the CAD elements.

"Blender2Helios" [237], it is designed for Blender 2.8. Regarding proprietary CAD models, DynamoPCsim [238] enables TLS acquisition with a visual programming environment, but it's limited to BIM models loadable in Revit [208].

5.2.2 Alternatives to virtual sensors

Other approaches have been proposed to create 3D labeled point clouds without relying on virtual sensors.

Sampling from 3D mesh The easiest way to produce a point cloud is to sample points on a mesh (Fig. 5.3). If the mesh is labeled, this creates 3D point clouds with high-quality labels. As there are lots of Building Information Models (BIMs) [239], 3D coordinates can be sampled on BIMs [240, 239] to create synthetic labeled datasets. This is now being used in the field of heritage [241], where it is possible to extract point clouds from 3D models of real historical buildings.

Deep generation These methods use convolutional deep belief networks [148], GANs [242], normalizing flows [243] or autoencoders [244] to generate synthetic point clouds. However, most of these methods only focus on objects and do not generate complex scenes. Generating complex 3D scenes is a more arduous process. Neural

Table 5.1: Overview of some datasets with indoor environments from the literature. These datasets encompass a variety of object categories and were produced using various methods, including optimization algorithms, artistic layouts and procedural rules. This dataset can be used to recreate 3D point clouds using photogrammetric approaches (see Section 5.2.4).

Name	Environment	Classes	Extent[1]	Free Assets	R/S	Simulator
SceneNet [248]	Indoor	255	NA	✅	S	Optimizer
InteriorNet [249]	Indoor	158	NA	❌	S	Artists layouts
Replica [250]	Indoor	88	18	✅	R	Real-world scans
Structured3D [251]	Indoor	40	3500	❌	S	Artist layouts
OpenRooms [252]	Indoor	44	NA	✅	R	Real-world scans
Hypersim [219]	Indoor	19	461	❌	S	Artist layouts
3D-Front [253]	Indoor	31	NA	✅	S	Artist layouts
ProcTHOR-10K [254]	Indoor	108	10K	✅	S	Procedural rules
HSSD-200 [255]	Indoor	28	211	❌	S	Artist layouts
Aria Synthetic Environments [256]	Indoor	NA	100K	✅	S	Procedural Rules
SpatialGen [257]	Indoor	NA	12328	NA	S	Artist layouts

NA: information Not Available. [1] Number of scenes

Radiance Fields (NeRF), for example, are used to generate synthetic LIDAR data that closely resembles real 3D data for use in autonomous driving [245, 246]. Another approach, SemCity [247], attempts to generate "real" outdoor point clouds. It uses a 3D diffusion model for inpainting (adding objects) and outpainting (enlarging the scene) tasks, learning from real outdoor scenes.

5.2.3 3D scene generation

Virtual sensors can generate high-quality synthetic point clouds from 3D scenes. However, they do not generate the 3D scenes themselves. In order to simulate 3D point clouds, it is necessary to produce the 3D data. Human modeling of 3D scenes is a time-consuming process, particularly when focusing on detailed scenes with small objects such as cups, trinkets and towels. Consequently, methods that do not rely on human involvement have been developed. In this section, we list some approaches dedicated to 3D scene generation.

Rely on humans One solution for creating indoor synthetic datasets is to rely on human environment artists. InteriorNet [249] was the first large-scale dataset to use human-created interior layouts for generating indoor environments and rendering images. More recently, Structured3D [251], HyperSim [219] and 3D-Front [253], HSSD-200 [255] introduced photorealistic datasets based on 3D environments created by professional designers and generate 2D images. Replica [250] and OpenRooms [252], are based on real-world scans. The previous

dataset does not furnish point clouds, but only images. Although it is possible to recreate point clouds from images, we discard this idea here due to differences between photogrammetric and TLS point clouds. In terms of 3D point cloud datasets, the single dataset, BelHouse3D [258], also relies on real indoor environment models created by humans (*e.g.*, real-world acquisitions). However, certain datasets do not provide the raw 3D scenes, rendering ray-casting of 3D point clouds impossible. Even if the raw 3D scenes are available, previous datasets are not suitable for generating a large variety of indoor environments, since they rely on human intervention and a fixed library of assets. Therefore, we opted not to use them to generate synthetic point clouds.

Neural 3D based generation methods Recently, some papers have attempted to generate indoor scenes using deep models trained on labeled 3D data [259]. The methods grouped in this section can be categorized in scene-parameter, graph, semantic layout or implicit layout. Scene parameter methods attempt to learn object relationships from labeled data. These rules are based on factors such as reach (*e.g.* a wheelchair needs to be close to a desk and less likely to be close to a bathroom), visibility (*e.g.* a TV and a sofa), lighting, and access (particularly with regard to doors). FastSynth [260], for instance, uses deep convolutional generative models to insert objects into scenes. LayoutEnhancer [261], ATISS [262] and SceneFormer [263] utilize Transformer architectures for the prediction of object parameters (*e.g.* class, location, size). SceneFactor [264] is a method that simplifies the editing of 3D scenes using a text guidance and diffusion approach. SceneFactor first generates a semantic box from the text input and then generates geometry from this box. However, the constraints learned by previous methods are limited by the quality of the raw data [259]. In contrast to methods based on scene parameter, scene graphs methods enable models to impose spatial relations. For instance, PlanIT [265] divides the layout generation problem into two phases: planning and instantiation. The planning phase is achieved with a generative model over graphs, and the placement of objects (*i.e.* instantiation) is performed using 2D images and 2D networks to place objects as realistically as possible. Para et al. [266] employ Transformers for graph generation; however, they use optimization (*i.e.* a Linear Programming problem [267]) to generate the final layout. SceneHGN [268] utilizes a hierarchical approach to generate indoor scenes, allowing users to edit parts of the geometry at various scales. Semantic layout can be either 2D [269] or 3D [270, 271], and furnishes a representation that guides 3D environment generation, ensuring coherence in object placement. Implicit layout methods [272, 273] are employed to learn the layout, which is then decoded (*i.e.* using NerF or a 3D Gaussian) for 3D generation [259]. Recently, SpatialGen [257] introduced a multi-modal diffusion model for generating high-fidelity indoor scenes. However, this approach relies on fixed data (human-designed floorplans) and struggles to generalize to highly diverse scenes. While generating scenes with high semantic consistency,

neural 3D methods are limited by their time efficiency and the datasets available.

Image or Video generation In addressing the scarcity of 3D labeled data, some approaches attempt to leverage images [274, 275] or videos [276, 277, 278] to generate 3D scenes [259]. However, image-based approaches failed to capture the global context, while video-based approaches encounter time efficiency issues.

Procedural generation One alternative approach to creating 3D scenes is procedural generation, whereby predefined rules are used to create and place objects [221]. This process allows for the creation of multiple variations of scenes without the need for real-scan or manual modeling [221]. Procedural generation can be categorized into three approaches: methods based on rules, methods based on optimization, and methods employing Large Language Model (LLM) [259]. One early alternative for creating 3D environments is to use hardcoded programs (*i.e.* ruled-based methods) to represent object constraints [279]. Another option is to use real indoor environments (*e.g.* man-made 3D scenes). For example, SceneNet [248] employs a library of indoor objects (such as the Stanford Database [280]) and places them with a hierarchical simulated annealing optimization in a real indoor environment to create new scenes. Furthermore they pay close attention to textures, ensuring that they provide variability in the scenes. However, this approach is limited to a fixed list of assets. An alternative approach is to use an optimizer, *i.e.* consider the generation as an optimization problem that minimizes predefined constraints. For instance, LUMINOUS [281] introduces a new method for indoor generation method called CSSG (Constrained Stochastic Scene Generation). This method synthesizes scenes until the layout satisfies the user's requirements. Procthor [254] utilizes multi-stage conditional sampling to iteratively define a comprehensive indoor layout, beginning with a floor plan, incorporating walls and doors, and subsequently adding objects. However, these two previous methods are limited in terms of objects and materials. Infinigen [282] is a rule-based method that enables the generation of outdoor scenes and relies on Blender [231]. Infinigen's generation process does not rely on external assets; each object or texture is procedurally generated. Each scene generated by Infinigen, as well as object arrangement, is based on mathematical rules and can be simply modified to create variety in the environment. Moreover, Infinigen is equipped with a procedure that details the process of creating new constraints, and this process is accessible to non-expert users (*i.e.* those who benefit from the simplicity of Blender node graphs). Infinigen is limited to natural scenes, but was extended to indoor scenes with Infinigen Indoors [221]. Infinigen Indoors, like its predecessor, is not dependent on external resources. This allows the pipeline to generate a wide range of indoor scenes. More recently, LLMs have been employed to generate layouts from prompts (GraphDreamer [283], LayoutGPT [284]), and to adjust

Table 5.2: Comparison of the characteristics of various synthetic point cloud datasets from the literature. It can be seen that the IPCS dataset is unique in that it is the only indoor synthetic dataset acquired by a virtual TLS. The IPCS dataset surpassed other indoor datasets in terms of number of points, demonstrating a very high point density. It is also worth noting that IPCS is currently the only synthetic dataset that incorporates intensity from the virtual sensor.

Name	Environment	Classes	Points (M)	Color	Intensity	R/S	Simulator	Source
SynthCity [119]	Outdoor	9	367.95	✅	❌	S	Blensor	MLS
KITTI-CARLA [287]	Outdoor	23	4500	✅	NA	S	CARLA	MLS
Paris-CARLA-3D [288]	Outdoor	23	700	✅	❌	R+S	CARLA	MLS
SynLiDAR [289]	Outdoor	32	19482	❌	✅	S	AirSim	MLS
SP3D [213]	Outdoor	10	34	❌	✅	R+S	HELIOS++	MLS
SynthmanticLIDAR [220]	Outdoor	30	7.2	❌	❌	S	CARLA	MLS
BelHouse3D [258]	Indoor	19	1310	❌	❌	S	Sample	3D model
IPCS	Indoor	**80**	**>64000**	❌	✅	S	BLAINDER	TLS

NA: information Not Available.

parameters of rule-based methods (Holodeck [285]). SceneScript [256] also utilizes LLMs and has released an indoor dataset called Aria Synthetic Environments to test layout estimation tasks. However, LLMs can produce unnatural placements [286].

5.2.4 Synthetic point cloud datasets

As previously discussed, a key bottleneck for point cloud segmentation is the availability and the accuracy of labeled data. One solution is to generate synthetic data for pre-training deep models. This section introduces some synthetic datasets that can be used for this purpose. Table 5.2 presents some characteristics of theses datasets.

Outdoor synthetic datasets All the datasets presented in this section were generated using an MLS sensor to produce 3D point cloud data. One of the earliest examples is SynthCity [119], which modeled 3D scenes in Blender and then used a MLS sensor with Blensor to generate 3D point clouds. SynthCity also provided RGB colors alongside XYZ coordinates. While SynthCity used Blensor, the KITTI-CARLA [287] and Paris-CARLA-3D [288] datasets relied on the CARLA simulator, as their names suggest. The most recent outdoor synthetic dataset, SynthmanticLIDAR [220], also used this simulator. The Paris-CARLA-3D dataset contains synthetic data as well as real point clouds with human labels acquired in Paris. These real labels allow the authors to evaluate transfer methods on real point clouds. SP3D [213] also includes point clouds with manual annotation.

As with previous datasets, SynLiDAR [289] used virtual scenes created by professionals using Unreal Engine 4 [290]. They then used AirSim [291] for LiDAR simulation. On SynLiDAR, the intensity was generated by a deep model trained on real point clouds. Furthermore, a Point Cloud Translator deep model was employed to generate more realistic point clouds. This deep model was also trained on real point clouds.

Additionally, MLS is a common acquisition method for outdoor point clouds, but differs from TLS, which is most commonly used in indoor scenes. Moreover, these outdoor point clouds do not have the same level of detail as indoor point clouds, which limits the use of these datasets for indoor applications.

Indoor synthetic datasets Structured3D [251] is a photorealistic indoor dataset comprising house designs created by professionals. It consists of 21,835 rooms and 3,500 scenes. The authors used professional indoor designs to create 2D renderings with labels. Although Structured3D does not provide XYZ point cloud coordinates, recent semantic segmentation architectures [186, 106] have used it to pre-train their point cloud segmentation models. Swin3D [186] uses all Structured3D RGB-D images, along with their intrinsic and extrinsic camera parameters, to reconstruct 3D point clouds. Another dataset, Hypersim, is dedicated to 2D indoor renderings [219]. The Hypersim authors benefit from 461 indoor scenes created by professional artists, which are used to generate 77,400 images. Similar to Structured3D, HyperSim includes parameters (such as focal length, rotation matrix, and translation vector) that enable the reconstruction of a 3D scenes and point clouds. However, to the best of our knowledge, this dataset has not been used to pre-train deep models for the semantic segmentation of point clouds since 3D models are not available. BellHouse3D [258] is a synthetic indoor dataset designed for 3D semantic segmentation. It was constructed using a real-world environment comprising 32 Belgian houses, acquired with an RGB-D sensor. The authors then used Blender and the raw point clouds to create 3D models of all the houses. The XYZ coordinates of the point clouds were obtained by sampling points on the surfaces of the 3D objects generated by Blender. Furthermore, the authors added a test set with occlusions to enable evaluation of out-of-distribution data. Rather than removing points arbitrarily or randomly, the authors used the original viewpoints of the RGB-D cameras to simulate real occlusions. The 3D-Front dataset [253] is not designed for the semantic segmentation of 3D point clouds. However, it contains a large number of 3D scenes (18,997 rooms) and can be used with "sampling" methods to create synthetic 3D point clouds. For example, this dataset enables the generation of synthetic point clouds for pre-training a transfer network [292]. However, some objects in 3D-Front do not contain textures (*e.g.* wall, floor) and require colors to be set arbitrarily.

These datasets are primarily dedicated to 2D rendering rather than 3D rendering. Furthermore, some datasets

do not provide the 3D models. Moreover, their creation process relies on humans, which means that they cannot be used to generate an very large number of indoor environments.

IPCS is the only indoor dataset that contains *3D point clouds generated by a virtual TLS*. A simulated intensity value is also available for each XYZ coordinate. IPCS provides a large number of semantic classes, with *80* classes as well as instance labels. With its high-density point cloud, IPCS provides a large number of points (over 64 billion), twice as many as SynLiDAR.

5.3 IPCS (Indoor synthetic Point Cloud Segmentation)

In this section, we present our framework for generating synthetic point clouds. First, we focus on generating 3D synthetic indoor environment (Section 5.3.1). After generating the 3D scene, we use a virtual sensor to produce the synthetic indoor point clouds(Section 5.3.2).

5.3.1 IPCS 3D scene generation

The objective of this chapter is to generate indoor point clouds without any existing data. Therefore for the 3D generation, we do not assume that CAD models are available. Moreover, to pre-train deep models, we need a huge amount of data with variety, consequently, we cannot rely on human artist layouts and be limited by assets. For these reasons, we turned to procedural generation and more precisely Infinigen Indoors. With its user-friendly interface and command-line functionality, Infinigen Indoors enables us to generate a large number of scenes. Fig. 5.4 illustrates several indoor rooms that were generated by Infinigen Indoors. This procedural generator enables us to create scenes with a high level of detail, as shown in Fig. 5.6. Infinigen Indoors was used to produce *384 scenes* for more than *100 000* m^2 of indoor environments. Table 5.4 illustrates the distribution of room types in IPCS. Our indoor scenes contain classical indoor room types (*e.g.* bathrooms, bedrooms, hallways, ...). We refer readers to Raistrick et al. [221] for a list of available room types in Infinigen Indoors. Fig. 5.5 plots the distribution of objects in IPCS and demonstrates the diversity of objects in our synthetic dataset.

5.3.2 3D point clouds simulation

Previously, we opted for a procedural approach to generate 3D indoor scenes with high levels of detail in Blender. To avoid an additional step (*i.e.*, exporting Blender scenes to another format), we limit ourselves to virtual sensors working with Blender scenes. Due to this constraint, we had to ignore certain virtual sensors,

(a) Bathroom (b) Bedroom

(c) Kitchen (d) Living room

Figure 5.4: Examples of indoor renders of IPCS created using Cycles [293] in Blender [231]. These renders were achieved using different rooms, such as “bathroom”, “bedroom”, “kitchen” and “living room”, containing everyday items (such as microwaves, sinks, towels, . . .).

Table 5.3: Area of IPCS. IPCS covers an area of more than 100 000 m^2 with different sizes of scenes.

Number of scenes	**Total area**	**Mean area**	**Max area**	**Min area**
387	104 831.37 m^2	270.88 m^2	789.75 m^2	132.38 m^2

Table 5.4: Distribution of scene categories across IPCS. For a list of available room types in Infinigen Indoors, refer to Raistrick et al. [221].

Category	Balcony	Bathroom	Bedroom	Closet	Dining-room	Garage	Hallway	Kitchen	Living room	Stair case	Utility
Number	195	887	737	443	388	131	244	389	434	90	161

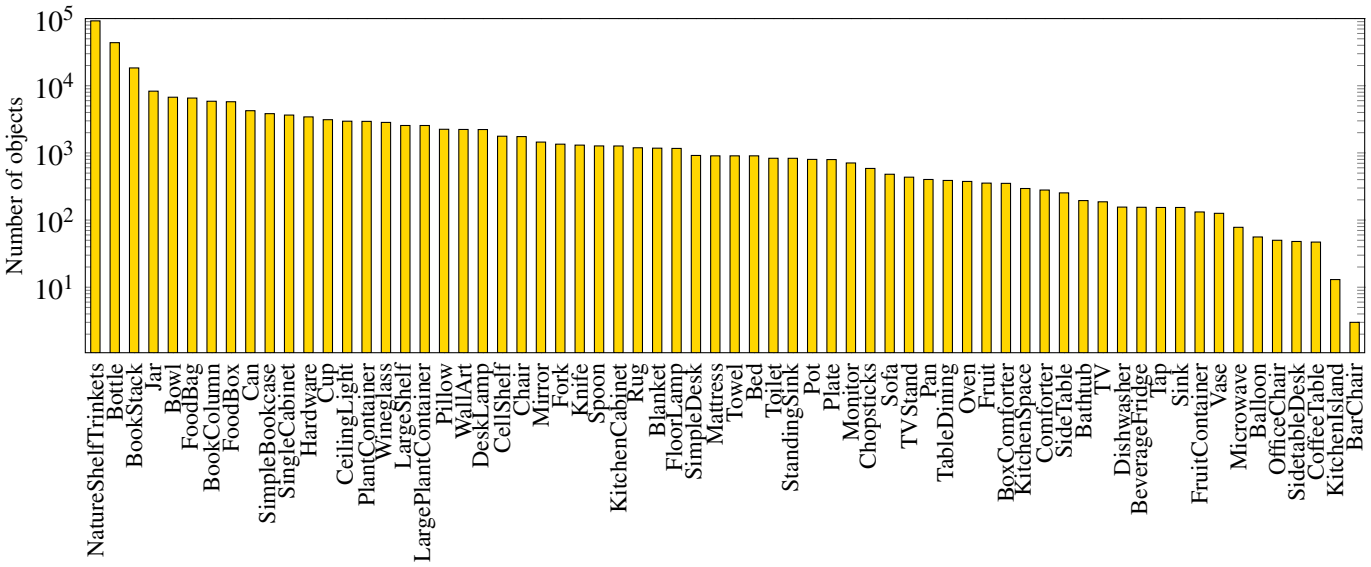


Figure 5.5: Distribution of IPCS objects. This graph illustrates the variety of objects in IPCS thanks to the procedural generation.

