



HAL
open science

Efficient Learning on Large-Scale 3D Point Clouds

Damien Robert

► **To cite this version:**

Damien Robert. Efficient Learning on Large-Scale 3D Point Clouds. Computer Vision and Pattern Recognition [cs.CV]. Université Gustave Eiffel, 2024. English. NNT: . tel-04448827

HAL Id: tel-04448827

<https://hal.science/tel-04448827>

Submitted on 13 Feb 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Efficient Learning on Large-Scale 3D Point Clouds

Thèse de doctorat de l'Université Gustave Eiffel

École doctorale N°532, Mathématiques et Sciences et Technologies de l'Information et de la Communication (MSTIC)

Spécialité de doctorat : Informatique

Unité de recherche : Laboratoire des Sciences et Technologies de l'Information Géographique (LASTIG), IGN & laboratoire Computer Science and Artificial Intelligence (CSAI), ENGIE LAB CRIGEN

Thèse présentée et soutenue à l'Université Gustave Eiffel,
le 22 Janvier 2024, par :

DAMIEN ROBERT

Composition du Jury

Sébastien LEFEVRE

Professeur, IRISA, Université Bretagne Sud

Rapporteur

Cédric DEMONCEAUX

Professeur, ICB, Université de Bourgogne

Rapporteur

Siyu TANG

Assistant professor, VLG, ETH Zurich

Examinatrice

Duygu CEYLAN

Senior researcher, Adobe Research London

Examinatrice

Patrick PEREZ

CEO, Kyutai

Examineur

Loïc LANDRIEU

Chargé de recherche, LIGM/École des Ponts ParisTech - LASTIG/IGN

Directeur de thèse

Bruno VALLET

Chargé de recherche, LASTIG/IGN

Co-Directeur de thèse

Dmitriy SLUTSKIY

Ingénieur de recherche, CSAI, ENGIE LAB CRIGEN

Invité

To whom it may concern.

Abstract

Over the past decade, deep learning has advanced the analysis of text, image, audio, and video. More recently, transformers and self-supervised learning have triggered a global competition to train gigantic models on Internet-scale datasets, with massive computational resources. This thesis deals with large-scale 3D point cloud analysis and adopts a different approach focused on efficiency. We introduce methods which improve several aspects of the state-of-the-art: faster training, fewer parameters, smaller compute or memory footprint, and better utilization of realistically-available data. In doing so, we strive to devise solutions towards a more frugal and accessible Artificial Intelligence (AI).

We first introduce a 3D semantic segmentation model that combines the efficiency of superpoint-based methods with the expressivity of transformers. We build a hierarchical data representation which allows us to drastically accelerate the parsing of large 3D point clouds. Our network proves to match or even surpass state-of-the-art approaches on a range of sensors and acquisition environments, while boasting orders of magnitude fewer parameters, with faster training and inference.

We then build on this framework to tackle panoptic segmentation of large-scale 3D point clouds. Existing instance and panoptic segmentation methods do not scale well to large scene with numerous objects because the computation of their loss function implies a costly matching step between true and predicted instances. Instead, we frame this task as a scalable graph clustering problem, which a small network is trained to address from local objectives only, without computing the actual object instances at train time. Our lightweight model can process ten-million-point scenes at once on a single GPU in a few seconds, opening the door to 3D panoptic segmentation at unprecedented scales.

Finally, we propose to exploit the complementarity of image and point cloud modalities to enhance 3D scene understanding. We place ourselves in a realistic acquisition setting where multiple arbitrarily-located images observe the same scene, with potential occlusions. Unlike previous 2D-3D fusion approaches, we *learn* to select information from various views of the same object based on their respective observation conditions: camera-to-object distance, occlusion rate, optical distortion, etc. Our efficient implementation achieves state-of-the-art results both in indoor and outdoor settings, with minimal requirements: raw point clouds, arbitrarily-positioned images, and their cameras poses.

Overall, this thesis upholds the principle that for settings with limited data availability, exploiting the structure of the problem unlocks both efficient and performant architectures.

Résumé

Au cours de la dernière décennie, l'apprentissage profond a fait progresser l'analyse de texte, d'image, d'audio et de vidéo. Plus récemment, les *transformers* et l'apprentissage auto-supervisé ont déclenché une compétition généralisée visant à entraîner des modèles gigantesques sur d'immenses jeux de données, au moyen d'énormes ressources de calcul. Cette thèse porte sur l'analyse de nuages de points 3D à grande échelle et adopte une approche différente centrée sur l'efficacité. Nous introduisons des méthodes qui améliorent plusieurs aspects de l'état de l'art : entraînement plus rapide, moins de paramètres, coût de calcul plus faible, plus économe en mémoire et meilleure utilisation des données disponibles de manière réaliste. Ce faisant, nous nous efforçons de concevoir des solutions en vue d'une Intelligence Artificielle (IA) plus sobre et plus accessible.

Nous introduisons d'abord un modèle de segmentation sémantique 3D qui combine l'efficacité des méthodes basées superpoints avec l'expressivité des *transformers*. Nous construisons une représentation hiérarchique des données qui nous permet d'accélérer considérablement l'analyse de grands nuages de points 3D. Notre réseau se révèle égal, voire surpasser, les approches de pointe sur une gamme de capteurs et d'environnements d'acquisition, tout en réduisant le nombre de paramètres et le temps d'entraînement de un à deux ordres de grandeur.