(a) Fork and spoon on table

(b) Shelf containing several small objects

Figure 5.6: Scenes rendered by Infinigen Indoors contain a variety of small objects, such as books, forks, spoons

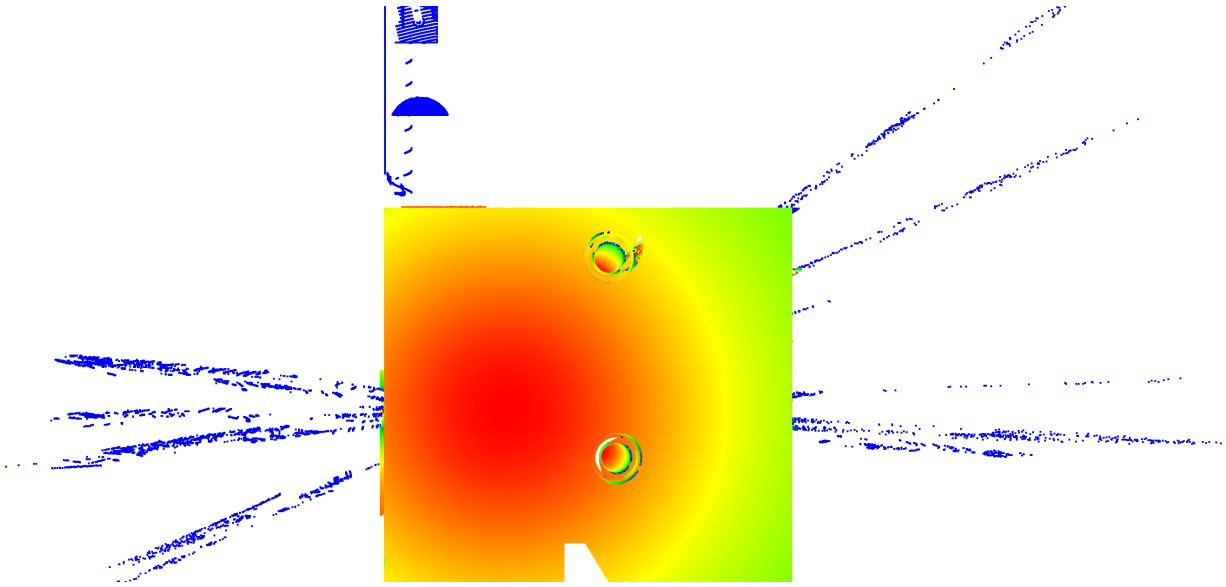


Figure 5.7: Visualization of a point cloud from IPCS. The scanning artifacts are in blue.

such as HELIOS, VRScan3D and HELIOS++. Although Blensor is integrated with Blender, it works with an outdated version of the software. Therefore, to raycast point clouds in our Blender scenes, we have chosen to use BLAINDER. BLAINDER enables raycasting via Blender command-line and is therefore particularly well-suited for our use case. BLAINDER focuses on active distance measurement and simulates a time-of-flight (ToF) LiDAR, as detailed in Chapter 2. Moreover, we enabled reflection and refraction in Blender to generate "scanning artifacts" and render a scene that is more realistic (see Fig. 5.7).

BLAINDER also enables the recording of an intensity information for each point cloud (cf. Fig. 5.8a). BLAINDER advances that, in a perfect case (*e.g.* mirror or smooth surface), the reflection law should be applied. However, under the assumption of the Lambertian model, the ray is reflected in multiple directions, making reflections complex to simulate [212]. To record intensity, BLAINDER employs Eq. (5.1) and uses R, the Bidirectional Reflectance Distribution Function (BRDF), and the incidence angle, denoted by θ:

$$I_r = R \cdot I_e \cdot \cos(\theta) \tag{5.1}$$

Therefore, we recorded the intensity during the virtual TLS acquisition and included it in our dataset.

In this chapter, we aim to simulate the RTC360 scanner from Leica Geosystems, a commonly used TLS in indoor surveys. This TLS is a high-speed 3D scanner that incorporates a high-dynamic-range (HDR) spherical imaging system. For the purposes of this study, colorimetric acquisition is not crucial, since we do not generate images from BLAINDER. The RTC360 is a rotating scanner that uses time-of-flight (ToF) technology to measure

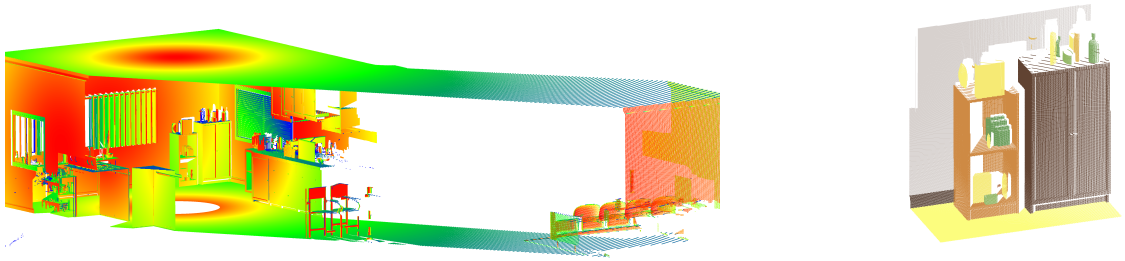
distances, as well as Waveform Digitizing to improve distance measurement accuracy. As previously observed, the field of view (FOV) is a critical parameter for virtual 3D point cloud acquisition. For the RTC360, the FOV is as follows: 360° for the horizontal circle and 300° for the vertical circle. Consequently, the RTC360 has a cone with a 60° angle below the scanner within which no objects can be scanned. Regarding resolution and distance range, the RTC360 offers three setups:

1. High resolution: 3 mm at 10 m with distance range $\in [0.5\,\text{m}, 65\,\text{m}]$
2. Medium resolution: 6 mm at 10 m with distance range $\in [0.5\,\text{m}, 130\,\text{m}]$
3. Low resolution: 12 mm at 10 m with distance range $\in [0.5\,\text{m}, 130\,\text{m}]$

Since the highest resolution produces very dense point clouds, we opted for the minimal resolution for our simulations. This resolution allows us to generate high-density point clouds while ensuring that the IPCS point clouds can be processed by the community. For the distance range, we kept the original parameters from the RTC360 user manual: $d_{min} = 0.5m$ and $d_{max} = 130m$ (where d_{min} is the minimum distance at which an object can be scanned by the RTC360, and d_{max} is the maximum distance at which an object can be digitized).

Door opening and pathway When generating indoor point clouds, it is essential to consider both the path of the virtual sensor and the door openings to reduce real-world acquisition. In real-world acquisition, numerous scans are achieved to scan all the environment, even below doors. These scans below doors serve to link room scans with hallway scans, which is essential for registering 3D point clouds. In our method, we simulate the way a person moves around an indoor environment when scanning it. This consists of fully opening a door and performing a scan below it. After performing this scan, the door in the virtual scene is opened at a random angle, and the virtual TLS moves to the middle of the room. We repeat the process for all rooms in the scene to obtain a point cloud of all the 3D scene.

IPCS point clouds statistics The IPCS point clouds are under generation, however, for now, the virtual TLS acquisition concerns *385 scenes*, corresponding to 103 717 m^2. IPCS actually contains *7356 virtual scans for 64 872 867 884 points*.



(a) Crop section of a kitchen room of IPCS

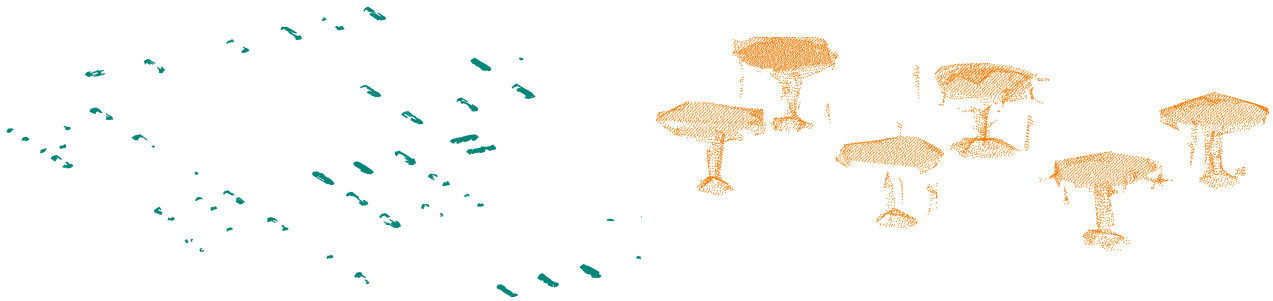
(b) Shelf containing many small objects

Figure 5.8: Thanks to Infinigen Indoors, the point clouds generated by BLAINDER [212] contain several small objects. Furthermore, BLAINDER enables us to produce point clouds with intensity (see Fig. 5.8a) and semantic labels (see Fig. 5.8b).

5.4 Perspectives

This section describes the preliminary use of IPCS for point cloud segmentation using deep learning. Our study employs a state-of-the-art algorithm: Superpoint Transformer (SPT) [294]. As IPCS is the largest and most dense synthetic dataset for indoor point cloud segmentation, we need to be cautious about the training and inference times of the models we choose. For this preliminary usage of IPCS, we use a subset of IPCS comprising of 1792 virtual scans and 15 781 512 046 points. As our primary objective is to evaluate the contribution of IPCS on S3DIS [120], we apply the same preprocessing as SPT uses for S3DIS, except with regard to features. We do not use color and elevation features for superpoint creation and during training. We do not use them, since IPCS does not provide color information, and elevation computation is based on a heuristic that currently yields ambiguous results. For training on IPCS, we use 1285 scans for training, 330 scans for validation and 117 scan for testing the model. The hyperparameters used for training are as follows: epochs $= 100$, AdamW optimizer, CosineScheduler, lr $= 0.1$ and weighted CrossEntropyLoss. The batch size is equal to 1, since all scans of an area are merged to create the superpoint (the same process as for S3DIS).

On our small IPCS test set, we obtain a mIoU of 61.16%. We then employ this model directly on Area 5 point cloud from S3DIS. Fig. 5.9 shows promising result. Without fine-tuning, the model trained on IPCS is already capable of correctly segmenting some points in S3DIS (*e.g.* “Ceiling light” and “Table dining”). Further experiments are now needed to validate the interest of IPCS for real indoor point cloud segmentation.



(a) Point from S3DIS tag as “Ceiling light” (b) Point from S3DIS tag as “Table dining”

Figure 5.9: 3D points from Area 5 of S3DIS [120] segmented using a deep model trained on a subset of IPCS.

5.5 Conclusion

In this chapter, we reviewed methods for generating 3D indoor environments and for raycasting synthetic 3D point clouds. We highlighted that no existing approach combines 3D environment generation and virtual TLS acquisition in indoor environments. No TLS point cloud datasets are available to pretrain deep models. We proposed a method for generating high-density TLS point clouds with detailed small objects. We first benefited from an indoor procedural generator to generate high-quality scenes containing a variety of common objects. Then, in line with land surveyor practices, we moved a virtual TLS inside each room to produce to 3D scans. Our approach enabled us to generate the largest synthetic indoor dataset, covering with more than 64 billion points covering an area superior to most of real indoor dataset: 103 717 m^2. IPCS is just the beginning, and we hope that the community will use this large-scale dataset to pre-train deep models for improve accuracy in downstream tasks on real point clouds. We will discuss this perspective in Chapter 6.

6

Conclusion

Content

6.1 Summary of contributions

In this thesis, we attempt to enhance deep learning-based segmentation of small objects, while enabling the use of existing data from land surveyor companies.

Improving segmentation with handcrafted features First, we study the contribution of expert features. We demonstrate that recent deep models, particularly the smallest architectures, benefit from expert features. Furthermore, we demonstrate that handcrafted features can assist models in recognizing certain under-represented classes of small objects. However, since handcrafted features may be unavailable at test time, we introduce a DropFeatures strategy, which randomly resets features during training. This strategy was designed to make deep models more robust to the absence of features. Our results demonstrate that the DropFeatures strategy not only makes the model more robust, but it also enhances deep segmentation performance. We also conducted an experiment on TLS intensity, and have shown that intensity can vary along the same element, as well as posing difficulties in normalizing it. With this study on handcrafted features, we can benefit from older surveyor point clouds without relying exclusively on XYZ coordinates when color is absent. However, these older 3D point clouds often do not come with labels but include a CAD model.

A realistic indoor point cloud segmentation dataset To address the issue of insufficient labeling, particularly with regard to small objects in older land surveyor data (*i.e.* 3D point clouds), we introduced two significant contributions in Chapter 4. First, we open-sourced a LiDAR dataset called 3DSES, which is designed for the semantic segmentation of small objects. We also introduced a model-to-cloud alignment algorithm to benefit from existing 3D models, in the hope that it will help the community to label 3D point clouds more efficiently. By introducing small objects, high-density point clouds and intensity with 3DSES, we challenge existing deep models and the community to generate methods that can process real indoor TLS point clouds. We also showed that the intensity can improve deep learning accuracy despite the difficulty of calibrating it across different scanners. However, what if no labels, point clouds, nor CAD models are available?

Synthetic 3D scenes for small objects segmentation In this chapter, we tackled the lack of annotated 3D data. We presented a large-scale synthetic dataset containing detailed indoor point clouds. We proposed a method that first generates indoor scenes containing small objects (such as forks, spoons, etc.), for an area greater than *103 000* m^2. We then used a virtual TLS to generate high-density indoor point clouds with intensity. Using a virtual TLS allowed us to obtain high-quality semantic labels (*i.e.*, 80 labels) as well as instance labels for over *64 billion* points. We hope that this dataset will help the community to pre-train supervised deep models

or design self-supervised deep models for the semantic and instance segmentation of point clouds.

6.2 Perspectives

The contributions previously presented enabled us to enhance segmentation of real indoor TLS scans with a high level of detail. However, some questions remain. These can be categorized as follows: ongoing work (Section 6.1) and long-term perspectives (Section 6.2.2).

6.2.1 On going work

Handcrafted features In this thesis, we used handcrafted features to improve deep learning segmentation (see Chapter 3) with the aim of benefiting from older surveyor acquisitions. However, these acquisitions often lack color information. Further experiments involving only features and 3D coordinates are required to validate the contribution of features, particularly in cases where RGB data is unavailable. In Chapter 3, we showed the potential of handcrafted features for small objects and smallest architectures. Further experiments are needed to confirm this contribution, for example with other small architectures (*e.g.* SuperPoint Transformer [294], RandLA-Net [101]) and datasets containing small objects (*e.g.* ScanNet++ [39]). This contribution could be pivotal for integrating deep learning into industrial applications. Industrial cases often involve segmenting small objects, such as manual triggers and valves, and companies typically lack access to high-performance GPUs.

Radiometric information "Does the intensity improve performance when mixing multiple TLS acquisitions?" In Chapter 3 and Chapter 4, we showed promising results with intensity features. In Chapter 3, we attempted to calibrate the intensity features, but faced issues due to the manufacturer's pre-processing. Further work is needed to calibrate the intensity, with the aim of retrieving the reflectance of objects. Moreover, large-scale point cloud acquisitions involve several TLS devices from different manufacturers, as well as ALS devices. As seen in this thesis, intensity differs according to the apparatus used, so calibrated intensity would be beneficial in these large-scale point cloud acquisition cases.

3DSES: instance segmentation "Can we apply 3DSES alignment to instance segmentation?" The method developed in Chapter 4 can be extended to produce instance segmentation labels. For instance, each object of the 3D CAD model is uniquely identified, enabling us to isolate them. Therefore, we can split the CAD models by object rather than by semantic class, as utilized in Chapter 4. Then using the methods introduced in this thesis, alignment between the CAD model and the 3D point cloud will give us "instance pseudo-labels". However, to

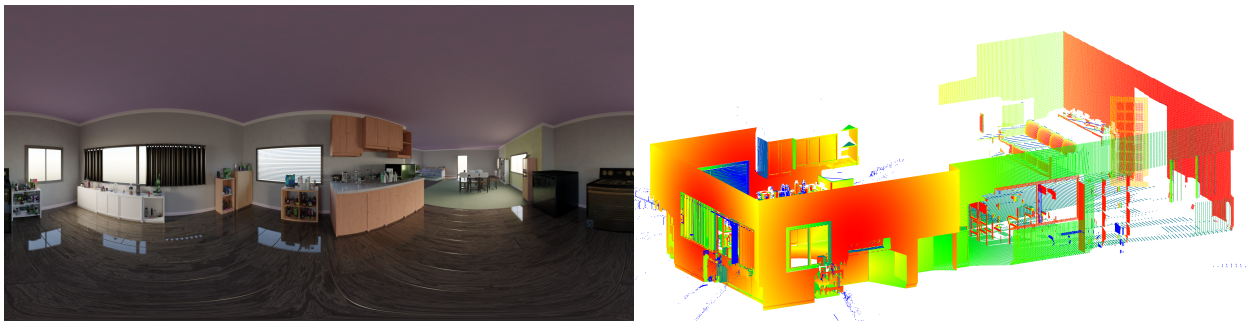
(a) Lamp class extracted from 3DSES (b) Lamp instances individually isolated

Figure 6.1: Lamps of 3DSES, isolated by connected components [4], and colorized by instance.

determine the accuracy of this alignment, we need an instance ground truth, which is not provided in 3DSES. To address this, we propose to use a Connected Components function [4] to create groups of points based on 3DSES semantic labels. Fig. 6.1 shows the result of Connected Components on the "Lamp" class of 3DSES. If a human step is added, it will enable us to obtain instance segmentation ground truth and thus compute these metrics.

Pretrained deep models "Can IPCS improve the segmentation of real indoor point clouds containing small objects?" In Chapter 5, we generated a large-scale synthetic indoor dataset comprising 3D models and 3D point clouds. To confirm the potential of IPCS in semantic segmentation of small objects, pretraining deep models on IPCS and testing them on real scenes needs is now required. Recently, PPT [105] introduced a multi-dataset pretraining approach for indoor datasets (S3DIS [120], ScanNet [121], Structured3D [251]). This pretraining approach has recently been used in PointTransformerV3 [106] to improve segmentation metrics. Sonata extends this pre-training approach to other indoor datasets. However, this pre-training is currently limited to photogrammetric datasets (real or synthetic). This approach needs to be extended to TLS acquisitions to improve the segmentation metrics of TLS point clouds. Therefore, IPCS could be used with other real TLS acquisitions (*e.g.* ScanNet++ [39], InLUT3D [134], 3DSES [1]) in order to create a multi-TLS dataset for joint learning.

Large-scale synthetic indoor dataset The methods presented in Chapter 5 enable the addition of new objects in 3D scenes by defining constraints. Therefore, existing object datasets can be used (such as Objaverse-XL [295]) or specific CAD models (such as fire extinguishers, switches, or outlets) can be added to improve the diversity of IPCS. In Chapter 5, we present 3D scenes and corresponding virtual TLS point clouds containing radiometric information, *i.e.* intensity. However, the 3D scenes contain colorized objects, and Blender enables



(a) Panoramic images from IPCS room

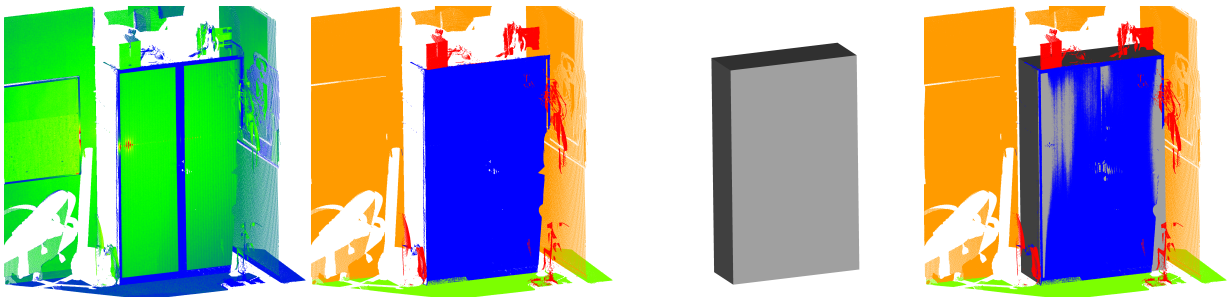
(b) Synthetic point cloud of the same room

Figure 6.2: Since IPCS contains point clouds raycast inside Blender scene, it is possible to use Cycles rendering in Blender. Using Cycles, we can render panoramic images at the scanner's position to reproduce a traditional TLS acquisition process.

2D rendering. For instance, Fig. 6.2 presents a panoramic image rendered from Cycles [293] for a specific virtual TLS position. We can envision a process that generates 2D images from each virtual TLS position of IPCS and then maps the color information onto the original point clouds. This synthetic color information would enable deep model to be trained with color information and allow color to be used during fine-tuning tasks (without requiring random initialization of weights dedicated to color). Furthermore, depending on the results of the pretraining of deep models on IPCS, it is conceivable to extend the synthetic data generation to other environments, in particular industrial ones which contains complex objects that are difficult to segment manually.

6.2.2 Long-term perspectives

Designing methods to automatically create the 3D models Of late, 3D model reconstruction methods, whether deep or not, have been limited to small-scale point clouds [296, 297, 298] or by limited input parameters [25]. The synthetic dataset presented in Chapter 5 can also be used to evaluate these methods. A primary approach to test reconstruction on 3DSES is Li et al. [299]. Li et al. use deep learning to segment point clouds and then employ a parameter estimator for primitive reconstruction. They achieve parameter optimization with the help of a cost function based on distance and IoU. A more advanced approach may rely on deep learning [297] for identifying and estimating the parameters of objects. These outputs are then used to find the CAD inside the database. For instance, Fig. 6.3 illustrates the manual placement of a cabinet CAD model on a point cloud. If this placement can be automated, it could save a lot of time when modeling large-scale point cloud acquisitions. However, an early alternative could be to propose a list of shapes to human modelers and have them position which just need to be validated.



(a) XYZ & intensity (b) XYZ & labels (c) CAD model (d) XYZ & labels & CAD

Figure 6.3: The 3DSES and IPCS datasets contain CAD models and TLS point clouds. Therefore, it is possible to use CAD shapes to reconstruct an entire building from TLS acquisitions. The figure above illustrates the desired process for a 3DSES cabinet.

As 3DSES disposes of a hidden test set, we can use it to qualify the reconstruction. The primary approach is to compute the distance between the reconstructed CAD model and the real point clouds. This method can highlight gaps directly on the point clouds, but cannot express accuracy in regions where points are missing [300]. Another approach is to use the Metro algorithm [209] to compute the distance between the reconstructed CAD model and our CAD model. Moreover, the approach detailed in Marchand et al. [300] is also practicable. This method uses real scanner positions to raycast points on a mesh and compares the distance between the closest intersection point and the real 3D point. Then, metrics defined by Marchand et al. can be used to determine the quality of the reconstruction using 3DSES as an benchmark for CAD indoor reconstruction.

Weakly supervised The methods detailed in the related work [143, 145] show promising results with fewer labels. Employing these methods could be an interesting area of research, particularly if they are linked with pseudo-labels from Chapter 4. The weakly supervised approach may provide solutions to some of the annotation difficulties of pseudo-annotation methods. While the reduction of label numbers attracts some attention in weakly supervised learning, the interesting axis of research for us is learning with noisy labels [301]. This type of learning could allow us to bypass some annotation errors due to CAD models.

Uncertainty qualification As land surveyors produce data with high accuracy, and have a lot of control during their processes, they require certainty in deep segmentation. However, it is well known that deep networks can be overconfident in their predictions [302, 303, 304], which can be detrimental for applications such as autonomous driving [305] and healthcare [306]. In 3D point clouds, a mislabeling is not as critical as in the previously mentioned applications, but can contribute to errors in 2D plans or 3D models and impact the renovation or construction of buildings. Therefore, it would be interesting to explore whether deep models

could predict high certainty for accurate labels and low certainty for mislabeled data [307]. This uncertainty could help to identify regions where deep models are uncertain about their predictions, and ultimately leave the final decision to humans in these regions. The uncertainty notion may be an interesting tool if linked with 3D reconstruction methods. We can envision an automatic reconstruction in regions where the models are certain, and rely on human modeling in regions with high uncertainty. In 3D point clouds, Bayesian models are used to estimate uncertainty in a bridge TLS dataset [308] or on factory elements [309]. Furthermore, uncertainty techniques are often employed to improve segmentation accuracy [310, 311]. Of late, Kong et al. [307] proposed a survey on uncertainty in 3D scenes and showed that most actual deep methods fail to predict uncertainty. Further work is needed to tag the XYZ coordinates of each point with a semantic label and a clear, precise certainty measure.

Bibliography

[1] M. Mérizette, N. Audebert, P. Kervella and J. Verdun, "3DSES: An indoor lidar point cloud segmentation dataset with real and pseudo-labels from a 3D model," in *Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 2: VISAPP*, INSTICC. SciTePress, 2025, pp. 707–716.

[2] D. Wujanz, M. Burger, F. Tschirschwitz, T. Nietzschmann, F. Neitzel and T. P. Kersten, "Determination of intensity-based stochastic models for terrestrial laser scanners utilising 3D-point clouds," *Sensors*, vol. 18, no. 7, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/7/2187

[3] A. X. Chang *et al.*, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[4] D. Girardeau-Montaut, "Détection de changement sur des données géométriques tridimensionnelles," Ph.D. dissertation, Télécom Paris, 2006.

[5] J. Jung, C. Stachniss, S. Ju and J. Heo, "Automated 3D volumetric reconstruction of multiple-room building interiors for as-built BIM," *Advanced Engineering Informatics*, vol. 38, pp. 811–825, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034618300600

[6] J. Wang, W. Sun, W. Shou, X. Wang, C. Wu, H.-Y. Chong, Y. Liu and C. Sun, "Integrating BIM and LiDAR for real-time construction quality control," *Journal of Intelligent & Robotic Systems*, vol. 79, pp. 417–432, 2015.

[7] W. Y. Yan, A. Shaker and N. El-Ashmawy, "Urban land cover classification using airborne lidar data: A review," *Remote Sensing of Environment*, vol. 158, pp. 295–310, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425714004374

[8] D. P. Pocobelli, J. Boehm, P. Bryan, J. Still and J. Grau-Bové, "BIM for heritage science: a review," *Heritage Science*, vol. 6, no. 1, pp. 1–15, 2018.

[9] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem," *Proceedings of the London Mathematical Society*, vol. s2-42, no. 1, pp. 230–265, 1937. [Online]. Available: https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/plms/s2-42.1.230

[10] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.

[11] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, p. 273–297, Sep. 1995. [Online]. Available: https://doi.org/10.1023/A:1022627411411

[12] Z. Gan, L. Zhong, Y. Li and H. Guan, "A random forest based method for urban object classification using lidar data and aerial imagery," in *2015 23rd International Conference on Geoinformatics*. IEEE, 2015, pp. 1–4.

[13] F. Rosenblatt, "The perceptron: A perceiving and recognizing automaton," Project PARA, Cornell Aeronautical Laboratory, Report, Jan. 1957.

[14] P. J. Werbos, *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*. USA: Wiley-Interscience, 1994.

[15] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *North American Chapter of the Association for Computational Linguistics*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:52967399

[16] A. Grattafiori *et al.*, "The Llama 3 herd of models," 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[17] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[18] A. Défossez, L. Mazaré, M. Orsini, A. Royer, P. Pérez, H. Jégou, E. Grave and N. Zeghidour, "Moshi: a speech-text foundation model for real-time dialogue," 2024. [Online]. Available: https://arxiv.org/abs/2410.00037

[19] A. Kirillov *et al.*, "Segment Anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 4015–4026.

[20] C. Schuhmann *et al.*, "LAION-5B: An open large-scale dataset for training next generation image-text models," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 25 278–25 294. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/a1859debfb3b59d094f3504d5ebb6c25-Paper-Datasets_and_Benchmarks.pdf

[21] N. Dufour, V. Kalogeiton, D. Picard and L. Landrieu, "Around the world in 80 timesteps: A generative approach to global visual geolocation," in *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, June 2025, pp. 23 016–23 026.

[22] Y. Ohta, T. Kanade and T. Sakai, "An analysis system for scenes containing objects with substructures," in *Proceedings of 4th International Joint Conference on Pattern Recognition (IJCPR '78)*, November 1978, pp. 752 – 754.

[23] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu and M. Bennamoun, "Deep Learning for 3D Point Clouds: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 12, pp. 4338–4364, Dec. 2021, number: 12 Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[24] A.-V. Vo, L. Truong-Hong, D. F. Laefer and M. Bertolotto, "Octree-based region growing for point cloud segmentation," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 104, pp. 88–100, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271615000283

[25] R. Schnabel, R. Wahl and R. Klein, "Efficient RANSAC for point-cloud shape detection," *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, 2007. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2007.01016.x

[26] F. Tarsha-Kurdi, T. Landes and P. Grussenmeyer, "Hough-Transform and Extended RANSAC Algorithms for Automatic Detection of 3D Building Roof Planes from Lidar Data," in *International Archives of Photogrammetry, Remote Sensing and Spatial Information Systems*, vol. XXXVI, Espoo, Finland, Sep. 2007, pp. 407–412. [Online]. Available: https://shs.hal.science/halshs-00264843

[27] J. Demantké, C. Mallet, N. David and B. Vallet, "Dimensionality based scale selection in 3D lidar point clouds," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXVIII-5/W12, pp. 97–102, 2011. [Online]. Available: https://isprs-archives.copernicus.org/articles/XXXVIII-5-W12/97/2011/

[28] B. Vallet, M. Brédif, A. Serna, B. Marcotegui and N. Paparoditis, "TerraMobilita/iQmulus urban point cloud analysis benchmark," *Computers & Graphics*, Jun. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S009784931500028X

[29] M. A. Günen, "Adaptive neighborhood size and effective geometric features selection for 3D scattered point cloud classification," *Applied Soft Computing*, vol. 115, p. 108196, Jan. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494621010395

[30] S. Ullman and S. Brenner, "The interpretation of structure from motion," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 203, no. 1153, pp. 405–426, 1979. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rspb.1979.0006

[31] R. C. Bolles, H. H. Baker and D. H. Marimont, "Epipolar-plane image analysis: An approach to determining structure from motion," *International Journal of Computer Vision*, vol. 1, pp. 7–55, 1987. [Online]. Available: https://api.semanticscholar.org/CorpusID:12598541

[32] J. Mure-Dubois and H. Hügli, "Fusion of Time of Flight Camera Point Clouds," in *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications - M2SFA2 2008*. Marseille, France: Andrea Cavallaro and Hamid Aghajan, Oct. 2008. [Online]. Available: https://inria.hal.science/inria-00326781

[33] A. Jelalian, *Laser Radar Systems*, ser. Artech House radar library. Artech House, 1992. [Online]. Available: https://books.google.fr/books?id=zfTxf9orjBkC

[34] F. Rottensteiner, G. Sohn, J. Jung, M. Gerke, C. Baillard, S. Bénitez and U. Breitkopf, "The ISPRS benchmark on urban object classification and 3D building reconstruction," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. I-3, Jul. 2012.

[35] I. Zolanvari, S. Ruano, A. Rana, A. Cummins, A. Smolic, R. Da Silva and M. Rahbar, "DublinCity: Annotated LiDAR Point Cloud and its Applications," in *30th British Machine Vision Conference*, Sep. 2019.

[36] C. Gaydon, M. Daab and F. Roche, "FRACTAL: An Ultra-Large-Scale Aerial Lidar Dataset for 3D Semantic Segmentation of Diverse Landscapes," May 2024. [Online]. Available: http://arxiv.org/abs/2405.04634

[37] X. Roynard, J.-E. Deschaud and F. Goulette, "Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification," *International Journal of Robotics Research*, May 2018. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364918767506

[38] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, J. Gall and C. Stachniss, "Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences," *International Journal of Robotics Research*, Apr. 2021.

[39] C. Yeshwanth, Y.-C. Liu, M. Nießner and A. Dai, "ScanNet++: A high-fidelity dataset of 3D indoor scenes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 12–22.

[40] D. D. Lichti and B. Harvey, "The effects of reflecting surface material properties on time-of-flight laser scanner measurements," 2002. [Online]. Available: https://api.semanticscholar.org/CorpusID:9926219

[41] C. Fröhlich and M. Mettenleiter, "Terrestrial laser scanning-new perspectives in 3D surveying," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, 01 2004.

[42] N. Pfeifer, P. Dorninger, A. Haring and H. Fan, "Investigating terrestrial laser scanning intensity data: quality and functional relations," *8th Conference on Optical 3-D Measurement Techniques*, 01 2007.

[43] N. Sanchiz-Viel, E. Bretagne, E. M. Mouaddib and P. Dassonvalle, "Radiometric Correction of Laser Scanning Intensity Data Applied For Terrestrial Laser Scanning." *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 172, pp. 1–16, Feb. 2021. [Online]. Available: https://hal.science/hal-03042545

[44] A. G. Kashani, M. J. Olsen, C. E. Parrish and N. Wilson, "A review of LIDAR radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration," *Sensors*, vol. 15, no. 11, pp. 28 099–28 128, 2015. [Online]. Available: https://www.mdpi.com/1424-8220/15/11/28099

[45] B. Schmitz, C. Holst, T. Medic, D. D. Lichti and H. Kuhlmann, "How to efficiently determine the range precision of 3D terrestrial laser scanners," *Sensors*, vol. 19, no. 6, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/6/1466

[46] X. Liu, Q. Li, Y. Xu and X. Wei, "Point cloud intensity correction for 2D LiDAR mobile laser scanning," *Wireless Communications and Mobile Computing*, vol. 2022, 01 2022.

[47] G. Biavati, G. Donfrancesco, F. Cairo and D. Feist, "Correction scheme for close-range lidar returns," *Applied Optics*, vol. 50, pp. 5872–5882, 10 2011.

[48] S. Kaasalainen, A. Jaakkola, M. Kaasalainen, A. Krooks and A. Kukko, "Analysis of incidence angle and distance effects on terrestrial laser scanner intensity: Search for correction methods," *Remote Sensing*, vol. 3, no. 10, pp. 2207–2221, 2011. [Online]. Available: https://www.mdpi.com/2072-4292/3/10/2207

[49] K. Tan and X. Cheng, "Intensity data correction based on incidence angle and distance for terrestrial laser scanner," *Journal of Applied Remote Sensing*, vol. 9, no. 1, p. 094094, 2015. [Online]. Available: https://doi.org/10.1117/1.JRS.9.094094

[50] W. Fang, X. Huang, F. Zhang and D. Li, "Intensity correction of terrestrial laser scanning data by estimating laser transmission function," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 2, pp. 942–951, 2015.

[51] T.-A. Teo and H.-L. Yu, "Empirical radiometric normalization of road points from terrestrial mobile lidar system," *Remote Sensing*, vol. 7, no. 5, pp. 6336–6357, 2015. [Online]. Available: https://www.mdpi.com/2072-4292/7/5/6336

[52] E. Bretagne, P. Dassonvalle and G. Caron, "Spherical target-based calibration of terrestrial laser scanner intensity. application to colour information computation," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 144, pp. 14–27, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271618301771

[53] M. A. Balduzzi, D. Van der Zande, J. Stuckens, W. W. Verstraeten and P. Coppin, "The properties of terrestrial laser system intensity for measuring leaf geometries: A case study with conference pear trees (pyrus communis)," *Sensors*, vol. 11, no. 2, pp. 1657–1681, 2011. [Online]. Available: https://www.mdpi.com/1424-8220/11/2/1657

[54] K. Anttila, S. Kaasalainen, A. Krooks, H. Kaartinen, A. Kukko, T. Manninen, P. Lahtinen and N. Siljamo, "Radiometric calibration of tls intensity: Application to snow cover change detection," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXVIII-5/W12, pp. 175–179, 09 2012.

[55] H. Yu, Z. Yang, L. Tan, Y. Wang, W. Sun, M. Sun and Y. Tang, "Methods and datasets on semantic segmentation: A review," *Neurocomputing*, vol. 304, pp. 82–103, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231218304077

[56] T. Landes, P. Grussenmeyer and H. Boulaassal, "Les principes fondamentaux de la lasergrammétrie terrestre : acquisition, traitement des données et applications," *XYZ : revue de l'Association française de topographie*, no. 129, pp. 25–38, Sep 2011. [Online]. Available: http://publis.icube.unistra.fr/3-LGB11

[57] Q. Zhan, L. Yu and Y. Liang, "A point cloud segmentation method based on vector estimation and color clustering," in *The 2nd International Conference on Information Science and Engineering*, 2010, pp. 3463–3466.

[58] M. A. Wani and H. R. Arabnia, "Parallel edge-region-based segmentation algorithm targeted at reconfigurable multiring network," *J. Supercomput.*, vol. 25, no. 1, p. 43–62, May 2003. [Online]. Available: https://doi.org/10.1023/A:1022804606389

[59] A. Golovinskiy and T. Funkhouser, "Min-cut based segmentation of point clouds," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, 2009, pp. 39–46.

[60] H. Macher, T. Landes and P. Grussenmeyer, "Point clouds segmentation as base for as-built bim creation," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. II-5/W3, pp. 191–197, 2015. [Online]. Available: https://isprs-annals.copernicus.org/articles/II-5-W3/191/2015/

[61] M. Hearst, S. Dumais, E. Osuna, J. Platt and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[62] J. Zhang, X. Lin and X. Ning, "SVM-based classification of segmented airborne LiDAR point clouds in urban areas," *Remote Sensing*, vol. 5, no. 8, pp. 3749–3775, 2013. [Online]. Available: https://www.mdpi.com/2072-4292/5/8/3749

[63] J. Niemeyer, F. Rottensteiner and U. Soergel, "Conditional random fields for lidar point cloud classification in complex urban areas," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. I-3, pp. 263–268, 2012. [Online]. Available: https://isprs-annals.copernicus.org/articles/I-3/263/2012/

[64] D. Munoz, J. A. Bagnell, N. Vandapel and M. Hebert, "Contextual classification with functional max-margin markov networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009, pp. 975–982.

[65] J. Zhang, X. Zhao, Z. Chen and Z. Lu, "A review of deep learning-based semantic segmentation for point cloud," *IEEE Access*, vol. 7, pp. 179 118–179 133, 2019.

[66] A. Xiao, X. Zhang, L. Shao and S. Lu, "A Survey of Label-Efficient Deep Learning for 3D Point Clouds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, pp. 9139–9160, Dec. 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10561540

[67] C. R. Qi, H. Su, K. Mo and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[68] Y. Li, R. Bu, M. Sun, W. Wu, X. Di and B. Chen, "PointCNN: convolution on χ-transformed points," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 828–838.