Nous étendons ensuite ce cadre à la segmentation panoptique de nuages de points à grande échelle. Les méthodes existantes de segmentation d'instance et de segmentation panoptique ne sont pas adaptées aux grandes scènes comportant de nombreux objets, car le calcul de leur fonction de coût implique une étape fastidieuse d'appariement entre les instances réelles et prédites. Au lieu de cela, nous formulons cette tâche comme un problème de *clustering* de graphe, qu'un petit réseau est entraîné pour résoudre à partir d'objectifs locaux uniquement, sans nécessiter le calcul d'instances durant l'entraînement. Notre modèle peut traiter des scènes de dix millions de points à la fois sur un seul GPU en quelques secondes, ouvrant la voie à la segmentation panoptique 3D à des échelles sans précédent.

Enfin, nous proposons d'exploiter la complémentarité des modalités image et nuage de points pour améliorer l'analyse de scènes 3D. Nous nous plaçons dans un cadre d'acquisition réaliste, où plusieurs images arbitrairement positionnées observent la même scène, avec de potentielles occultations. Contrairement aux approches existantes de fusion 2D-3D, nous *apprenons* à sélectionner des informations à partir de différentes vues du même objet en fonction de leurs conditions d'observation respectives : distance caméra-objet, taux d'occultation, distorsion optique, etc. Notre implémentation efficace atteint l'état de l'art tant pour des scènes d'intérieur que d'extérieur, avec des exigences minimales : nuages de points bruts, images positionnées de manière

arbitraire et les poses de leurs caméras.

Dans l'ensemble, cette thèse soutient le principe que, dans des régimes où les données sont rares, exploiter la structure du problème permet de développer des architectures à la fois efficaces et performantes.

Acknowledgements

As this PhD nears its end, I have a lot of gratitude to express.

I am very grateful to my advisor Loïc Landrieu for making a researcher out of me. I have seen myself grow as the past 3 years flew by, and I have no one to thank more for this than him.

I would like to thank people whom I have worked closely with. Bruno Vallet for his trust, guidance, and support, always keeping in mind the potential applications of our work. Hugo Raguet for making half my thesis possible with awesome ideas and open code.

I am thankful to Clément Mallet and Ana-Maria Raimond from the LASTIG, and Philippe Calvez and Fabrice Boudaud from the CRIGEN, for welcoming me in their respective labs. This PhD was made possible thanks to funding from the ENGIE Lab CRIGEN, the IGN LASTIG lab, and the ANR project READY3D ANR-19-CE23-0007. A large portion of my work was carried out using HPC resources of IDRIS under the allocation AD011013388R1 made by GENCI.

My stay at IGN would never have been the same without all the great people I met there. I must thank Romain and Yanis for making our office the liveliest during the pandemic. Emile for his generosity, animated chats, and honey waffles. Vivien and Raphael for letting me join their arkose-sandwich sessions and all ensuing fun. Ewelina and Valerio for their constant warmth and kindness. The IGN restaurant's lunches would not have been as tasty without the company of all my labmates, whom will recognize themselves here and forgive my non-exhaustive list.

My occasional visits at CRIGEN were always a pleasure. Thanks to Marcos for his swift help with GPU servers and open-hearted discussions, and thanks Dmitriy and Irene for their enthusiasm for my projects.

Doing a PhD has been a lot of fun to me, I heard myself claim in multiple occasions: "I wish it would never end... only with a little salary raise each year !" I am grateful to the many people who have influenced the meandering path that led me here today. MLF for being the first to ever mention "machine learning" and "neural networks" to me. Andrew Ng and Sebastian Thrun for introducing me to all the fun stuff. Simon, Antoine, Hugo L., Alexis F., and Alex G. for helping me see how much I wanted to be a researcher and that there is no age limit to do a PhD.

None of the things I have accomplished in this thesis would have been possible without the rest of the scientific community. I want to thank all the (double-blind) reviewers and the members of my jury for taking the time to assess and

challenge my work for the sake of science. I feel very lucky to evolve in an environment where open science occupies such a central place. I would probably be knapping flint right now without the amazing, freely-available work of developers and maintainers of Linux, Python, Conda, CUDA, PyTorch, PyTorch Lightning, Torch-Points3D, PyTorch Geometric, Git, Latex, Overleaf, Plotly, Weights & Biases. I must also pay my respect to Yannick Kilcher, Tim Scarfe, Kurzgesagt, and Veritasium for their awesome YouTube channels which have been comforting shelters for my darkest procrastination times.

Obviously, I would not be where I am today without the support of people I hold dear. I thank my parents and sisters for supporting me in the foggy endeavor that this thesis must have been for them, for their unwavering trust in my ability to achieve what I set out to do, and for the safe haven they have always represented. For all the awesome memories of the past 3 years beyond endless lines of python and latex code, for lending an ear to my occasional complaints or excitement for ideas they probably feigned interest in: Simon, Antoine, Guichem, Vince, Thomas D., Marie, Anne, Guishaume, Val, Sam, Chloé, Hugo L., Ysé, Alexis F., Naila, Alban, thank you.

Finally, I am deeply grateful to Nathalie, for adding joy to my ups and laughter to my downs, for helping me slow down when life goes too fast, and enjoy it the rest of the time, for bringing me to Paris and following me to Zurich. My career as a researcher is a success, for I have already found something of great value.

My ultimate thanks go to the inventors of decaf, Dinausorus and Gaviscon, which make up for an ideal, carefully-balanced, pre-deadline intake for a thirty-something PhD student.

Contents

1	Introduction	1
1.1	A Glance at 3D Deep Learning	3
1.2	Motivations	11
1.3	Challenges	18
1.4	Contributions	22
1.5	Publications and Research Activities	24
1.6	Outline	26
2	Related Work	29
2.1	3D Deep Learning	30
2.2	Efficient Point Cloud Analysis	35
2.3	Superpoint-Based Learning	39
2.4	Leveraging Images for 3D Understanding	43
3	Superpoint Transformer	45
3.1	Introduction	47
3.2	Method	49
3.3	Experiments	61
3.4	Conclusion	71
4	Superpoint Graph Clustering	75
4.1	Introduction	77
4.2	Related Work	79
4.3	Method	81
4.4	Experiments	88
4.5	Conclusion	102
5	DeepViewAgg	105
5.1	Introduction	107
5.2	Related Work	109
5.3	Method	112
5.4	Experiments	123
5.5	Conclusion	134

6 Conclusion	139
6.1 Contributions	140
6.2 Perspectives	141
Bibliography	145
List of Figures	170
List of Tables	172
 Appendices	
Appendix A Generalities	175
A-1 Dataset Colormaps	175
Appendix B Superpoint Transformer	177
B-1 Interactive Visualization	178
B-2 Source Code	178
B-3 Influence of Handcrafted Features	179
B-4 Details on Hierarchical Partitions	181
B-5 Parameterizing the Partition	183
B-6 Implementation Details	183
B-7 Detailed Results	185
Appendix C Superpoint Graph Clustering	189
C-1 Interactive Visualization	189
C-2 Source Code	190
C-3 Implementation Details	190
C-4 Detailed Results	191
Appendix D DeepViewAgg	197
D-1 Interactive Visualization and Code	197
D-2 Fusion schemes	198
D-3 Dynamic-Size Image-Batching	199
D-4 Implementation Details	200
D-5 S3DIS Adjustment	202
D-6 Detailed Results	202
Annexe E Résumé Long en Français	207
E-1 Introduction	207
E-2 État de l'Art	211
E-3 Segmentation Sémantique Efficace à Grande Echelle	214
E-4 Segmentation Panoptique Efficace à Grande Echelle	216
E-5 Apprentissage sur Nuages de Points et Images Arbitrairement Localisées	217
E-6 Conclusion	218

Chapter 1

Introduction



Figure 1.1 – **Large-Scale 3D Scene Understanding.** We develop methods for the efficient semantic analysis of large 3D point clouds. Other methods analyze indoor 3D scenes room by room, ours can consider hundred-room buildings at once, in a few seconds on a single GPU.

3D point cloud processing finds applications in disciplines as diverse as environment monitoring [289, 339, 116], city planning [343, 324, 153], and autonomous navigation [194, 7]. These domains benefit from the acquisition of large amounts of 3D data and its automated analysis. As both acquisition technologies and computer vision have seen recent advances, deep learning on 3D point clouds has become a promising research direction.

Of late, the prevailing recipe for deep learning on text or images consists in training gigantic models on extensive datasets using ever-growing computational resources. This practice is motivated by the observation that increasing model and dataset sizes leads to gains in performance, albeit with diminishing returns. However, unlike its image processing counterpart, the 3D community does not benefit from web-scale open annotated datasets to train on, which currently hampers the emergence of very large 3D models. To illustrate, the largest open 3D dataset to date Objaverse-XL [69, 68] contains 10 million 3D objects collected from the Web, while its image counterpart LAION-5B [279] encompasses 5 billion images. In addition, this “bigger is better” trend comes at a high energy cost and progressively excludes practitioners with limited hardware and data resources.

We propose a different stance and call for more sober approaches in the context of 3D point cloud analysis. Specifically, we seek efficient methods accessible to researchers and practitioners alike. These methods should scale to large 3D scenes, without sacrificing performance or efficiency. They will preferably involve small models, with fast training and reasonable hardware requirements. Besides, we favor solutions evaluated on publicly available datasets, providing open-source code for easy reproducibility. In this thesis, we present 3 works that follow these principles. We propose scalable methods for 3D point cloud analysis which are compact and resource efficient, and a multimodal approach capable of leveraging arbitrarily localized images to improve 3D scene parsing.

In this introductory chapter, we first situate this thesis in the context of a recent 3D deep learning history in Section 1.1. From there, we present in Section 1.2 our motivations for working on the efficient analysis of large 3D point clouds, and the corresponding challenges that stand up against our goals in Section 1.3. Next, we present in Section 1.4 the main contributions of this thesis and summarize our scientific publications and research activities in Section 1.5. Finally, Section 1.6 summarizes the outline of the present document.