[69] W. Wu, Z. Qi and L. Fuxin, "PointConv: Deep Convolutional Networks on 3D Point Clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[70] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette and L. J. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[71] A. Boulch, "ConvPoint: Continuous convolutions for point cloud processing," *Computers & Graphics*, vol. 88, pp. 24 – 34, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0097849320300224

[72] H. Thomas, Y.-H. H. Tsai, T. D. Barfoot and J. Zhang, "KPConvX: Modernizing kernel point convolution with kernel attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 5525–5535.

[73] D. Halperin and N. Eisl, "Point Cloud Based Scene Segmentation: A Survey," 2025. [Online]. Available: https://arxiv.org/abs/2503.12595

[74] H. Su, S. Maji, E. Kalogerakis and E. G. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proc. ICCV*, 2015.

[75] X. Chen, H. Ma, J. Wan, B. Li and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[76] A. Boulch, J. Guerry, B. Le Saux and N. Audebert, "SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks," *Computers & Graphics*, vol. 71, pp. 189–198, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849317301942

[77] D. Robert, B. Vallet and L. Landrieu, "Learning multi-view aggregation in the wild for large-scale 3D semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 5575–5584.

[78] M. Jaritz, J. Gu and H. Su, "Multi-view PointNet for 3D Scene Understanding," in *ICCV Workshop 2019*, 2019.

[79] K. Abou Zeid, K. Yilmaz, D. de Geus, A. Hermans, D. Adrian, T. Linder and B. Leibe, "DINO in the Room: Leveraging 2D Foundation Models for 3D Segmentation," *arXiv preprint arXiv:2503.18944*, 2025.

[80] Y. Yang, X. Wu, T. He, H. Zhao and X. Liu, "SAM3D: Segment Anything in 3D Scenes," Jun. 2023, arXiv:2306.03908 [cs]. [Online]. Available: http://arxiv.org/abs/2306.03908

[81] H. Tai, Q. He, J. Zhang, Y. Qian, Z. Zhang, X. Hu, Y. Wang and Y. Liu, "Open-Vocabulary SAM3D: Understand Any 3D Scene," Jun. 2024, arXiv:2405.15580 [cs]. [Online]. Available: http://arxiv.org/abs/2405.15580

[82] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE Press, 2015, p. 922–928. [Online]. Available: https://doi.org/10.1109/IROS.2015.7353481

[83] G. Riegler, A. Osman Ulusoy and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[84] P.-S. Wang, Y. Liu, Y.-X. Guo, C. Sun and X. Tong, "O-CNN: Octree-based convolutional neural networks for 3d shape analysis," *ACM Transactions on Graphics*, vol. 36, pp. 1–11, 07 2017.

[85] P.-S. Wang, C.-Y. Sun, Y. Liu and X. Tong, "Adaptive O-CNN: a patch-based deep representation of 3D shapes," *ACM Transactions on Graphics*, vol. 37, no. 6, Dec. 2018. [Online]. Available: https://doi.org/10.1145/3272127.3275050

[86] C. Choy, J. Gwak and S. Savarese, "4D spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[87] T. Le and Y. Duan, "PointGrid: A deep network for 3D shape understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[88] M. Ye, R. Wan, S. Xu, T. Cao and Q. Chen, "DRINet++: Efficient voxel-as-point point cloud segmentation," *ArXiv*, vol. abs/2111.08318, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:244130129

[89] C. Zhang, H. Wan, X. Shen and Z. Wu, "PVT: Point-voxel transformer for point cloud learning," *International Journal of Intelligent Systems*, vol. 37, no. 12, pp. 11 985–12 008, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/int.23073

[90] P.-S. Wang, "OctFormer: Octree-based transformers for 3D point clouds," *ACM Transactions on Graphics (SIGGRAPH)*, vol. 42, no. 4, 2023.

[91] J. Bruna, W. Zaremba, A. Szlam and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

[92] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[93] G. Te, W. Hu, A. Zheng and Z. Guo, "RGCNN: Regularized Graph CNN for Point Cloud Segmentation," in *Proceedings of the 26th ACM International Conference on Multimedia*, ser. MM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 746–754. [Online]. Available: https://doi.org/10.1145/3240508.3240621

[94] Y. Zhang and M. Rabbat, "A Graph-CNN for 3D Point Cloud Classification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE Press, 2018, p. 6279–6283. [Online]. Available: https://doi.org/10.1109/ICASSP.2018.8462291

[95] G. Te, W. Hu, Z. Guo and A. Zheng, "RGCNN: Regularized graph cnn for point cloud segmentation," *Proceedings of the 26th ACM international conference on Multimedia*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:47015108

[96] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein and J. M. Solomon, "Dynamic Graph CNN for Learning on Point Clouds," *ACM Transactions on Graphics (TOG)*, 2019.

[97] N. Zhang, Z. Pan, T. H. Li, W. Gao and G. Li, "Improving graph representation for point cloud segmentation via attentive filtering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 1244–1254.

[98] D. Robert, H. Raguet and L. Landrieu, " Scalable 3D Panoptic Segmentation As Superpoint Graph Clustering ," in *2024 International Conference on 3D Vision (3DV)*. IEEE Computer Society, Mar. 2024, pp. 179–189. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/3DV62453.2024.00135

[99] D. Robert, "Efficient learning on large-scale 3D point clouds," Theses, Université Gustave Eiffel, Jan. 2024. [Online]. Available: https://theses.hal.science/tel-04615714

[100] G. Qian, Y. Li, H. Peng, J. Mai, H. Hammoud, M. Elhoseiny and B. Ghanem, "PointNeXt: Revisiting PointNet++ with improved training and scaling strategies," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 23 192–23 204. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/9318763d049edf9a1f2779b2a59911d3-Paper-Conference.pdf

[101] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni and A. Markham, "RandLA-Net: efficient semantic segmentation of large-scale point clouds," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[102] H. Zhao, L. Jiang, J. Jia, P. H. Torr and V. Koltun, "Point Transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 16 259–16 268.

[103] X. Lai, J. Liu, L. Jiang, L. Wang, H. Zhao, S. Liu, X. Qi and J. Jia, "Stratified Transformer for 3D point cloud segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 8500–8509.

[104] X. Deng, W. Zhang, Q. Ding and X. Zhang, "Pointvector: A vector representation in point cloud analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 9455–9465.

[105] X. Wu, Z. Tian, X. Wen, B. Peng, X. Liu, K. Yu and H. Zhao, "Towards large-scale 3D representation learning with multi-dataset point prompt training," in *CVPR*, 2024.

[106] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He and H. Zhao, "Point Transformer V3: Simpler, faster, stronger," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 4840–4851.

[107] D. Robert, H. Raguet and L. Landrieu, "Efficient 3D Semantic Segmentation with Superpoint Transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 17 195–17 204.

[108] L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[109] Z. Liang, Z. Li, S. Xu, M. Tan and K. Jia, "Instance segmentation in 3D scenes using semantic superpoint tree networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 2783–2792.

[110] J. Sun, C. Qing, J. Tan and X. Xu, "Superpoint Transformer for 3D Scene Instance Segmentation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 2, pp. 2393–2401, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/25335

[111] P. Arbelaez, "Boundary extraction in natural images using ultrametric contour maps," *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, pp. 182–182, 2006.

[112] Y. Xu, T. Géraud and L. Najman, "Hierarchical image simplification and segmentation based on mumford-shah-salient level line selection," *Pattern Recognition Letters*, vol. 83, 03 2016.

[113] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[114] A. Dai and M. Nießner, "3DMV: Joint 3D-Multi-View Prediction for 3D Semantic Scene Segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[115] J. Xu, R. Zhang, J. Dou, Y. Zhu, J. Sun and S. Pu, "Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 16 024–16 033.

[116] S. Srivastava and G. Sharma, "Omnivec: Learning robust representations with cross modal sharing," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2024, pp. 1236–1248.

[117] M. A. Uy, Q.-H. Pham, B.-S. Hua, D. T. Nguyen and S.-K. Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *International Conference on Computer Vision (ICCV)*, 2019.

[118] T. Wu *et al.*, "OmniObject3D: Large-vocabulary 3D object dataset for realistic perception, reconstruction and generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 803–814.

[119] D. Griffiths and J. Boehm, "SynthCity: A large scale synthetic point cloud," Jul. 2019, arXiv:1907.04758 [cs]. [Online]. Available: http://arxiv.org/abs/1907.04758

[120] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer and S. Savarese, "3D semantic parsing of large-scale indoor spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[121] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser and M. Niessner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[122] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010. [Online]. Available: http://link.springer.com/10.1007/s11263-009-0275-4

[123] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.

[124] A. Kuznetsova *et al.*, "The Open Images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *IJCV*, 2020.

[125] A. Gupta, P. Dollar and R. Girshick, "LVIS: A dataset for large vocabulary instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[126] T. Ridnik, E. Ben-Baruch, A. Noy and L. Zelnik, "ImageNet-21K Pretraining for the Masses," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, J. Vanschoren and S. Yeung, Eds., vol. 1, 2021. [Online]. Available: https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/98f13708210194c475687be6106a3b84-Paper-round1.pdf

[127] P. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, pp. 167–181, 09 2004.

[128] J. D. Stets, Y. Sun, W. Corning and S. W. Greenwald, "Visualization and labeling of point clouds in virtual reality," in *SIGGRAPH Asia 2017 Posters*, ser. SA '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3145690.3145729

[129] P. Z. Ramirez, C. Paternesi, L. D. Luigi, L. Lella, D. D. Gregorio and L. D. Stefano, "Shooting labels: 3D semantic labeling by virtual reality," in *2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, 2020, pp. 99–106.

[130] T. Lin, Z. Yu, M. McGinity and S. Gumhold, "An immersive labeling method for large point clouds," *Computers & Graphics*, vol. 124, p. 104101, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S009784932400236X

[131] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niebner, M. Savva, S. Song, A. Zeng and Y. Zhang, "Matterport3D: Learning from RGB-D Data in Indoor Environments," in *International Conference on 3D Vision (3DV)*, Oct. 2017.

[132] D. Rozenberszki, O. Litany and A. Dai, "Language-grounded indoor 3D semantic segmentation in the wild," in *Computer Vision – ECCV 2022*, 2022, pp. 125–141.

[133] Y. Guo *et al.*, "LiDAR-Net: A Real-scanned 3D Point Cloud Dataset for Indoor Scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 21 989–21 999.

[134] J. Walczak, "InLUT3D: Challenging real indoor dataset for point cloud analysis," Jul. 2024, arXiv:2408.03338 [cs]. [Online]. Available: http://arxiv.org/abs/2408.03338

[135] E. Agapaki, A. Glyn-Davies, S. Mandoki and I. Brilakis, "CLOI: A shape classification benchmark dataset for industrial facilities," *Computing in Civil Engineering 2019*, 2019. [Online]. Available: https://www.repository.cam.ac.uk/handle/1810/289351

[136] H. Kim, C. Yeo, I. D. Lee and D. Mun, "Deep-learning-based retrieval of piping component catalogs for plant 3D CAD model reconstruction," *Computers in Industry*, vol. 123, p. 103320, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166361520305546

[137] C. Yin, B. Wang, V. Gan, M. Wang and J. Cheng, "Automated semantic segmentation of industrial point clouds using ResPointNet++," *Automation in Construction*, vol. 130, Oct. 2021.

[138] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, no. 1, pp. 44–53, 08 2017. [Online]. Available: https://doi.org/10.1093/nsr/nwx106

[139] S. Guinard and L. Landrieu, "Weakly supervised segmentation-aided classification of urban scenes from 3D lidar point clouds," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-1/W1, pp. 151–157, 2017. [Online]. Available: https://isprs-archives.copernicus.org/articles/XLII-1-W1/151/2017/

[140] X. Xu and G. H. Lee, "Weakly supervised semantic point cloud segmentation: Towards 10x fewer labels," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[141] Y. Zhang, Y. Qu, Y. Xie, Z. Li, S. Zheng and C. Li, "Perturbed self-distillation: Weakly supervised large-scale point cloud semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 15 520–15 528.

[142] Q. Hu, B. Yang, G. Fang, Y. Guo, A. Leonardis, N. Trigoni and A. Markham, "SQN: Weakly-supervised semantic segmentation of large-scale 3D point clouds," in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella and T. Hassner, Eds. Cham: Springer Nature Switzerland, 2022, pp. 600–619.

[143] Y. Zhang, Z. Li, Y. Xie, Y. Qu, C. Li and T. Mei, "Weakly supervised semantic segmentation for large-scale point cloud," 2022. [Online]. Available: https://arxiv.org/abs/2212.04744

[144] M. Li, Y. Xie, Y. Shen, B. Ke, R. Qiao, B. Ren, S. Lin and L. Ma, "HybridCR: Weakly-supervised 3D point cloud semantic segmentation via hybrid contrastive regularization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 14 930–14 939.

[145] H. Kweon, J. Kim and K.-J. Yoon, "Weakly supervised point cloud semantic segmentation via artificial oracle," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 3721–3731.

[146] X. Zhou, V. Koltun and P. Krähenbühl, "Simple multi-dataset detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 7571–7580.

[147] Y. Yang, C. Feng, Y. Shen and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[148] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[149] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou and J. Lu, "Point-BERT: Pre-training 3D point cloud transformers with masked point modeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[150] Y. Pang, W. Wang, F. E. Tay, W. Liu, Y. Tian and L. Yuan, "Masked autoencoders for point cloud self-supervised learning," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*. Springer, 2022, pp. 604–621.

[151] R. Zhang, L. Wang, Y. Qiao, P. Gao and H. Li, "Learning 3D representations from 2D pre-trained models via image-to-point masked autoencoders," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 21 769–21 780.

[152] N. Hu, H. Cheng, Y. Xie, S. Li and J. Zhu, "3D-JEPA: A joint embedding predictive architecture for 3d self-supervised representation learning," 2024. [Online]. Available: https://arxiv.org/abs/2409.15803

[153] Y. LeCun, "A Path Towards Autonomous Machine Intelligence Version," vol. 62, 2022.

[154] S. Xie, J. Gu, D. Guo, C. R. Qi, L. Guibas and O. Litany, "PointContrast: Unsupervised pre-training for 3D point cloud understanding," in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III*. Berlin, Heidelberg: Springer-Verlag, 2020, p. 574–591. [Online]. Available: https://doi.org/10.1007/978-3-030-58580-8_34

[155] C. Choy, J. Park and V. Koltun, "Fully convolutional geometric features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[156] X. Wu, X. Wen, X. Liu and H. Zhao, "Masked Scene Contrast: A scalable framework for unsupervised 3D representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 9415–9424.

[157] T. Chen, S. Kornblith, M. Norouzi and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 1597–1607. [Online]. Available: https://proceedings.mlr.press/v119/chen20j.html

[158] C. Wang, L. Jiang, X. Wu, Z. Tian, B. Peng, H. Zhao and J. Jia, "Groupcontrast: Semantic-aware self-supervised representation learning for 3d understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 4917–4928.

[159] J. Hou, B. Graham, M. Niessner and S. Xie, "Exploring data-efficient 3D scene understanding with Contrastive Scene Contexts," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 15 587–15 597.

[160] X. Wu, D. DeTone, D. Frost, T. Shen, C. Xie, N. Yang, J. Engel, R. Newcombe, H. Zhao and J. Straub, "Sonata: Self-supervised learning of reliable point representations," in *CVPR*, 2025.

[161] H. Zhang, C. Wang, S. Tian, B. Lu, L. Zhang, X. Ning and X. Bai, "Deep learning-based 3D point cloud classification: A systematic survey and outlook," *Displays*, vol. 79, p. 102456, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258913453

[162] H. Ran, J. Liu and C. Wang, "Surface representation for point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 18 942–18 952.

[163] I. de Gélis, S. Lefèvre and T. Corpetti, "DC3DCD: Unsupervised learning for multiclass 3D point cloud change detection," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 206, pp. 168–183, Dec. 2023. [Online]. Available: https://hal.science/hal-04304643

[164] M. Weinmann, B. Jutzi, S. Hinz and C. Mallet, "Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 105, pp. 286–304, Jul. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271615000349

[165] T. Hackel, J. D. Wegner and K. Schindler, "Contour Detection in Unstructured 3D Point Clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[166] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *ArXiv*, Jul. 2012. [Online]. Available: https://www.semanticscholar.org/paper/Improving-neural-networks-by-preventing-of-feature-Hinton-Srivastava/0060745e006c5f14ec326904119dca19c6545e51

[167] L. Linsen and H. Prautzsch, "Local Versus Global Triangulations," in *Eurographics 2001 - Short Presentations*. Eurographics Association, 2001.

[168] I. Lee and A. Schenk, "Perceptual organization of 3D surface points," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 34, Jan. 2002.

[169] S. Filin and N. Pfeifer, "Neighborhood Systems for Airborne Laser Data," *Photogrammetric Engineering & Remote Sensing*, vol. 71, pp. 743–755, Jun. 2005.

[170] M. Weinmann, *Reconstruction and Analysis of 3D Scenes: From Irregularly Distributed 3D Points to Object Classes*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[171] A. Gressin, C. Mallet, J. Demantké and N. David, "Towards 3D lidar point cloud registration improvement using optimal neighborhood knowledge," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 79, pp. 240–251, May 2013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0924271613000622

[172] M. Pauly, R. Keiser and M. Gross, "Multi-scale Feature Extraction on Point-Sampled Surfaces," *Computer Graphics Forum*, vol. 22, no. 3, pp. 281–289, 2003, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00675. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.00675

[173] N. J. Mitra and A. Nguyen, "Estimating surface normals in noisy point cloud data," in *Proceedings of the nineteenth annual symposium on Computational geometry*, ser. SCG '03. New York, NY, USA: Association for Computing Machinery, Jun. 2003, pp. 322–328. [Online]. Available: https://doi.org/10.1145/777792.777840

[174] M. CHEN, Y. Wan, M. Wang and J. Xu, "Automatic Stem Detection in Terrestrial Laser Scanning Data With Distance-Adaptive Search Radius," *IEEE Transactions on Geoscience and Remote Sensing*, vol. PP, pp. 1–12, Jan. 2018.

[175] M. Yang and E. Lee, "Segmentation of measured point data using a parametric quadric surface approximation," *Computer-Aided Design*, vol. 31, no. 7, pp. 449–457, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010448599000421

[176] T.-J. Fan, G. Medioni and R. Nevatia, "Segmented descriptions of 3-D surfaces," *IEEE Journal on Robotics and Automation*, vol. 3, no. 6, pp. 527–538, 1987.

[177] D. Belton and D. Lichti, "Classification and segmentation of terrestrial laser scanner point clouds using local variance information," *IAPRS*, vol. 36, pp. 44–49, 01 2006.