Prerequisites. This thesis assumes that the reader is familiar with deep learning, and deep learning for computer vision, in particular. For an introduction to these concepts, we suggest referring to Goodfellow *et al.* [100].

1.1 A Glance at 3D Deep Learning

First, we provide contextual background on the past 10 years of deep learning research and how they have shaped the field of 3D deep learning;

from the recent supremacy of artificial neural networks to the wide adoption of Transformer architectures and the revolution of self-supervised learning.

1.1.1 The Rise of Large Neural Networks

Deep learning is a subfield of Machine Learning, which focuses on training *deep* artificial neural networks. Contrary to other algorithms in the machine learning toolbox [315, 23, 126], artificial neural networks do not need carefully engineered input features, but learn to construct useful feature representations directly from raw data [186]. Inspired first by the study of biological neural networks [218], early work on artificial neural networks dates back to the mid-20th century [271]. However, as shown in Figure 1.2, deep learning only gained its current popularity in the early 2010s, when advances in hardware and the availability of large public datasets allowed deep neural network training that outperformed all previous approaches in speech recognition [64], text processing [60], and image classification [169]. From then on, deep learning helped push the boundaries in research fields as diverse as speech-to-text [105] and text-to-speech [241] translation, video classification [160], image generation [101], playing Go [286], or protein folding prediction [155]. Important breakthroughs came from methods that enabled the training of deeper networks, such as Rectified Linear Unit (ReLU) [5], batch normalization [143], or residual connections [118].

Although artificial neural networks do not account for all the complexity and mechanisms of biological neural networks, deep learning has long been influenced by neurosciences [360]. In particular, the study of neuroplasticity shows that, when a person loses one sense during childhood, brain cells that used to be dedicated to processing the associated input signals are progressively taken over by other senses, indiscriminately [59, 274]. Going even further, the human brain can learn to process and interpret new sensory inputs such

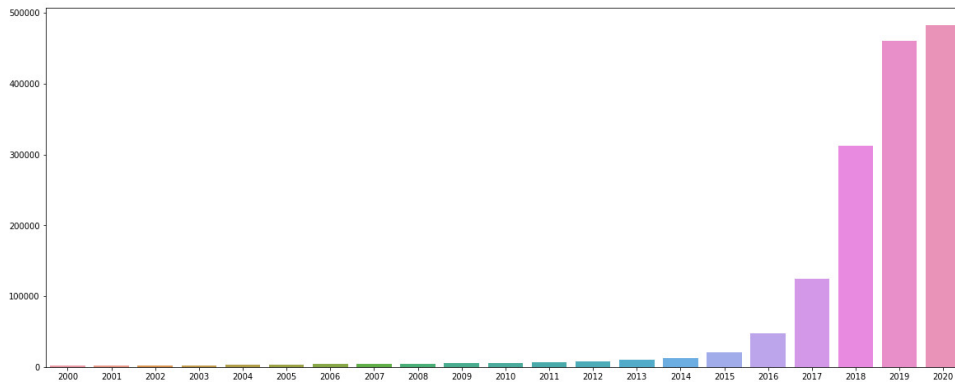


Figure 1.2 – **The Rising Popularity Deep Learning.** Historic occurrence of “deep learning” in academic articles on Google Scholar between 2000 and 2020.

as electro-tactile signals encoding a camera input sent to the tongue or the skin [77]. These findings suggest the possibility of a learning algorithm for training neural networks to process any input sensory signal, without priors on its structure. This underlying idea has been encouraging deep learning research towards a unified, input-agnostic learning architecture [147].

For a while, deep learning models mostly relied on modality- and task-specific building blocks with relatively strong inductive priors. Inductive priors [102] (or inductive biases) are assumptions introduced in a learning algorithm to predict outputs for previously unseen inputs. Simply put, these are design choices which force the algorithm to learn one specific pattern over another. There are many ways to encode such biases in an artificial neural network, among which regularization strategies [21, 290, 170], architectural restrictions [118, 210, 347], parameter sharing [127, 249], training recipes [125, 81, 72, 51], or invariance or equivariance to known transformations [184, 264, 353, 169, 87]. While beneficial for practical use cases, the reliance on inductive biases tailored to specific tasks and modalities does not align with the quest for a unified deep learning algorithm.

1.1.2 Transformers Take Over

In 2017, the introduction of the transformer architecture [313] for neural machine translation significantly impacted the Natural Language Processing (NLP) community. Unlike a Recurrent Neural Network (RNN), which iteratively processes elements of a sequence, the transformer constructs a representation for each element based on its surrounding context, in parallel. This approach outperforms previous methods, while making fewer assumptions, hence having fewer inductive priors. Soon, transformer-based architectures permeated other fields, such as image [76], audio [99], speech [75], or video [19] processing, replacing domain-specific building blocks with more versatile self-attention modules. Pushing this idea further, Perceiver [147] proposed a modality-agnostic transformer architecture to process image, audio, video, and point clouds.

While this may have been a bitter lesson for researchers attempting to instill domain expertise into their models [295], today’s deep learning community seems to largely share the belief that larger models with as little human-engineered inductive priors are always better, if trained end-to-end on sufficiently large datasets. This strategy is not suitable for 3D computer vision, whose publicly available datasets are smaller, as will be discussed in Section 1.2.2. In this thesis, we uphold the principle that in data-scarce regimes, exploiting the structure of the problem to design strong inductive priors is key to achieving good performance.

1.1.3 The Revolution of Self-Supervised Learning

Humans and animals are able to learn representations about the world from multiple sensory inputs at once without explicit supervision and can generalize concepts from very few examples [43, 176, 216]. However, deep learning models trained with supervised learning require large amounts of

annotated data to address a single task and tend to poorly generalize to other domains or tasks. For instance, training an image classification model requires a collection of images, each tagged with a class label. The vast majority of available data are gathered from the Internet in the form of text, images, and videos. Yet, these do not come with explicit task-specific annotations for models to learn from, and producing such annotations or labels is costly and labor-intensive. Thus, there is a need for methods capable of learning generalizable representations from readily accessible unlabeled data.

Arguably different from evolutionary and natural learning mechanisms, self-supervised learning [183] has recently shown promising results for building generalizable concepts from large amounts of raw data. In self-supervised learning, the model is trained on unlabeled data to address a *pretext task* designed to provide a supervision signal, in the hope of learning generic representations expressive enough to be useful for downstream tasks. The learned representations can then be used as a starting point for *fine-tuning* a model on a task for which few annotations are available. Pretext tasks for self-supervised learning usually rely on a priori knowledge of the input data structure and are designed to encourage the model to learn semantic representations of the data. For example, in image processing, relevant pretext tasks may consist of predicting the relative position of two image patches [74], the order of a sequence of images [223], or the rotation angle of an image [97] as presented in Figure 1.3. In text processing, a common pretext task is to predict masked words from a sentence [258, 72]. Of the many self-supervised learning pretext tasks in the literature, two predominant families emerge: masked modeling, in which models aim to reconstruct hidden portions of an input signal, and contrastive learning, in which models are trained to map two distinct views of the same input to similar representations.

Self-supervised learning has been central to several breakthroughs in AI

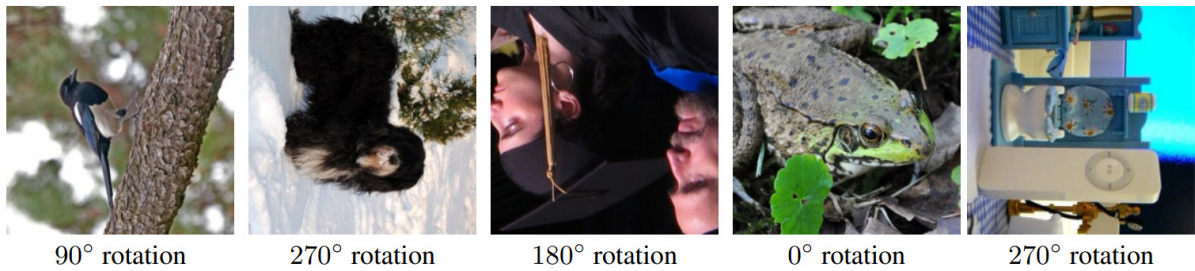


Figure 1.3 – **Learning Image Rotations.** The self-supervised learning task proposed in RotNet [97] is the following. Given an image rotated by a random multiple of 90° , the model is tasked with learning to recover the rotation angle. The authors show that addressing this task requires learning meaningful semantic representations of the image content: the sky is up, the ground is down, eyes are above the nose which is above the mouth, etc. Source: [97]

research since 2020, covering domains as diverse as language [258, 72, 259, 32, 304], image [51, 103, 106, 45, 17, 121], audio [204], video [277], and time series [330] processing. In computer vision, self-supervised representations have been able to match and even surpass representations learned in a fully-supervised fashion on image classification [71], provided that model and pretraining datasets are sufficiently large [120, 45, 121]. Furthermore, the contrastive framework has proven valuable for aligning representations across multiple modalities without explicit supervision, particularly exemplified in the domain of image-text alignment [260, 262, 236, 8, 326]. These notable advances can be attributed to the recent combination of transformer-based models, vast web-scale datasets, and massive computational resources. In this context, the AI community has seen the appearance of loosely defined *foundation models* [22]: large networks pretrained on extensive datasets and serving as the basis for various downstream tasks with little to no supervision. While crucial in text and image processing tasks, training such models demands colossal computational resources and extensive datasets that only a handful of private actors such as OpenAI, Meta, or Google currently have. As an example, to train their GPT-3 [32] language model, OpenAI used 1024 V100 GPUs for 95 GPU-hours and spent \$4.6M in compute alone [230]. Nevertheless, the

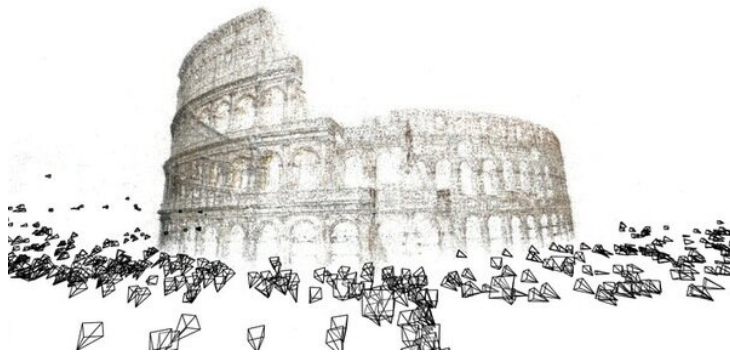
open-source community has been actively pushing to share these essential tools, contributing to their widespread availability and usability.