[178] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha and M. Beetz, "Towards 3D point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008, semantic Knowledge in Robotics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889008001140

[179] M. Weinmann, B. Jutzi, C. Mallet and M. Weinmann, "Geometric features and their relevance for 3D point cloud classification," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-1/W1, pp. 157–164, 2017. [Online]. Available: https://isprs-annals.copernicus.org/articles/IV-1-W1/157/2017/

[180] E. Grilli, E. M. Farella, A. Torresani and F. Remondino, "Geometric features analysis for the classification of cultural heritage point clouds," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-2/W15, pp. 541–548, 2019. [Online]. Available: https://isprs-archives.copernicus.org/articles/XLII-2-W15/541/2019/

[181] H. Harshit, S. K. P. Kushwaha and K. Jain, "Geometric features interpretation of photogrammetric point cloud from unmanned aerial vehicle," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. X-4/W2-2022, pp. 83–88, 2022. [Online]. Available: https://isprs-annals.copernicus.org/articles/X-4-W2-2022/83/2022/

[182] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, "Surface reconstruction from unorganized points," *SIGGRAPH Comput. Graph.*, vol. 26, no. 2, p. 71–78, Jul. 1992. [Online]. Available: https://doi.org/10.1145/142920.134011

[183] A. Boulch and R. Marlet, "Fast and Robust Normal Estimation for Point Clouds with Sharp Features," *Computer Graphics Forum*, vol. 31, no. 5, pp. 1765–1774, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2012.03181.x

[184] A. Boulch and R. Marlet, "Deep learning for robust normal estimation in unstructured point clouds," *Computer Graphics Forum*, vol. 35, no. 5, pp. 281–290, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12983

[185] H. Xiu, X. Liu, W. Wang, K.-S. Kim and M. Matsuoka, "MSECNet: Accurate and robust normal estimation for 3d point clouds by multi-scale edge conditioning," in *Proceedings of the 31st ACM International Conference on Multimedia*, ser. MM '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 2535–2543. [Online]. Available: https://doi.org/10.1145/3581783.3613762

[186] Y.-Q. Yang, Y.-X. Guo, J.-Y. Xiong, Y. Liu, H. Pan, P.-S. Wang, X. Tong and B. Guo, "Swin3D: A pretrained transformer backbone for 3D indoor scene understanding," 2023.

[187] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[188] L. Wan, M. Zeiler, S. Zhang, Y. LeCun and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, p. III–1058–III–1066.

[189] G. Larsson, M. Maire and G. Shakhnarovich, "FractalNet: Ultra-deep neural networks without residuals." *CoRR*, vol. abs/1605.07648, 2016. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1605.html#LarssonMS16a

[190] Z. Zhong, L. Zheng, G. Kang, S. Li and Y. Yang, "Random erasing data augmentation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 08 2017.

[191] T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *ArXiv*, vol. abs/1708.04552, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:23714201

[192] G. Ghiasi, T.-Y. Lin and Q. V. Le, "DropBlock: a regularization method for convolutional networks," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 10750–10760.

[193] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[194] J. Johnson, N. Ravi, J. Reizenstein, D. Novotny, S. Tulsiani, C. Lassner and S. Branson, "Accelerating 3D deep learning with PyTorch3D," in *SIGGRAPH Asia 2020 Courses*, ser. SA '20. New York, NY, USA: Association for Computing Machinery, Dec. 2020, p. 1. [Online]. Available: https://doi.org/10.1145/3415263.3419160

[195] A. Bradley, H. Li, R. Lark and S. Dunn, "BIM for infrastructure: An overall review and constructor perspective," *Automation in Construction*, vol. 71, pp. 139–152, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092658051630173X

[196] M. G. Angelini, V. Baiocchi, D. Costantino and F. Garzia, "Scan to BIM for 3D reconstruction of the papal basilica of Saint Francis in Assissi in Italy," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-5/W1, 2017. [Online]. Available: https://isprs-archives.copernicus.org/articles/XLII-5-W1/47/2017/

[197] D. Munoz, J. A. Bagnell, N. Vandapel and M. Hebert, "Contextual classification with functional Max-Margin Markov Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2009.

[198] A. Serna, B. Marcotegui, F. Goulette and J.-E. Deschaud, "Paris-rue-Madame Database - A 3D Mobile Laser Scanner Dataset for Benchmarking Urban Detection, Segmentation and Classification Methods:," in *3rd International Conference on Pattern Recognition Applications and Methods*, 2014. [Online]. Available: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0004934808190824

[199] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler and M. Pollefeys, "SEMANTIC3D.NET: A NEW LARGE-SCALE POINT CLOUD CLASSIFICATION BENCHMARK," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-1/W1, pp. 91–98, May 2017. [Online]. Available: https://isprs-annals.copernicus.org/articles/IV-1-W1/91/2017/

[200] W. Tan, N. Qin, L. Ma, Y. Li, J. Du, G. Cai, K. Yang and J. Li, "Toronto-3D: A Large-scale Mobile LiDAR Dataset for Semantic Segmentation of Urban Roadways," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[201] K. Khoshelham, L. Díaz Vilariño, M. Peter, Z. Kang and D. Acharya, "THE ISPRS BENCHMARK ON INDOOR MODELLING," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-2/W7, pp. 367–372, Sep. 2017. [Online]. Available: https://isprs-archives.copernicus.org/articles/XLII-2-W7/367/2017/

[202] N. Abreu, R. Souza, A. Pinto, A. Matos and M. Pires, "Labelled Indoor Point Cloud Dataset for BIM Related Applications," *Data*, vol. 8, no. 6, p. 101, Jun. 2023. [Online]. Available: https://www.mdpi.com/2306-5729/8/6/101

[203] Z. Ye, Y. Xu, R. Huang, X. Tong, X. Li, X. Liu, K. Luan, L. Hoegner and U. Stilla, "LASDU: A Large-Scale Aerial LiDAR Dataset for Semantic Labeling in Dense Urban Areas," *ISPRS International Journal of Geo-Information*, vol. 9, no. 7, p. 450, Jul. 2020. [Online]. Available: https://www.mdpi.com/2220-9964/9/7/450

[204] N. Varney, V. K. Asari and Q. Graehling, "DALES: A Large-scale Aerial LiDAR Data Set for Semantic Segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9150622/

[205] X. Li, C. Li, Z. Tong, A. Lim, J. Yuan, Y. Wu, J. Tang and R. Huang, "Campus3d: A photogrammetry point cloud benchmark for hierarchical understanding of outdoor scene." in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020.

[206] M. Kölle, D. Laupheimer, S. Schmohl, N. Haala, F. Rottensteiner, J. D. Wegner and H. Ledoux, "The Hessigheim 3D (H3D) benchmark on semantic segmentation of high-resolution 3D point clouds and textured meshes from UAV LiDAR and Multi-View-Stereo," *ISPRS Open Journal of Photogrammetry and Remote Sensing*, vol. 1, p. 100001, Oct. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667393221000016

[207] Q. Hu, B. Yang, S. Khalid, W. Xiao, N. Trigoni and A. Markham, "Towards Semantic Segmentation of Urban-Scale 3D Point Clouds: A Dataset, Benchmarks and Challenges," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 4977–4987.

[208] Autodesk, "Revit." [Online]. Available: https://www.autodesk.com/fr/products/revit/overview

[209] P. Cignoni, C. Rocchini and R. Scopigno, "Metro: Measuring Error on Simplified Surfaces," *Computer Graphics Forum*, vol. 17, no. 2, pp. 167–174, 1998. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.00236

[210] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun and X. Tong, "O-CNN: octree-based convolutional neural networks for 3D shape analysis," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 72:1–72:11, Jul. 2017. [Online]. Available: https://doi.org/10.1145/3072959.3073608

[211] X. Wu, Y. Lao, L. Jiang, X. Liu and H. Zhao, "Point Transformer V2: Grouped Vector Attention and Partition-based Pooling," *Advances in Neural Information Processing Systems*, vol. 35, pp. 33 330–33 342, Dec. 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/d78ece6613953f46501b958b7bb4582f-Abstract-Conference.html

[212] S. Reitmann, L. Neumann and B. Jung, "BLAINDER—A Blender AI Add-On for Generation of Semantically Labeled Depth-Sensing Data," *Sensors*, vol. 21, no. 6, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/2144

[213] S. Lytkin, V. Badenko, A. Fedotov, K. Vinogradov, A. Chervak, Y. Milanov and D. Zotov, "Saint Petersburg 3D: Creating a Large-Scale Hybrid Mobile LiDAR Point Cloud Dataset for Geospatial Applications," *Remote Sensing*, vol. 15, no. 11, p. 2735, Jan. 2023, number: 11 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2072-4292/15/11/2735

[214] M. Jaderberg, K. Simonyan, A. Vedaldi and A. Zisserman, "Synthetic data and artificial neural networks for natural scene text recognition," *ArXiv*, vol. abs/1406.2227, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:11072772

[215] A. Gupta, A. Vedaldi and A. Zisserman, "Synthetic data for text localisation in natural images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[216] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon and S. Birchfield, " Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization ," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE Computer Society, Jun. 2018, pp. 1082–10 828. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPRW.2018.00143

[217] T. Scheck, R. Seidel and G. Hirtz, "Learning from THEODORE: A synthetic omnidirectional top-view indoor dataset for deep transfer learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[218] G. Ros, L. Sellart, J. Materzynska, D. Vazquez and A. M. Lopez, "The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[219] M. Roberts, J. Ramapuram, A. Ranjan, A. Kumar, M. A. Bautista, N. Paczan, R. Webb and J. M. Susskind, "Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 10 912–10 922.

[220] J. Montalvo, P. Carballeira and A. Garcia-Martin, "Synthmanticlidar: A synthetic dataset for semantic segmentation on lidar imaging," in *2024 IEEE International Conference on Image Processing (ICIP)*, 2024, pp. 137–143.

[221] A. Raistrick *et al.*, "Infinigen Indoors: Photorealistic indoor scenes using procedural generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2024, pp. 21 783–21 794.

[222] J. Barron, D. Fleet and S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision*, vol. 12, pp. 43–77, 02 1994.

[223] W. Freeman and E. Pasztor, "Learning low-level vision," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1182–1189 vol.2.

[224] B. Lohani and R. Mishra, "Generating LiDAR data in laboratory: LiDAR simulator," *International Archives of the Photogrammetry, Remote Sensing*, vol. 52, 01 2007.

[225] A. Kukko and J. Hyyppä, "Small-footprint laser scanning simulator for system validation, error assessment, and algorithm development," *Photogrammetric Engineering and Remote Sensing*, vol. 75, p. 1177, 10 2009.

[226] L. Winiwarter, A. M. Esmorís Pena, H. Weiser, K. Anders, J. Martínez Sánchez, M. Searle and B. Höfle, "Virtual laser scanning with HELIOS++: A novel take on ray tracing-based simulation of topographic full-waveform 3D laser scanning," *Remote Sensing of Environment*, vol. 269, 2022.

[227] R. A. Hodge, "Using simulated terrestrial laser scanning to analyse errors in high-resolution scan data of irregular surfaces," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 65, no. 2, pp. 227–240, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092427161000002X

[228] S.-J. Kim, S.-H. Min, G. Kim, I. Lee and C. Jun, "Data simulation of an airborne lidar system," in *Defense + Commercial Sensing*, 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:120213648

[229] Y. Wang, D. Xie, G. Yan, W. Zhang and X. Mu, "Analysis on the inversion accuracy of LAI based on simulated point clouds of terrestrial LiDAR of tree by ray tracing algorithm," *International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 532–535, 07 2013.

[230] M. Gschwandtner, R. Kwitt, A. Uhl and W. Pree, "BlenSor: Blender Sensor Simulation Toolbox," in *Advances in Visual Computing*, G. Bebis *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 199–208.

[231] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[232] S. Bechtold and B. Höfle, "HELIOS: A multi-purpose lidar simulation framework for research, planning and training of laser scanning operations with airborne, ground-based mobile and stationary platforms," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. III-3, pp. 161–168, 2016. [Online]. Available: https://isprs-annals.copernicus.org/articles/III-3/161/2016/

[233] B. Wu, A. Wan, X. Yue and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3D LiDAR point cloud," *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[234] X. Yue, B. Wu, S. A. Seshia, K. Keutzer and A. L. Sangiovanni-Vincentelli, "A LiDAR point cloud generator: from a virtual world to autonomous driving," in *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, ser. ICMR '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 458–464. [Online]. Available: https://doi.org/10.1145/3206025.3206080

[235] D. Popovas, M. Chizhova, D. Gorkovchuk, J. Gorkovchuk, M. Hess and T. Luhmann, "Virtual terrestrial laser scanner simulator in digital twin environment," in *Proceedings of the joint international event 9th ARQUEOLÓGICA 2.0 & 3rd GEORES ('GEOmatics and pREServation), Valencia (Spain). 26–28 April 2021*. Valencia, Spain: Universitat Polytecnica de Valencia, 2021, pp. 85–92. [Online]. Available: http://ocs.editorial.upv.es/index.php/arqueologica20/arqueologica9/paper/view/12091

[236] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke and K. Goldberg, Eds., vol. 78. PMLR, Nov. 2017, pp. 1–16. [Online]. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html

[237] M. Neumann, D. Borrmann and A. Nüchter, "Semantic classification in uncolored 3d point clouds using multiscale features," in *Intelligent Autonomous Systems 17*, I. Petrovic, E. Menegatti and I. Marković, Eds. Cham: Springer Nature Switzerland, 2023, pp. 342–359.

[238] K. Korus, T. Czerniawski and M. Salamak, "Visual programming simulator for producing realistic labeled point clouds from digital infrastructure models," *Automation in Construction*, vol. 156, p. 105126, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926580523003862

[239] E. Frías, J. Pinto, R. Sousa, H. Lorenzo and L. Díaz-Vilariño, "Exploiting BIM objects for synthetic data generation toward indoor point cloud classification using deep learning," *Journal of Computing in Civil Engineering*, vol. 36, no. 6, p. 04022032, 2022. [Online]. Available: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29CP.1943-5487.0001039

[240] J. Ma, T. Czerniawski and F. Leite, "Semantic segmentation of point clouds of building interiors with deep learning: Augmenting training datasets with synthetic BIM-based point clouds," *Automation in Construction*, vol. 113, p. 103144, 05 2020.

[241] C. Morbidoni, R. Pierdicca, M. Paolanti, R. Quattrini and R. Mammoli, "Learning from synthetic point cloud data for historical buildings semantic segmentation," *Journal on Computing and Cultural Heritage*, vol. 13, 12 2020.

[242] P. Achlioptas, O. Diamanti, I. Mitliagkas and L. J. Guibas, "Learning representations and generative models for 3D point clouds," in *International Conference on Machine Learning*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:23102425

[243] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie and B. Hariharan, "Pointflow: 3d point cloud generation with continuous normalizing flows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[244] M. Zamorski, M. Zięba, P. Klukowski, R. Nowak, K. Kurach, W. Stokowiec and T. Trzciński, "Adversarial autoencoders for compact representations of 3d point clouds," *Computer Vision and Image Understanding*, vol. 193, p. 102921, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S107731422030014X

[245] J. Zhang, F. Zhang, S. Kuang and L. Zhang, "NeRF-LiDAR: Generating realistic LiDAR point clouds with neural radiance fields," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2024.

[246] Z. Zheng, F. Lu, W. Xue, G. Chen and C. Jiang, "LiDAR4D: Dynamic neural fields for novel space-time view LiDAR synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 5145–5154.

[247] J. Lee, S. Lee, C. Jo, W. Im, J. Seon and S.-E. Yoon, "SemCity: Semantic scene generation with triplane diffusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 28 337–28 347.

[248] A. Handa, V. Patraucean, S. Stent and R. Cipolla, "SceneNet: An annotated model generator for indoor scene understanding," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5737–5743, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:15886458

[249] W. Li, S. Saeedi, J. McCormac, R. Clark, D. Tzoumanikas, Q. Ye, Y. Huang, R. Tang and S. Leutenegger, "InteriorNet: Mega-scale multi-sensor photo-realistic indoor scenes dataset," in *British Machine Vision Conference (BMVC)*, 2018.

[250] J. Straub *et al.*, "The replica dataset: A digital replica of indoor spaces," 2019. [Online]. Available: https://arxiv.org/abs/1906.05797

[251] J. Zheng, J. Zhang, J. Li, R. Tang, S. Gao and Z. Zhou, "Structured3D: A Large Photo-Realistic Dataset for Structured 3D Modeling," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, vol. 12354, pp. 519–535, series Title: Lecture Notes in Computer Science. [Online]. Available: https://link.springer.com/10.1007/978-3-030-58545-7_30

[252] Z. Li *et al.*, "OpenRooms: An open framework for photorealistic indoor scene datasets," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 7190–7199.

[253] H. Fu *et al.*, "3D-FRONT: 3D furnished rooms with layouts and semantics," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, 2021, pp. 10 933–10 942.

[254] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, K. Ehsani, J. Salvador, W. Han, E. Kolve, A. Kembhavi and R. Mottaghi, "ProcTHOR: Large-Scale Embodied AI Using Procedural Generation," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 5982–5994. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/27c546ab1e4f1d7d638e6a8dfbad9a07-Paper-Conference.pdf

[255] M. Khanna, Y. Mao, H. Jiang, S. Haresh, B. Shacklett, D. Batra, A. Clegg, E. Undersander, A. X. Chang and M. Savva, "Habitat synthetic scenes dataset (hssd-200): An analysis of 3d scene scale and realism tradeoffs for objectgoal navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 16 384–16 393.

[256] A. Avetisyan *et al.*, "SceneScript: Reconstructing scenes with an autoregressive structured language model," in *Computer Vision – ECCV 2024*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler and G. Varol, Eds. Cham: Springer Nature Switzerland, 2025, pp. 247–263.

[257] C. Fang, H. Li, Y. Liang, J. Zheng, Y. Mao, Y. Liu, R. Tang, Z. Zhou and P. Tan, "Spatialgen: Layout-guided 3d indoor scene generation," *arXiv preprint*, 2025.

[258] U. Raman Kumar, A. R. Fayjie, J. Hannaert and P. Vandewalle, "BelHouse3D: A Benchmark Dataset for Assessing Occlusion Robustness in 3D Point Cloud Semantic Segmentation," in *Computer Vision – ECCV 2024 Workshops*, A. Del Bue, C. Canton, J. Pont-Tuset and T. Tommasi, Eds. Cham: Springer Nature Switzerland, 2025, pp. 308–327.

[259] B. Wen, H. Xie, Z. Chen, F. Hong and Z. Liu, "3D scene generation: A survey," 2025. [Online]. Available: https://arxiv.org/abs/2505.05474

[260] D. Ritchie, K. Wang and Y.-A. Lin, "Fast and flexible indoor scene synthesis via deep convolutional generative models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[261] K. Leimer, P. Guerrero, T. Weiss and P. Musialski, "LayoutEnhancer: Generating good indoor layouts from imperfect data," in *SIGGRAPH Asia 2022 Conference Papers*, ser. SA '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3550469.3555425

[262] D. Paschalidou, A. Kar, M. Shugrina, K. Kreis, A. Geiger and S. Fidler, "ATISS: Autoregressive transformers for indoor scene synthesis," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[263] X. Wang, C. Yeshwanth and M. Nießner, "SceneFormer: Indoor scene generation with transformers," in *2021 International Conference on 3D Vision (3DV)*, 2021, pp. 106–115.

[264] A. Bokhovkin, Q. Meng, S. Tulsiani and A. Dai, "SceneFactor: Factored latent 3D diffusion for controllable 3D scene generation," in *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, June 2025, pp. 628–639.

[265] K. Wang, Y.-A. Lin, B. Weissmann, M. Savva, A. X. Chang and D. Ritchie, "PlanIT: planning and instantiating indoor scenes with relation graph and spatial prior networks," *ACM Trans. Graph.*, vol. 38, no. 4, Jul. 2019. [Online]. Available: https://doi.org/10.1145/3306346.3322941

[266] W. Para, P. Guerrero, T. Kelly, L. J. Guibas and P. Wonka, "Generative layout modeling using constraint graphs," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 6690–6700.