In this work, we explore 3D computer vision problems where dataset size and compute power tend to be more modest, hindering the emergence of foundation models for 3D understanding. For this reason, we choose to focus on efficient approaches trained with supervised learning.

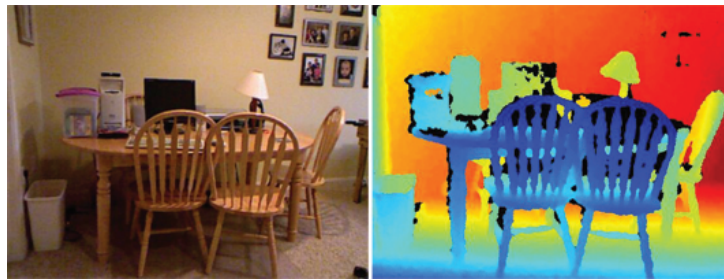
1.1.4 The 3D Data Boom

In recent years, the field of 3D computer vision has evolved significantly under the influence of 2D deep learning and the growing availability of 3D data. Indeed, advances in 3D sensing techniques like structure-from-motion (SFM) [303, 84, 114], time-of-flight cameras [240], structured-light cameras [354, 90], and LiDAR [150, 328] have led to the proliferation of affordable 3D acquisition devices capable of capturing the geometry of large scenes, as represented in Figure 1.4. Consequently, the increasing quantity of 3D data calls for scalable and efficient processing methods.

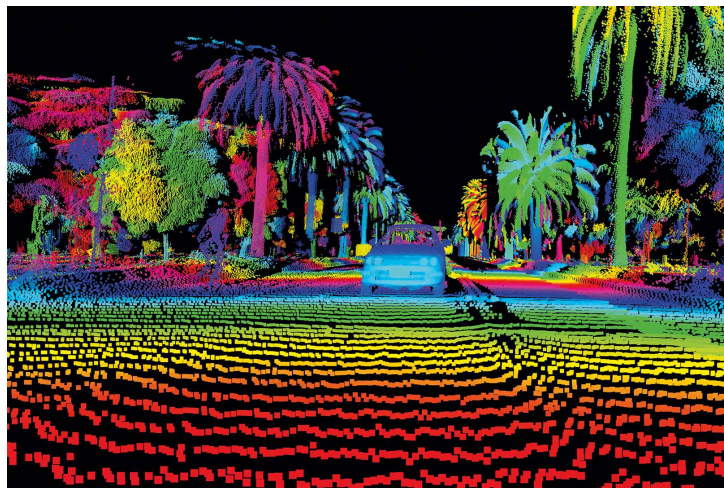
Various 3D deep learning architectures have been proposed for processing point clouds, which may be categorized by the data structure on which they operate. Point clouds can be treated as sets [252, 253], rendered images [25], converted to voxel grids [58], or graphs of groups of points [180]. For a detailed review of the field, the reader is referred to Chapter 2. Like other deep learning fields, the “transformer craze” has permeated the 3D community [359, 175]. More recently, self-supervised learning also made inroads into 3D computer vision, with promising avenues including pretraining for downstream 3D reconstruction and registration tasks [338, 356], 3D masked modeling [254], 2D-3D alignment [254, 130], and text-3D alignment [237, 247, 1] leveraging 2D as a pivot modality. However, self-supervised learning has yet to impact 3D semantic understanding as it did image processing.



(a) Structure From Motion [164]



(b) Depth Camera [285]



(c) LiDAR [94]

Figure 1.4 – **3D Point Cloud Acquisition Techniques.** Multiple acquisition methods exist to capture 3D scenes. Structure From Motion 1.4a relies on stereo vision to reconstruct a 3D scene from multiple images. Depth cameras 1.4b use structured light patterns to recover depth from a single image. LiDAR 1.4c emits laser pulses and measures the time of flight to infer depth.

Overall, progress in 3D deep learning has been partly driven by insights from 2D deep learning. However, its smaller community, limited datasets, and the challenges of its data structures have hindered the development of

foundation models for 3D computer vision (more details in Section 1.3).

The past decade has been eventful for deep learning research, and methods for processing data as diverse as text, image, video, and 3D point clouds have been converging towards more unified transformer-based architectures pretrained in a self-supervised manner. However, the success of these models generally hinges on the availability of enormous datasets and computational resources. In the following Section 1.2, we will discuss the limitations of these trends and our motivation for more efficient 3D deep learning methods.