[267] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[268] L. Gao, J.-M. Sun, K. Mo, Y.-K. Lai, L. J. Guibas and J. Yang, "SceneHGN: Hierarchical graph networks for 3D indoor scene generation with fine-grained geometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[269] Z. Chen, G. Wang and Z. Liu, "SceneDreamer: Unbounded 3D scene generation from 2D image collections," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 45, no. 12, pp. 15 562–15 576, dec 2023.

[270] R. Po and G. Wetzstein, "Compositional 3D scene generation using locally conditioned diffusion," in *2024 International Conference on 3D Vision (3DV)*, 2024, pp. 651–663.

[271] X. Yang, Y. Man, J.-K. Chen and Y.-X. Wang, "SceneCraft: Layout-guided 3D scene generation," in *Advances in Neural Information Processing Systems*, 2024.

[272] X. Li, Z. Lai, L. Xu, Y. Qu, L. Cao, S. Zhang, B. Dai and R. Ji, "Director3D: Real-world camera trajectory and 3d scene generation from text," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. [Online]. Available: https://openreview.net/forum?id=08A6X7FSTs

[273] A. R. Kosiorek, H. Strathmann, D. Zoran, P. Moreno, R. Schneider, S. Mokra and D. J. Rezende, "NeRF-VAE: A geometry aware 3D scene generative model," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 5742–5752. [Online]. Available: https://proceedings.mlr.press/v139/kosiorek21a.html

[274] Y. Liang, X. Yang, J. Lin, H. Li, X. Xu and Y. Chen, "LucidDreamer: Towards high-fidelity text-to-3D generation via interval score matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 6517–6526.

[275] H.-X. Yu *et al.*, "WonderJourney: Going from anywhere to everywhere," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 6658–6667.

[276] H. Yu, C. Wang, P. Zhuang, W. Menapace, A. Siarohin, J. Cao, L. A. Jeni, S. Tulyakov and H.-Y. Lee, "4Real: Towards photorealistic 4D scene generation via video diffusion models," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. [Online]. Available: https://openreview.net/forum?id=SO1aRpwVLk

[277] R. Li *et al.*, "4K4DGen: Panoramic 4D generation at 4K resolution," in *ICLR*, 2025. [Online]. Available: https://openreview.net/forum?id=qxRoo7ULCo

[278] H. Che, X. He, Q. Liu, C. Jin and H. Chen, "GameGen-X: Interactive open-world game video generation," in *International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=8VG8tpPZhe

[279] K. Xu, J. Stewart and E. Fiume, "Constraint-based automatic placement for scene composition," in *Graphics Interface*, 2002. [Online]. Available: https://api.semanticscholar.org/CorpusID:1874109

[280] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser and P. Hanrahan, "Example-based synthesis of 3D object arrangements," in *ACM SIGGRAPH Asia 2012 papers*, ser. SIGGRAPH Asia '12, 2012.

[281] Y. Zhao, K. Lin, Z. Jia, Q. Gao, G. Thattai, J. Thomason and G. S. Sukhatme, "LUMINOUS: indoor scene generation for embodied AI challenges," *CoRR*, vol. abs/2111.05527, 2021. [Online]. Available: https://arxiv.org/abs/2111.05527

[282] A. Raistrick *et al.*, "Infinite photorealistic worlds using procedural generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 12 630–12 641.

[283] G. Gao, W. Liu, A. Chen, A. Geiger and B. Schölkopf, "GraphDreamer: Compositional 3D scene synthesis from scene graphs," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[284] W. Feng, W. Zhu, T.-j. Fu, V. Jampani, A. Akula, X. He, S. Basu, X. E. Wang and W. Y. Wang, "LayoutGPT: Compositional visual planning and generation with large language models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[285] Y. Yang *et al.*, "Holodeck: Language guided generation of 3D embodied ai environments," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2024, pp. 16 227–16 237.

[286] L. Ling, C.-H. Lin, T.-Y. Lin, Y. Ding, Y. Zeng, Y. Sheng, Y. Ge, M.-Y. Liu, A. Bera and Z. Li, "Scenethesis: A language and vision agentic framework for 3D scene generation," 2025. [Online]. Available: https://arxiv.org/abs/2505.02836

[287] J.-E. Deschaud, "KITTI-CARLA: a KITTI-like dataset generated by CARLA Simulator," Nov. 2021, working paper or preprint. [Online]. Available: https://hal.science/hal-03445117

[288] J.-E. Deschaud, D. Duque, J. P. Richa, S. Velasco-Forero, B. Marcotegui and F. Goulette, "Paris-CARLA-3D: A real and synthetic outdoor point cloud dataset for challenging tasks in 3D mapping," *Remote Sensing*, vol. 13, no. 22, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/22/4713

[289] A. Xiao, J. Huang, D. Guan, F. Zhan and S. Lu, "Transfer learning from synthetic to real LiDAR point cloud for semantic segmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 3, 2022, pp. 2795–2803.

[290] Epic Games, "Unreal Engine." [Online]. Available: https://www.unrealengine.com

[291] S. Shah, D. Dey, C. Lovett and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, M. Hutter and R. Siegwart, Eds. Cham: Springer International Publishing, 2018, pp. 621–635.

[292] Y. Song, Z. Sun, Y. Wu, Y. Sun, S. Luo and Q. Li, "Learning Semantic Segmentation on Unlabeled Real-World Indoor Point Clouds via Synthetic Data," in *2022 26th International Conference on Pattern Recognition (ICPR)*, 2022, pp. 3750–3756.

[293] B. Iraci, *Blender Cycles: Lighting and Rendering Cookbook*. Packt Publishing, 2013.

[294] D. Robert, H. Raguet and L. Landrieu, "Efficient 3D Semantic Segmentation with Superpoint Transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 17 195–17 204.

[295] M. Deitke *et al.*, "Objaverse-XL: A Universe of 10M+ 3D Objects," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 35 799–35 813. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/70364304877b5e767de4e9a2a511be0c-Paper-Datasets_and_Benchmarks.pdf

[296] L. Li, M. Sung, A. Dubrovina, L. Yi and L. J. Guibas, "Supervised fitting of geometric primitives to 3D point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[297] S. Hu, A. Polette and J.-P. Pernot, "SMA-Net: Deep learning-based identification and fitting of CAD models from point clouds," *Eng. with Comput.*, vol. 38, no. 6, p. 5467–5488, Dec. 2022. [Online]. Available: https://doi.org/10.1007/s00366-022-01648-z

[298] Y. Liu, A. Obukhov, J. D. Wegner and K. Schindler, "Point2CAD: Reverse engineering CAD models from 3D point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 3763–3772.

[299] Z. Li, W. Zhang and J. Shan, "Holistic parametric reconstruction of building models from point clouds," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLIII-B2-2020, pp. 689–695, 2020. [Online]. Available: https://isprs-archives.copernicus.org/articles/XLIII-B2-2020/689/2020/

[300] Y. Marchand, L. Caraffa, R. Sulzer, E. Clédat and B. Vallet, "Evaluating Surface Mesh Reconstruction Using Real Data," *Photogrammetric engineering and remote sensing*, vol. 89, no. 10, pp. 625–638, Oct. 2023. [Online]. Available: https://hal.science/hal-04428121

[301] S. Ye, D. Chen, S. Han and J. Liao, "Learning with noisy labels for robust point cloud segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 6443–6452.

[302] A. Nguyen, J. Yosinski and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[303] M. Hein, M. Andriushchenko and J. Bitterwolf, "Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[304] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *Proceedings of International Conference on Learning Representations*, 2017.

[305] L. Kong, Y. Liu, X. Li, R. Chen, W. Zhang, J. Ren, L. Pan, K. Chen and Z. Liu, "Robo3D: Towards robust and reliable 3D perception against corruptions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 19 994–20 006.

[306] A. Mehrtash, W. M. Wells, C. M. Tempany, P. Abolmaesumi and T. Kapur, "Confidence calibration and predictive uncertainty estimation for deep medical image segmentation," *IEEE Transactions on Medical Imaging*, vol. 39, pp. 3868–3878, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:208512953

[307] L. Kong, X. Xu, J. Cen, W. Zhang, L. Pan, K. Chen and Z. Liu, "Calib3D: Calibrating model preferences for reliable 3D scene understanding," in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, February 2025, pp. 1965–1978.

[308] H. Vassilev, M. Laska and J. Blankenbach, "Uncertainty-aware point cloud segmentation for infrastructure projects using Bayesian deep learning," *Automation in Construction*, vol. 164, p. 105419, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926580524001559

[309] C. Petschnigg and J. Pilz, "Uncertainty estimation in deep neural networks for point cloud segmentation in factory planning," *Modelling*, vol. 2, no. 1, pp. 1–17, 2021. [Online]. Available: https://www.mdpi.com/2673-3951/2/1/1

[310] J. Xu, S. Yang, X. Li, Y. Tang, Y. Hao, L. Hu and M. Chen, "PDF: A probability-driven framework for open world 3D point cloud semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 5977–5986.

[311] J. Li, C. Saltori, F. Poiesi and N. Sebe, "Cross-modal and uncertainty-aware agglomeration for open-vocabulary 3D scene understanding," in *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, June 2025, pp. 19 390–19 400.

[312] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

A

Additional details on segmentation metrics

In this appendix, we briefly present the segmentation metrics used in this manuscript, specifically the Overall Accuracy (OA), the Average Accuracy (AA), and the mean Intersection over Union (mIoU). These metrics are classically used in the field of semantic segmentation tasks [23, 120, 39]. To define these metrics, we use the confusion matrix (Table A.1) and the relations below:

- TP (True Positive): the number of points correctly classified as belonging to the class of interest
- TN (True Negative): the number of points correctly classified as belonging to another class of interest
- FP (False Positive): the number of points incorrectly classified as belonging to the class of interest
- FN (False Negative): the number of points incorrectly classified as belonging to another class of interest

Ground Truth	**Predicted: Class 1**	**Predicted: Class 2**	**Predicted: Class 3**	...
Class 1	True Positive (TP)	False Positive (FP)	False Positive (FP)	...
Class 2	False Negative (FN)	True Positive (TP)	False Positive (FP)	...
Class 3	False Negative (FN)	False Negative (FN)	True Positive (TP)	...
...	...	...	...	...

Table A.1: Example of a confusion matrix for multi-class segmentation.

The Overall Accuracy (OA) measures the proportion of correctly classified points to the total number of points:

$$OA = \frac{\text{Number of correctly segmented points}}{\text{Total number of points}} = \frac{\text{TP + TN}}{\text{TP + TN + FP + FN}} \tag{A.1}$$

In the case of an balanced dataset, this metric can stand as a good indicator of the semantic segmentation performance. However, in cases of class imbalances, which are common in point clouds applications, this metric emphasizes the majority class (*e.g.*, "Covering", "Slab", "Wall").

The Average Accuracy (AA) represents the average accuracy across all semantic classes:

$$AA = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i + TN_i}{FP_i + FN_i} \tag{A.2}$$

where C is the number of classes and TP_i, TN_i, FP_i, and FN_i are the True Positives, True Negatives, False Positives and False Negatives for class i. Contrary to OA, AA takes into account class imbalance and enables treating all classes equally.

The Intersection over Union, also known as the Jaccard index, is a more representative metric since it measures the overlap between the predicted mask and the ground truth mask. Due to this property, this metric is particularly adapted to evaluate segmentation tasks. For class i, the IoU is defined as:

$$IoU_i = \frac{TP_i}{TP_i + FP_i + FN_i} \tag{A.3}$$

where TP_i, FP_i and FN_i are the True Positives, False Positives and False Negatives for class i.

The mean Intersection over Union (mIoU), commonly used in point cloud semantic segmentation [120, 37, 204, 107], represents the average IoU of all classes and can help mitigate class imbalances.

$$mIoU = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i + FP_i + FN_i} \tag{A.4}$$

where C is the number of classes and TP_i, FP_i and FN_i are the True Positives, False Positives and False Negatives for class i.

B

Additional results on handcrafted features

In this appendix, we present additional results and explanations regarding the experiments conducted with PointNeXt-S [100] and handcrafted features in Chapter 3.

Table B.1: Standard deviation for PointNeXt-S on 3DSES-Gold 🥇 with several handcrafted features. *Results from PointNeXt-S are the results of three runs.* Mean intersection over union (mIoU), overall accuracy (OA), average accuracy (AA).

RGB	LPS_cV	OAS_eS_v	OA	AA	mIoU	σ OA	σ AA	σ mIoU
✅	❌	❌	83.14	34.75	29.13	0.83	1.36	0.91
✅	❌	❌	81.67	37.46	30.91			
✅	❌	❌	81.73	36.29	30.37			
✅	✅	❌	84.53	35.91	30.51	1.70	1.51	1.77
✅	✅	❌	87.58	37.98	33.46			
✅	✅	❌	87.37	38.84	33.69			
✅	❌	✅	88.15	36.65	32.62	2.18	1.46	1.84
✅	❌	✅	85.46	36.90	31.90			
✅	❌	✅	83.84	34.25	29.13			

Table B.2: Standard deviation for PointNeXt-S on 3DSES-Silver 🥈 with several handcrafted features. *Results from PointNeXt-S are the results of three runs.* Mean intersection over union (mIoU), overall accuracy (OA), average accuracy (AA).

RGB	LPS_cV	OAS_eS_v	OA	AA	mIoU	σ OA	σ AA	σ mIoU
✅	❌	❌	93.63	65.90	60.26	2.20	2.76	3.83
✅	❌	❌	93.49	63.62	58.77			
✅	❌	❌	89.76	60.40	53.00			
✅	✅	❌	93.61	64.12	59.29	0.29	1.04	1.26
✅	✅	❌	94.19	65.68	61.31			
✅	✅	❌	93.87	66.10	61.60			
✅	❌	✅	93.18	62.05	57.29	0.39	0.77	1.19
✅	❌	✅	92.58	62.64	57.08			
✅	❌	✅	92.44	61.12	55.13			

Table B.3: Standard deviation for PointNeXt-S on 3DSES-Bronze 🥉 with several handcrafted features. *Results from PointNeXt-S are the results of three runs.* Mean intersection over union (mIoU), overall accuracy (OA), average accuracy (AA).

RGB	LPS_cV	OAS_eS_v	OA	AA	mIoU	σ OA	σ AA	σ mIoU
✅	❌	❌	94.57	66.47	61.79	0.11	1.78	1.15
✅	❌	❌	94.48	69.92	63.81			
✅	❌	❌	94.36	67.42	61.83			
✅	✅	❌	93.38	69.38	63.58	0.52	1.96	1.48
✅	✅	❌	94.21	71.85	65.70			
✅	✅	❌	94.34	73.24	66.43			
✅	❌	✅	93.18	62.05	57.29	0.42	3.85	3.60
✅	❌	✅	92.58	62.64	57.08			
✅	❌	✅	92.44	61.12	55.13			

C

Additional details about 3DSES

In this appendix, we present additional results and explanations regarding the indoor dataset created during this thesis and presented in Chapter 4: 3DSES.

C.1 Dataset structure

All the information presented in this section is reproduced in the "README" file of the 3DSES dataset archive. The dataset is hosted on Zenodo for public release under the Creative Commons CC BY-SA 4.0 license. It is currently available through an open-access repository: `https://zenodo.org/records/13323342`. The `zip` archive contains three folders, one for each variant:

- '`Gold/`': contains Gold point clouds,
- '`Silver/`': contains Silver point clouds,
- '`Bronze/`': contains all raw point clouds[1] for the Bronze version.

We provide our three variants of 3DSES: Gold, Silver and Bronze. We use the NumPy [312] `.npy` format to store the point clouds. Point clouds are organized per scan, identified by `SXXX`, where `XXX` is a three-digits integer. Three test point clouds are available with private labels: scans `S170`, `S171` and `S180`. These labels for theses point clouds are kept hidden for use as evaluation on a competition: `https://www.codabench.org/competitions/6927/`. Point clouds can be opened with NumPy using Python, *e.g.* with `numpy.load`.

[1] Directly exported from Register360.

Table C.1: Column signification in the point clouds files.

Index	Feature	Description
0	x	Point coordinates (xyz)
1	y	in an orthonormal basis
2	z	with z the height.
3	r	Color in RGB format
4	g	encoded as uint8 $[0, 255]$
5	b	
6	Intensity	Lidar intensity encoded as float32 $[0, 1]$
7	Real label	Manually annotated class in $[\![0, n^{\dagger}]\!]$
8	Pseudo label	Automatically annotated class in $[\![0, n^{\dagger}]\!]$

† $n = 17$ for Gold and $n = 11$ for Silver/Bronze.

For Gold and Silver versions, the `.npy` files contain 9 columns, where each row describe one point in the scan. Column signification is detailed in Table C.1.

Labels, for Gold version, are in the range of $[\![0, 17]\!]$. For the Silver versions, labels are in the range $[\![0, 11]\!]$ instead since the classification uses only 12 classes.

Since the Bronze variant of 3DSES only contains pseudo-labels, column #7 contains instead the pseudo label for these scans, and column #8 is dropped.

Preprocessing We provide unnormalized color information, *i.e.* RGB values are comprised in $[0 - 255]$. Regarding the (xyz) coordinates, we apply a translation from the initial georeferenced point cloud to obtained centered and smaller coordinates that are more convenient for use in deep models.

CAD model The CAD model is currently distributed as an `.obj` file and will be made available in an open format supporting the IFC standard for public release. This CAD model, can be visualize with a 3D data processing software (such as CloudCompare [4])

C.2 Classification and label signification

Taxonomy 3DSES uses a BIM-oriented class taxonomy, that focuses on modeling both the structure of a building and its functional equipment. Classes composed of structural elements (*e.g.* covering, slab, clutter) are the easiest to understand, since they are the technical terms for the common parts of a building: walls, floors,

Table C.2: Class definitions for the 3DSES dataset.

Index	Class name	Definition
0	Column	vertical structural element that supports weight, typically made of concrete, or metal.
1	Components	building equipment excluding furniture (*e.g.* trash bin, hotspot wifi, electronics, etc.)
2	Covering	upper interior element of a room (*e.g.* suspended ceiling)
3	Damper	smoke detectors
4	Door	moving building element that provides access for people to pass through
5	Exit sign	building element that indicates emergency exit, typically with green lighting
6	Fire terminal	building equipment for fire safety, that provides fluid to suppress fire or that triggers audible alarms
7	Furniture	Common furnishings such as chairs, tables, etc.
8	Heater	building element that provides heat, includes the pipes
9	Lamp	building element that provides artificial light
10	Outlet	utility element that provides access to electrical power
11	Railing	frame assembly adjacent to some boundaries or human circulations (*e.g.* stairs)
12	Slab	structural element providing the lower support (often made in concrete)
13	Stair	structural element that allows moving between floors
14	Switch	utility element that controls the flow of electricity, typically to a lamp
15	Wall	vertical structural element, often made of stone or concrete, that divides or encloses a space
16	Window	building element that provides natural light and/or fresh air
17	Clutter	all elements that are unrelated to the building structure and equipment, *e.g.* clothes, plants, small office supplies, persons, etc.

ceilings, etc. However, 3DSES also includes domain-specific classes that regroup many different types of objects. We have grouped such utilities by domain according to their purpose, *e.g.* fire suppression, heating, electrical systems, lighting, etc.. We detail in Table C.2 the definition of every class in the dataset.