1.2 Motivations

This thesis focuses on efficient methods for analyzing large 3D point clouds. In the section at hand, we first present domains of application for 3D point cloud processing. Next, we justify the need for efficient approaches, in light of the current trends in 3D deep learning, previously outlined in Section 1.1. We then introduce a simple typology for algorithmic efficiency which will help us navigate the rest of this work.

1.2.1 Automated 3D Scene Understanding

The semantic analysis of large 3D scenes finds applications in diverse domains, where both private and public actors resort to point clouds for distinct purposes. In the field of autonomous driving, car manufacturers employ 3D scene analysis to empower their automation systems [194], while other industrial companies use it for facility management [4, 11]. Meanwhile, public institutions rely on 3D point cloud processing for various objectives such as urban planning with digital twins [173, 324], natural disaster prevention [348, 145], or forest inventory [336, 156]. Figure 1.5 shows point cloud acquisitions with their potential application domains.

These real-world applications typically deal with data scales that exceed the processing capabilities of conventional academic methods. Our aim is to engineer techniques capable of effectively handling vast 3D scenes and extensive objects, such as long pipes in an industrial plant or entire buildings in a city [134, 288, 199]. In this process, we rely on deep learning, which, as mentioned in Section 1.1, has proven to be a powerful tool for computer vision.

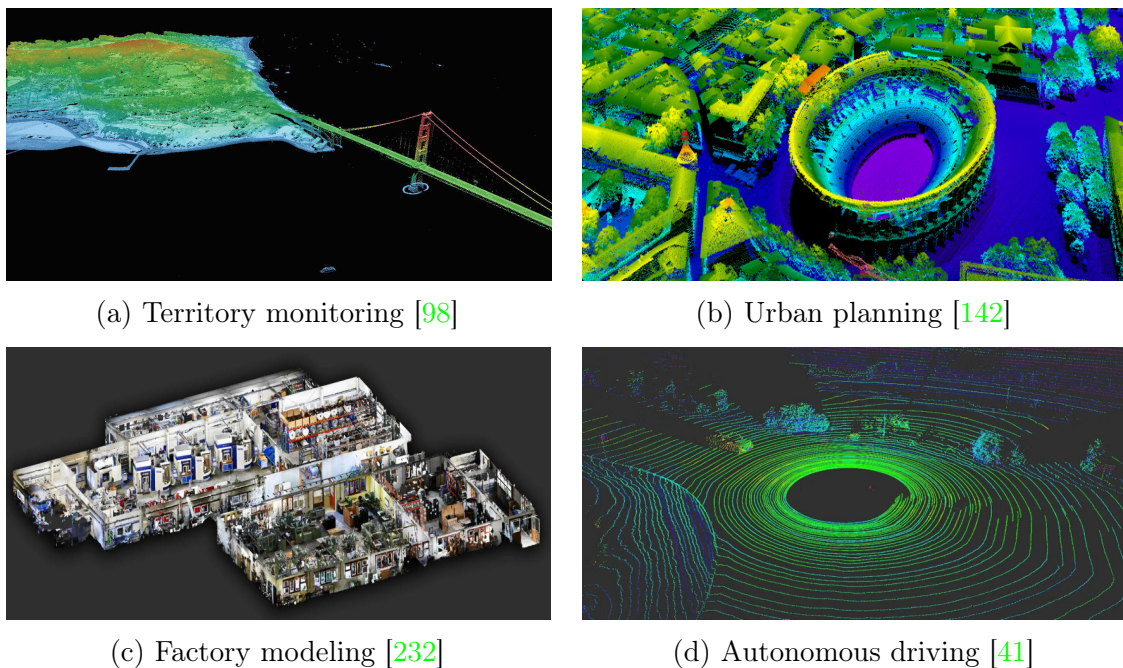


Figure 1.5 – **Domains of Application for 3D Point Cloud Processing.** Point clouds from diverse acquisition platforms may be used in a variety of applications.

1.2.2 Bigger Is Not Always Better

As presented in Section 1.1, the rapid progress of deep learning in the past decade has been marked by an increase in model size, dataset scale, and computational resources (see Figure 1.6). Although the performance of the model correlates with these three factors [159], the constant competition for more powerful models sometimes seems reduced to “the bigger the better” [67], with little consideration of the impacts of this ideology. Yet, the race for training

enormous transformer models with self-supervision on massive datasets is not without inconvenience, as we explain in this section.

Laborious Training. Despite their wide adoption, training a transformer-based model is notoriously difficult. First, transformers need more training data to match the performance of their CNN counterparts [76]. One explanation could be that their reduced inductive priors make transformers more “data-hungry”. This observation suggests that models with inductive priors may be necessary to achieve good performance with limited data. Second, optimizing numerous attention layers is challenging, requiring careful architecture design and hyperparameter tuning [235, 136, 202].

Likewise, in spite of its current popularity and exciting results, self-supervised learning remains challenging. To such extent that training a model to learn representations with self-supervision even calls for a “cook-book” [16]. In particular, designing a relevant pretext task in the hope that the learned features will be useful for other tasks of interest is not trivial. For example, pretext tasks suitable for 3D reconstruction or registration do not necessarily produce good representations for semantic understanding [338, 356].