Simplification for the Silver and Bronze variants The 3DSES Silver variant uses a less detailed classification than 3DSESGold. In particular, it focuses more on structural elements. To obtain this simplified classification, we followed three principles:

1. remove all small individual objects,
2. remove all objects that do not have a well defined 3D CAD model,
3. remove all objects that are not widely represented in the point clouds.

This resulted in the following simplifications applied the classification:

- we merge all objects from classes “Outlet” and “Switch” into the class “Wall”,
- objects from the “Damper” class are either merged into “Covering” or the “Clutter” class, depending on

Table C.3: Simplified class definitions for the 3DSES dataset.

Index	Class name	Definition
0	Column	vertical structural element that supports weight, typically made of concrete, or metal.
1	Covering	upper interior element of a room (*e.g.* suspended ceiling)
2	Door	moving building element that provides access for people to pass through
3	Exit sign	building element that indicates emergency exit, typically with green lighting
4	Heater	building element that provides heat, includes the pipes
5	Lamp	building element that provides artificial light
6	Railing	frame assembly adjacent to some boundaries or human circulations (*e.g.* stairs)
7	Slab	structural element providing the lower support (often made in concrete)
8	Stair	structural element that allows moving between floors
9	Wall	vertical structural element, often made of stone or concrete, that divides or encloses a space
10	Window	building element that provides natural light and/or fresh air
11	Clutter	all elements that are unrelated to the building structure and equipment, *e.g.* clothes, plants, small office supplies, persons, etc.

their distance to the closest points from these classes (smoke detectors are usually mounted to the ceiling),

- objects from the "Fire terminal" class are either merged into "Wall" or the "Clutter" class, depending on their distance to the closest points from these classes (fire alarms are wall-mounted, extinguishers tend to stuck out),
- non-structural elements, *i.e.* objects from the "Component" and "Furniture" classes, are reclassified as "Clutter".

Note that the only exception to the general simplification principles is the "Exit sign" class, since this class always represent the same object and is therefore accurately modeled in the CAD model, and is present in most scans due to safety regulations. For these reasons, we chose to keep the "Exit sign" class in the simplified classification of the Silver variant. The final Silver/Bronze classification is summarized in Table C.3.

C.3 Additional details on the pseudo-labeling alignment algorithm

As stated in Chapter 4, the pseudo-labels are generated using an alignment algorithm based on cloud-to-mesh distance computation. This distance uses the Metro algorithm [209], as implemented in CloudCompare [4]. We give below some additional insights on this algorithm, its requirements and its accuracy.

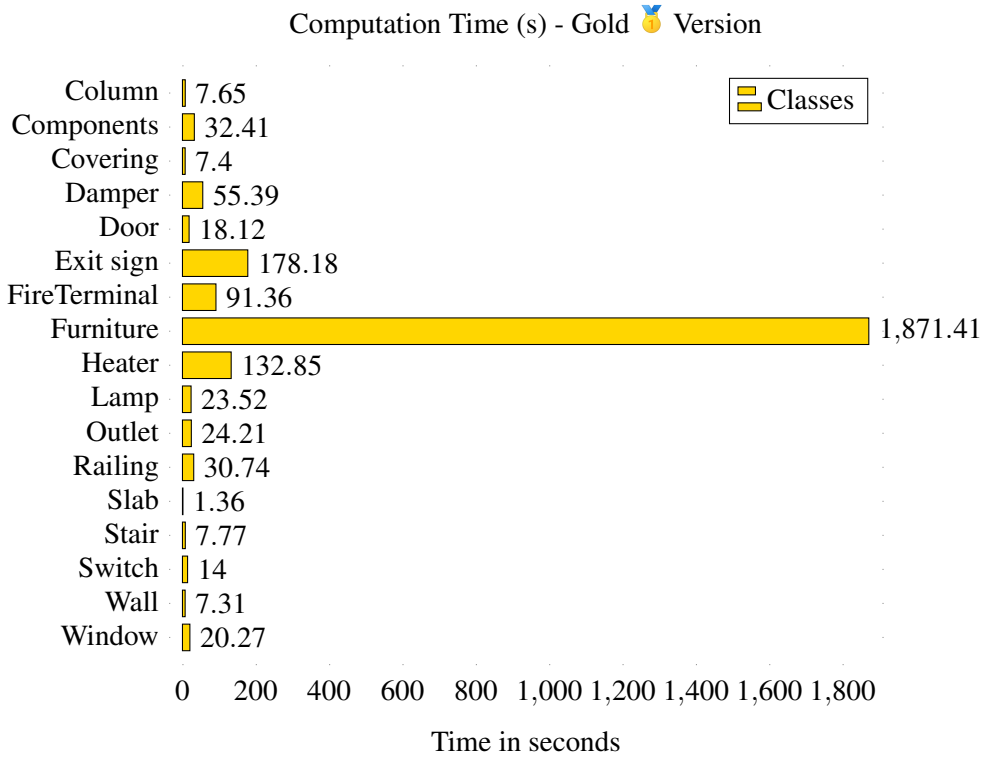


Figure C.1: Computation time in seconds for each classes of Gold version

C.3.1 Computational requirements

Operations on 3D point clouds tend to be computationally demanding, especially when density increases. To be useful, the pseudo-labeling strategy should be cost effective, with less dependency on human annotators, but also time effective. In practice, the bottleneck for pseudo-labeling based on the 3D model is the creation of the 3D model. Indeed, we present in Histogram C.1 the processing times required to label the point clouds on the Gold version of 3DSES. All computations have been run on a consumer Intel(R) Xeon(R) CPU E5-2643 v4 @ 3.40 GHz.

As can be expected, processing times to align the mesh to the point cloud vary depending on the complexity of the 3D shapes of the objects, and the size of the 3D point clouds. For example, pseudo-labeling of slab is significantly faster than on furniture. With 1871.41 s for "Furniture" against only 1.36 s for "Slab". It's $\approx$1376 times longer for our alignment methods to create "Furniture" pseudo-labels. In Histogram C.1, we observed that all complex object shapes, such as "Exit sign", "Fire Terminal", "Furniture" and "Heater" have computation time $> 90s$ (largely superior as other classes). Overall, it takes approximately 42 minutes to

pseudo-label one scan. The complete process therefore takes about 7 hours to annotate all point clouds for the Gold version with 18 classes. Since the Silver and Bronze variants of 3DSES contain fewer classes, the processing time for the alignment algorithm is significantly lower. This is due in part to the absence of the "Furniture" in those datasets. In practice, Silver can be pseudo-labeled in less than 4 hours and half ($\approx$ 9 minutes per scan) and Bronze is pseudo-labeled in around 9 hours ($\approx$ 13 minutes per scan). The additional time required for Bronze version can be explained by the higher number of points in each scan compared to the Silver version.

C.3.2 Evaluation of the pseudo-labels

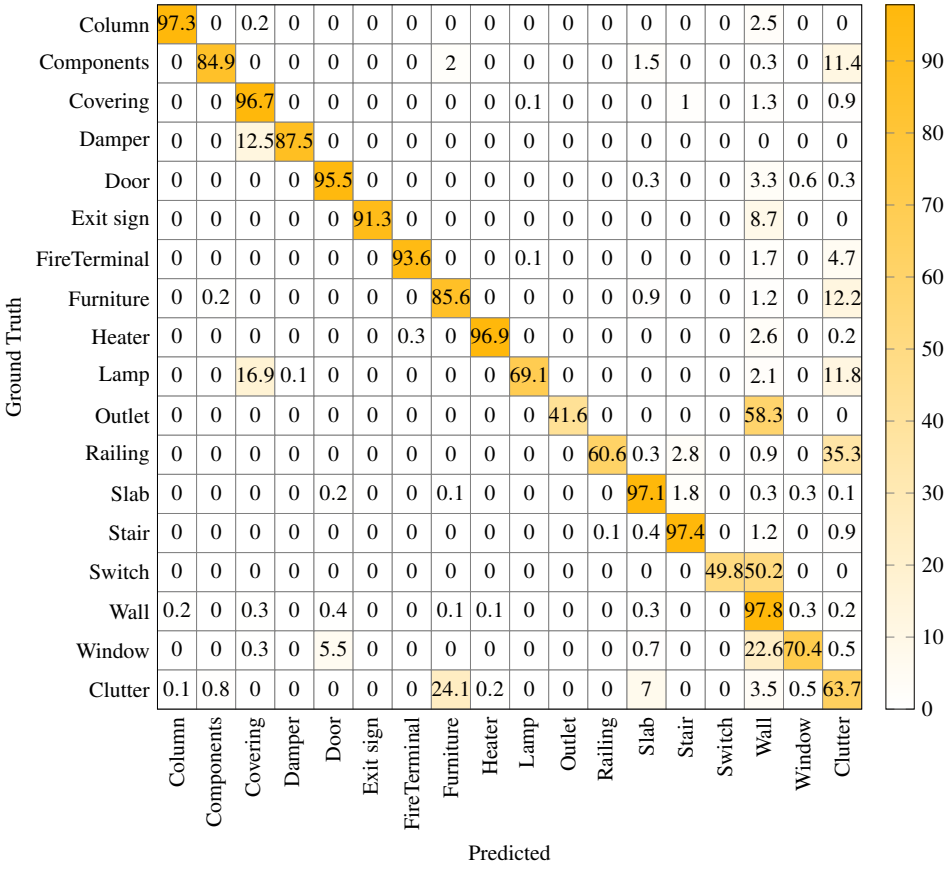


Figure C.2: Confusion matrix for Gold Version 🥇

In addition to the main metrics provided in the Chapter 4, we detail below the full confusion matrices between the pseudo-labels and the manually annotated ground truth for the Gold (Fig. C.2) and Silver (Fig. C.3) variants. Note that these two matrices are normalized by line.

We observe in the confusion matrix that these points are classified as "Wall". Indeed, the alignment

algorithms merge these small objects into the wall. Reducing the threshold for classifying points as “Wall” could alleviate this problem, but would in return generate more false positive “Clutter” points. Note also the relatively low score (60.6 %) for “Railing”, which can be explained by the mismatch between the 3D railings and the actual physical railings at ESGT. The same observation holds partially for “Window”, as the windows are modeled with a standard frame that does not perfectly match the actual windows.

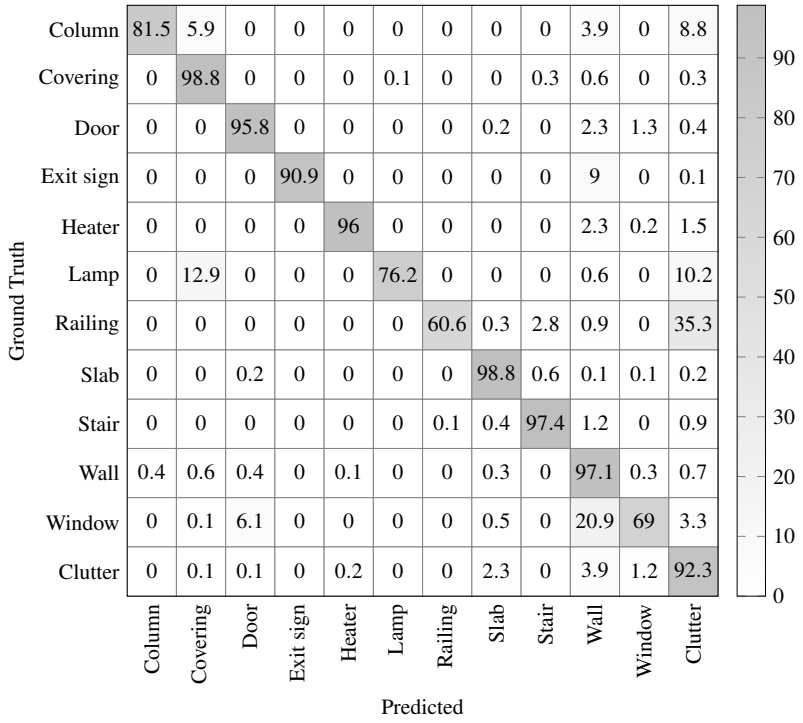


Figure C.3: Confusion matrix for Silver Version

Table C.4: Standard deviation for segmentation metrics obtained with intensity & features on the 3DSES test set. *Results from PointNeXt-S are the results of 3 runs.* Mean intersection over union (mIoU), overall accuracy (OA), average accuracy (AA).

	RGB	I	LPS_cV	OAS_eS_v	OA	AA	mIoU	σ OA	σ AA	σ mIoU
Gold	✅	❌	❌	❌	82.18	36.17	30.14	0.83	1.36	0.91
	✅	✅	❌	❌	90.66	49.09	43.65	0.80	0.15	0.31
	✅	✅	✅	❌	89.88	47.86	41.99	0.48	0.80	0.64
	✅	✅	❌	✅	89.69	47.85	42.32	0.40	0.65	0.77
Silver	✅	❌	❌	❌	92.29	63.30	57.34	2.20	2.76	3.83
	✅	✅	❌	❌	93.30	69.74	63.44	0.83	0.61	0.85
	✅	✅	✅	❌	93.90	68.73	63.44	0.53	3.25	3.35
	✅	✅	❌	✅	93.47	69.42	63.72	0.67	0.94	0.71
Bronze	✅	❌	❌	❌	94.47	67.94	62.48	0.11	1.78	1.15
	✅	✅	❌	❌	93.76	74.55	66.89	0.57	1.76	1.89
	✅	✅	✅	❌	94.87	75.22	68.21	0.20	0.53	0.62
	✅	✅	❌	✅	94.13	75.75	67.91	0.73	1.21	1.60

D

Résumé long en français

Segmentation de petits objets dans les nuages de points 3D

D.1 Introduction

Cette thèse vise à améliorer la segmentation sémantique des petits objets dans les nuages de points 3D. Ces nuages de points sont couramment utilisés pour représenter des espaces intérieurs, tels que des bâtiments ou des maisons, et produire des maquettes numériques (cf. Fig. D.1) ou des plans. Au cours des dernières années, les technologies d'acquisition 3D se sont considérablement améliorées, ce qui a rendu les scanners lasers plus accessibles aux utilisateurs et abordables. Les scanners lasers sont maintenant couramment utilisés dans

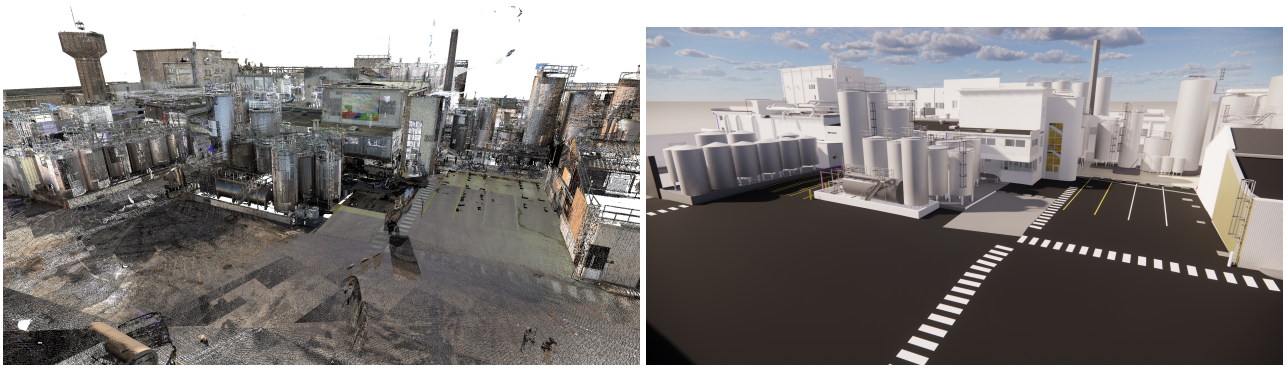


Figure D.1: Dans cette thèse, nous abordons la segmentation des petits objets dans les nuages de points 3D pour la création de maquettes numériques. Les méthodes existantes n'exploitent pas entièrement les possibilités offertes par les modèles 3D pré-existants.

divers domaines, tels que la construction, l'industrie et l'urbanisme. Malgré ces avancées, le traitement des données 3D reste un défi, notamment pour l'identification et la segmentation des petits objets. Cette thèse est une CIFRE réalisée avec QUARTA, société de géomètres-experts. Elle s'appuie sur l'exploitation de données 3D anciennes et volumineuses. Elle s'inscrit dans un contexte marqué par l'essor rapide de l'intelligence artificielle et de l'apprentissage automatique. Face aux limites des méthodes traditionnelles (comme RANSAC [25] ou Hough [26]) pour la segmentation des nuages de points, cette thèse explore le potentiel de l'apprentissage profond pour détecter des objets dans les nuages de points. La segmentation sémantique, qui consiste à attribuer une classe (par exemple, "prise électrique" ou "chaise") à chaque point du nuage, permettrait d'améliorer la modélisation des petits objets, de faciliter la visualisation des données et d'optimiser l'espace de stockage. Cependant, cette tâche reste complexe en raison des occlusions et artefacts fréquents dans les nuages de points, ainsi que du manque de jeux de données annotés, essentiels pour l'apprentissage supervisé. Pour répondre à ces problématiques, cette thèse propose des approches innovantes pour améliorer la précision de la segmentation sémantique.

Axes de recherche

Cette thèse est organisée autour de trois axes de recherche principaux :

1. **Caractéristiques géométriques et radiométriques :** Les caractéristiques additionelles à la géométrie de TLS peuvent-elles améliorer la segmentation sémantique des petits objets (telles que des caractéristiques expertes et radiométriques) ?
2. **Temps nécessaire pour étiqueter les petites instances :** Les données de topographie peuvent-elles être exploitées pour segmenter les petits objets ?
3. **Pénurie de données 3D étiquetées :** Des données 3D synthétiques avec des petits objets peuvent-elles être générées pour pré-entraîner des modèles d'apprentissage profond ?

Chapitre 2 : Etat de l'art : Ce chapitre présente les différentes approches décrites dans la littérature scientifique pour appliquer l'apprentissage profond aux nuages de points. Il met également en lumière la prédominance des méthodes d'apprentissage supervisé, ainsi que les difficultés liées à l'annotation des données 3D.

Chapitre 3 : Amélioration de la segmentation avec des caractéristiques additionelles : Ce chapitre explore les possibilités d'intégrer des caractéristiques géométriques et radiométriques dans les architectures profondes pour améliorer la précision de la segmentation de nuages de points par apprentissage profond.

Chapitre 4 : Un jeu de données de segmentation de nuages de points intérieurs réaliste : Dans ce chapitre, nous proposons une méthode pour valoriser les données volumineuses disponibles dans le monde industriel. Ce chapitre présente une approche innovante qui utilise des données de topographie pour réduire le temps d'annotation des nuages de points 3D. Nous mettons à la disposition de la communuté un nouveau jeu de données de nuages de points d'intérieur de bâtiments.

Chapitre 5 : Scènes synthétiques 3D pour la segmentation de petits objets : Dans ce chapitre, nous abordons le manque de données 3D annotées. Ce chapitre présente une approche qui permet la génération de nuages de points synthétiques d'intérieurs de bâtiments avec des petits objets pour pré-entraîner des modèles d'apprentissage profond.

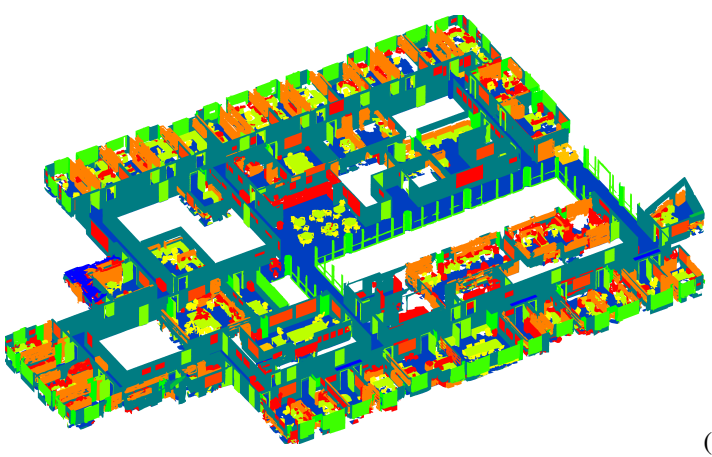
Chapitre 6 : Conclusion : Résumé des contributions de la thèse et présentation des perspectives.

D.2 Etat de l'art

Nos contributions s'appuient sur diverses parties de la littérature détaillées dans ce chapitre, et notamment sur trois éléments clés : la faible utilisation de l'intensité LiDAR, la forte dépendance des méthodes de segmentation profonde à l'apprentissage supervisé et le faible nombre de jeux de nuages de points annotés.

Balayage laser terrestre : Les mesures TLS (Terrestrial Laser Scanner) reposent sur l'émission et la réception de rayons laser. Lors des acquisitions, l'intensité du signal est souvent enregistrée, accompagnée des coordonnées 3D. L'intensité correspond au ratio entre la puissance lumineuse reçue et la puissance émise par le scanner laser. Pour pouvoir être utilisable, cette grandeur doit être corrigée des paramètres d'acquisition (capteur, distance à l'objet, atmosphère, ...). Plusieurs méthodes d'étalonnage existent dans la littérature, basées soit sur des données empiriques, soit sur des modèles mathématiques. Cependant, aucune méthode n'a fait consensus chez les constructeurs, raison pour laquelle les valeurs d'intensité mesurées diffèrent selon les scanners lasers utilisés.

Apprentissage supervisé : Nous présentons dans un second temps, un état de l'art des approches et méthodes permettant d'appliquer l'apprentissage profond pour la segmentation sémantique de nuages de points, depuis les méthodes utilisant des images (MVCNN [74]), à celles utilisant des grilles régulières (VoxNet [82]) ou plus récemment, celles opérant directement sur le nuage de points (PointNet [67]). Nous présentons également les différentes avancées des architectures en pointe de l'état de l'art, comme PointNeXt [100], Swin3D [186]. . . . Ces méthodes relèvent de l'apprentissage supervisé et nécessitent donc des données annotées.



(a) Vue de haut de l'aire 5 de S3DIS, colorisée par classe.

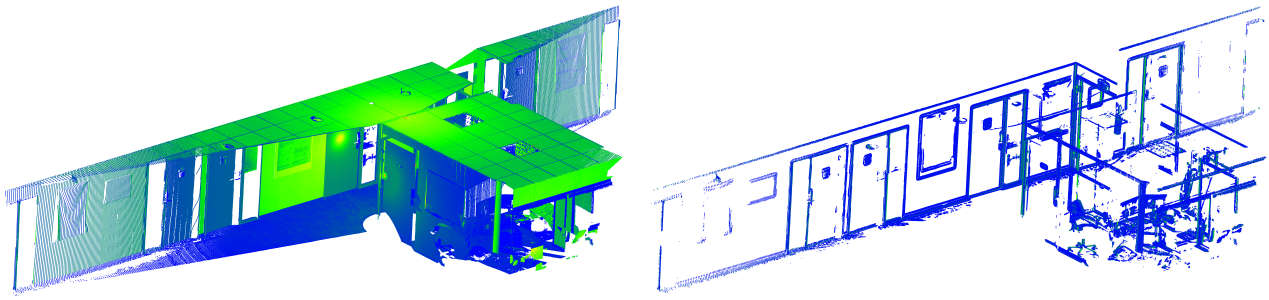
(b) Vue de haut d'un nuage de points issu de ScanNet, colorisé par classe.

Figure D.2: Vue de dessus des nuages de points 3D provenant de deux jeux de données couramment utilisés pour la segmentation de nuages de points intérieurs : S3DIS (Fig. D.2a) et ScanNet (Fig. D.2b). Dans ces deux jeux de données, la classification ne représente pas avec précision la diversité des objets dans le monde réel, et certains objets sont regroupés en structures plus grandes (par exemple, les interrupteurs sont inclus dans les murs).

Un faible nombre de jeux de données annotés : La segmentation de nuages de points intérieurs pâtit du manque de données annotées. Les jeux de données existants (cf. Fig. D.2) ont des limitations quant aux classes disponibles (par exemple ScanNet [121]) et à la surface couverte (par exemple S3DIS [120]). Cependant, l'étiquetage des nuages de points est un travail laborieux, réservé à des utilisateurs expérimentés, qui nécessite plusieurs dizaines d'heures (par exemple, 500 heures pour ScanNet). De ce fait, les jeux de données 3D ne reflètent pas la répartition réelle des objets. Les plus petits d'entre-eux sont regroupés dans des objets plus imposants (comme les interrupteurs dans les murs). Dans cette partie, nous avons étudié les solutions alternatives aux données annotées manuellement, comme les données synthétiques et l'apprentissage auto-supervisé, et montré les limites de ces méthodes.

D.3 Amélioration de la segmentation avec des caractéristiques additionelles

Dans ce chapitre, nous avons cherché à évaluer l'apport des caractéristiques géométriques (cf. Fig. D.3) sur la segmentation sémantique de nuages de points 3D dans différents modèles profonds. Nous avons démontré que même à l'ère du deep learning 3D, les caractéristiques conçues à la main permettent encore d'améliorer les performances de la segmentation sémantique, en particulier pour les petits modèles et les petits jeux de données. Ces résultats prometteurs indiquent la possibilité de tirer profit des anciens nuages de points 3D de géomètres



(a) Points avec l'intensité LiDAR (b) Points filtrés en fonction de la courbure locale

Figure D.3: Les caractéristiques manuelles peuvent aider à identifier les points intéressants, tels que ceux situés dans les coins. Propres à l'acquisition LiDAR, d'autres caractéristiques peuvent être disponibles telles que l'intensité. Exemple provenant de 3DSES [1].

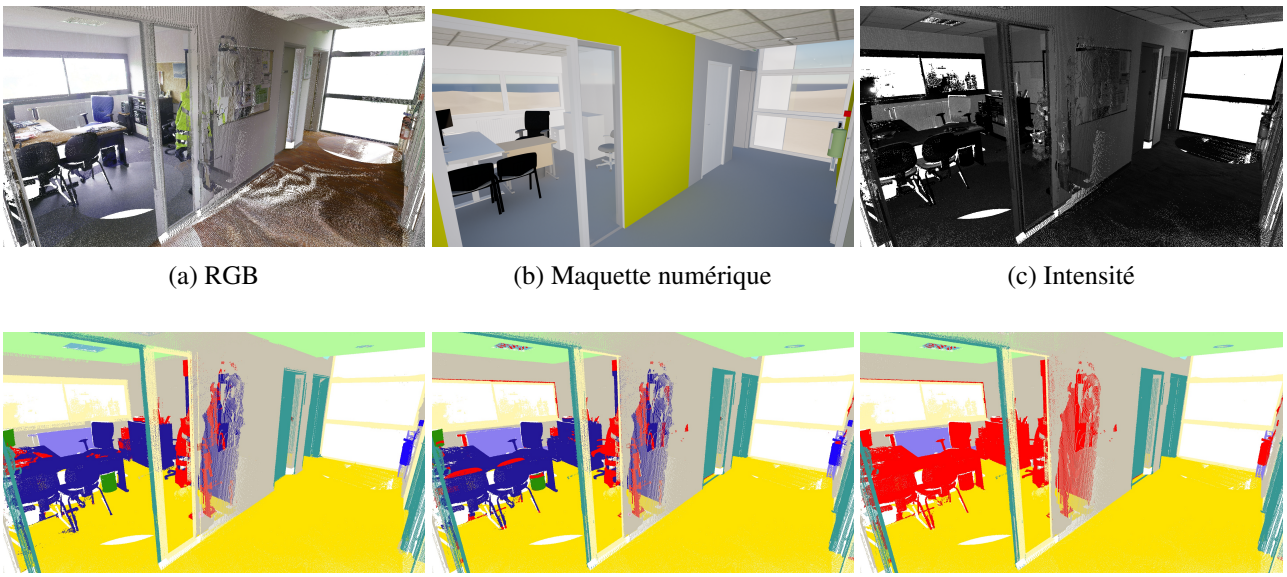
qui ne contiennent que les coordonnées XYZ et l'intensité. De plus, les résultats présentés dans ce chapitre suggèrent que les caractéristiques peuvent être plus utiles pour les architectures profondes de petites tailles. Cependant, le rôle des caractéristiques conçues à la main sur les petits objets est ambigu et nécessite d'autres expériences.

Nous avons également introduit la méthode dite "DropFeatures" pour régulariser les modèles profonds et les rendre robustes face aux caractéristiques manquantes en les réinitialisant aléatoirement à zéro pendant l'entraînement. En pratique, cette stratégie nous permet d'obtenir des modèles qui surpassent les modèles de référence lors de l'évaluation avec des caractéristiques, tout en maintenant une segmentation de performance égale sans ces caractéristiques.

Concernant la caractéristique radiométrique, à savoir l'intensité, nous avons testé plusieurs TLS dans la salle de métrologie de l'ESGT. Ces tests ont montré les difficultés de l'étalonnage de cette grandeur étant donné sa grande sensibilité aux conditions expérimentales de lever lasergrammétrique (nature de la surface sur laquelle s'effectue la réflexion, angle d'incidence du rayon laser, distance à l'objet levé).

D.4 Un jeu de données de segmentation de nuages de points intérieurs réaliste

Dans ce chapitre, nous présentons 3DSES (3D Segmentation of ESGT point clouds), un nouveau jeu de données pour la segmentation sémantique de nuages de points 3D intérieurs denses acquis par LiDAR. 3DSES (cf. Fig. D.4) répond au besoin de jeux de données intérieurs TLS conçus pour l'étude et la modélisation de



(a) RGB (b) Maquette numérique (c) Intensité

(d) Gold - Etiquettes manuelles (e) Gold - Pseudo étiquettes (f) Silver - Pseudo étiquettes

Figure D.4: Modalités et variants d'annotation de 3DSES. Les étiquettes de Gold sont des annotations manuelles sur 18 classes, y compris les petits objets tels que les interrupteurs électriques et les prises de courant. Les pseudo-étiquettes sont obtenues en alignant automatiquement le modèle 3D sur le nuage de points, introduisant un peu de bruit dans l'annotation (voir par exemple le dessus des chaises). Les étiquettes de Silver utilisent une classification simplifiée avec uniquement 12 catégories (par exemple, la poubelle est maintenant simplement "Clutter"). Légende : Poteaux en violet foncé, Composants en vert foncé, Plafonds en vert clair, Portes en vert, Signe d'urgence en bleu clair, Terminaux d'incendie en bleu foncé, Chauffage en violet clair, Lampes en bleu, Sol en jaune, Mur en gris, Fenêtre en jaune clair, Clutter en rouge.

bâtiments. Il contient une combinaison unique d'étiquettes de nuages de points pour la segmentation sémantique, un modèle 3D géoréférencé avec des objets orientés BIM. De plus, les nuages de points contiennent l'intensité LiDAR, caractéristique radiométrique non fournie dans les jeux de données existants.

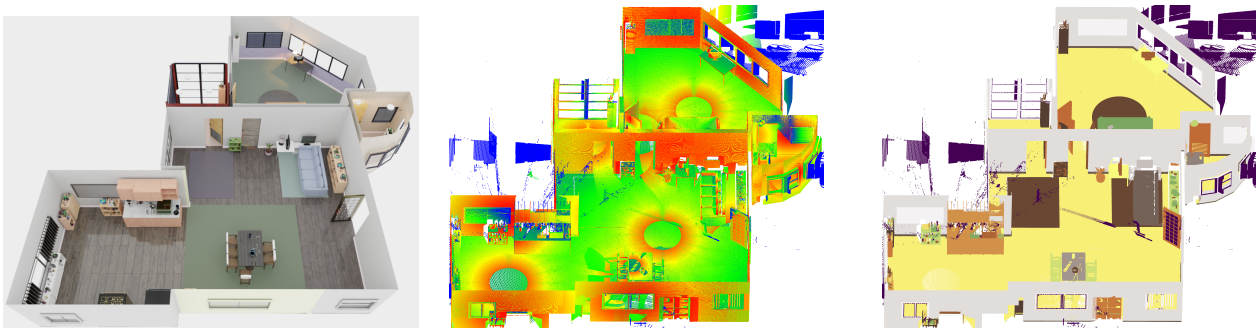
Nous montrons que l'utilisation de modèles 3D CAD pour annoter automatiquement les nuages de points est une stratégie efficace pour produire rapidement des pseudo-étiquettes avec une précision de 95% par rapport à une référence basée sur des données annotées manuellement. De plus, nous démontrons que l'entraînement sur des pseudo-étiquettes fournit des performances similaires à celles d'un entraînement sur des étiquettes manuelles dans 3DSES.

Nous montrons également que la précision de la segmentation peut bénéficier de l'intensité LiDAR dans des environnements intérieurs, malgré qu'elle soit souvent ignorée dans les travaux précédents. Les résultats de segmentation démontrent que 3DSES est un jeu de données difficile, en particulier pour les classes orientées BIM, par exemple des composants de petits bâtiments tels que les terminaux électriques et les systèmes de

sécurité. Dans ce chapitre, nous montrons également que nos méthodes peuvent efficacement mettre en valeur les "données gelées" provenant d'entreprises de géomètres et contribuer à créer des jeux de données avec des petits objets.

D.5 Scènes synthétiques 3D pour la segmentation de petits objets

Dans ce chapitre, nous avons mis en évidence qu'aucun travail existant ne combine la génération d'environnements 3D et l'acquisition virtuelle TLS dans des environnements intérieurs. Nous avons ensuite examiné les méthodes pour générer des environnements 3D intérieurs et simuler des levés par TLS. Nous avons proposé une méthode pour générer des nuages de points TLS denses avec des petits objets (comme des livres, des fourchettes, des lampes ou des cuillères). Dans un premier temps, notre méthode s'appuie sur un générateur procédural intérieur pour créer des scènes de hautes qualités contenant divers objets courants. Ensuite, conformément aux pratiques des géomètres, nous avons déplacé un TLS virtuel dans chaque pièce pour produire des nuages de points. Comme lors d'une acquisition réelle, nous portons une attention particulière au scan inter-pièces en réalisant un balayage laser sous chaque porte.



(a) Scène Blender générée par Infinigen [221]. (b) Nuage de points avec l'intensitée simulée (BLAINDER [212]). (c) Nuage de points avec des étiquettes sémantiques (BLAINDER [212]).

Figure D.5: Modalités de IPCS (Indoor Point Cloud Synthetic dataset). Les scènes Blender sont générées à l'aide d'Infinigen [221]. Ensuite, un système virtuel TLS [212] se déplace à l'intérieur de chaque environnement intérieur pour produire des nuages de points synthétiques avec intensité. Cette méthode permet d'attribuer des noms d'objet à chacun des points. Ces objets sont ensuite regroupés dans différentes classes sémantiques.

Notre approche a permis de générer le plus grand jeu de données synthétique intérieur disponible (cf. Fig. D.5), couvrant *plus de 64 milliards* de points et une superficie supérieure à celle de la plupart des jeux de données intérieurs réels : *103 717* m^2. De plus, cette approche permet d'obtenir des nuages de points avec une information d'intensité, des étiquettes sémantiques et des étiquettes d'instances.

D.6 Conclusion et perspectives

Ce chapitre résume nos contributions à l'amélioration de la segmentation sémantique de petits objets dans les nuages de points, avant d'envisager des perspectives de recherche. Dans cette thèse, nous proposons des approches pour améliorer la segmentation de petits objets dans les nuages de points par apprentissage profond, tout en valorisant des données existantes disponibles dans le monde industriel aujourd'hui sous exploitées.

Chapitre 3 : Amélioration de la segmentation avec des caractéristiques additionnelles : Dans ce chapitre, nous montrons que les récents modèles profonds, en particulier les architectures les plus petites, bénéficient des caractéristiques expertes. Cependant, puisque les caractéristiques conçues à la main ne sont pas toujours disponibles au moment du test, nous introduisons une stratégie "DropFeatures" qui réinitialise aléatoirement les caractéristiques pendant l'entraînement. Cette stratégie a été conçue pour rendre les modèles plus robustes à l'absence de caractéristiques. Nos résultats démontrent que la stratégie "DropFeatures" rend non seulement le modèle plus robuste, mais améliore également sa performance de segmentation.

Chapitre 4 : Un jeu de données de segmentation de nuages de points intérieurs réaliste : Ce chapitre présente une approche innovante qui utilise des données topographiques pour réduire le temps d'annotation des nuages de points 3D. Nous présentons un nouveau jeu de données de segmentation sémantique de nuages de points d'intérieurs de bâtiments, mis à la disposition de la communauté scientifique.

Chapitre 5 : Scènes synthétiques 3D pour la segmentation de petits objets : Nous proposons une méthode qui génère d'abord des scènes intérieures contenant des petits objets (comme des fourchettes, des cuillères, etc.), pour une surface supérieure à $103\,000\,m^2$. Nous utilisons ensuite un simulateur TLS virtuel pour générer des nuages de points intérieurs denses avec intensité. Grâce au simulateur TLS virtuel, nous pouvons obtenir des étiquettes sémantiques ainsi que des étiquettes d'instance pour plus de *64 milliards* de points.

Les travaux menés dans cette thèse ouvrent sur des perspectives à court terme et des perspectives à plus long terme:

- Perspectives à court terme
  - **Caractéristiques manuelles :** les acquisitions dans le monde industriel sont souvent réalisées sans couleurs. Des expériences supplémentaires sont nécessaires pour valider l'apport des caractéristiques géométriques notamment lorsque l'information colorimétrique est absente
  - **Information radiométrique :** des résultats prometteurs avec l'intensité ont été obtenus, mais des

expériences supplémentaires sont nécessaires pour étalonner cette grandeur notamment dans le cas d'acquisitions avec des capteurs différents

- **3DSES: segmentation d'instance :** il est possible d'étendre la méthode de transfert d'étiquettes (présentée dans le chapitre 4) pour produire des étiquettes de segmentation d'instances (en se basant sur l'identifiant d'objet dans le modèle 3D).
- **Très grands jeux de données annotés :** les nuages de points synthétiques du Chapitre 5 peuvent être colorisés à l'aide d'image issue de Blender pour créer un jeu de données de nuage de points colorisé.

- Perspectives à long terme

  - **Conception de méthodes pour créer automatiquement des modèles 3D :** l'apport de l'apprentissage profond ou de méthodes traditionnelles pour la reconstruction d'objets et l'estimation de paramètres (position et orientation) sera à étudier.
  - **Apprentissage faiblement supervisé :** les approches d'apprentissage semi-supervisé pourraient fournir des solutions aux difficultés de pseudo-annotation du Chapitre 4
  - **Qualification de l'incertitude :** des recherches sur la qualification d'incertitude seraient bienvenues pour qualifier les prédictions des modèles, surtout dans le monde des géomètres.