Environmental Impact. Deep learning requires considerable computational resources in the form of GPU or TPU clusters. The construction and operation of these systems incur substantial energy costs, carbon emissions, and nonrenewable resource consumption. Keeping with our previously used example, the training of GPT-3 [32] on 1024 V100 GPUs emitted 552 tCO₂e [246], which matches the annual CO₂ emissions of 62 French citizens in 2021 [305]. For this reason, the rapid growth of deep learning has been characterized by a negative environmental impact, which is slow to be taken into account by the community [198, 332, 276]. In this context, the general trend to blindly

increase model size and compute budget for (often marginally) improved performance is not sustainable and calls for more sober methods.

Social Impact. The growing disparity in access to computational resources widens the gap between the research capabilities of a select few private entities (*e.g.* OpenAI, Google, Meta) and the broader scientific community. For illustration, reproducing a single training experiment of the “efficient foundation language model” LLaMA [304], would take more than two centuries on a single Nvidia V100 GPU. In addition, the achievements of models such as ChatGPT [242] and DALL-E [262] have sparked enthusiasm within society for their widespread distribution. However, the deployment of very large models is, again, challenged by the resources they demand. Therefore, the development of powerful yet efficient models is of prime importance for protecting open academic research and to permit their adoption by laboratories with limited compute or knowledge of deep learning, and by society at large.

Scaling 3D Models. Contrary to image and text, “large” academic point cloud datasets are relatively small. For comparison purposes, the ImageNet dataset [71], which is now considered relatively “small” within the realm of 2D computer vision, comprises roughly 10^4 times the number of pixels compared to the popular S3DIS [13] and ScanNet [66] point cloud datasets. For this reason, the model scaling strategies [159] used in other fields cannot be directly implemented on 3D deep learning, for lack of comparably large open datasets. Instead, we favor approaches designing smaller architectures with more inductive priors, which are better suited for “low-data” regimes.

To summarize, far from dismissing the power of large transformers and self-supervised learning on extensive datasets, we propose to focus on developing efficient 3D point cloud analysis methods matching the performance of larger

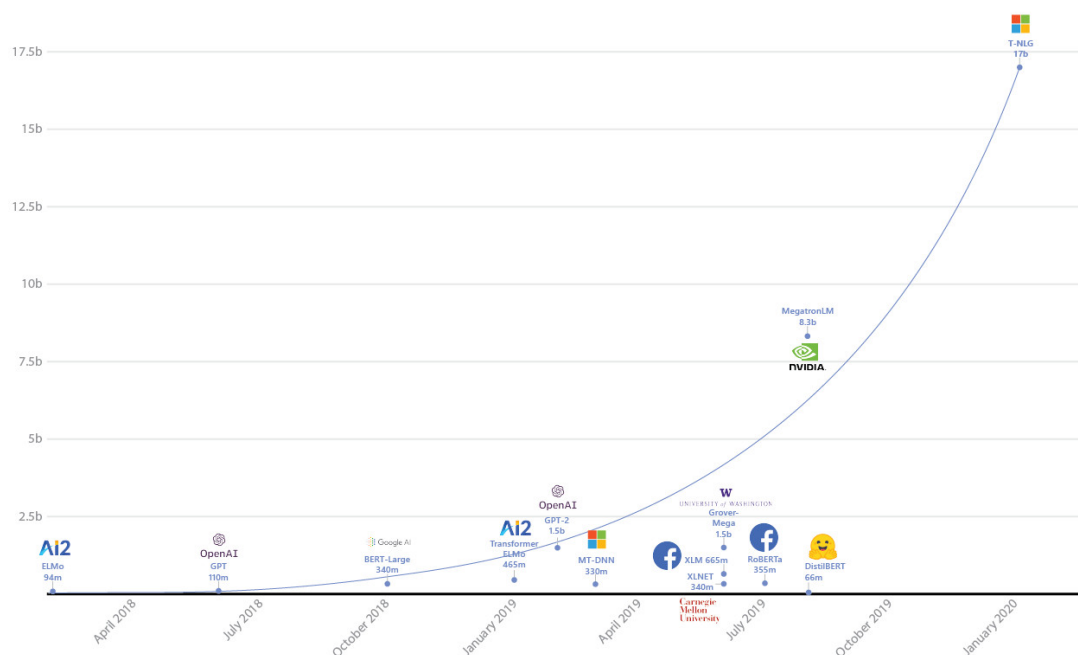


Figure 1.6 – **Size of Language Models Across Time.** The number of parameters of language models has been experiencing exponential growth over the past years. Although scaling model, dataset, and compute has proven to consistently improve performance [159], these large language models also have negative environmental and social repercussions. Figure taken from a 2020 Microsoft blog post [272]

models, due to environmental, social, and practical considerations.

1.2.3 Five Shades of Efficiency

We propose a simple typology of five properties characterizing the efficiency of a machine learning algorithm. This loose categorization encompasses five key dimensions of efficiency: compute, memory, hardware, data, and human labor.

Memory Efficiency. This aspect refers to the memory footprint of an algorithm. Unless specified otherwise, in the context of deep learning, memory efficiency relates to GPU or CPU memory usage, but it could also refer to disk occupancy. In the general deep learning setup, the GPU is used to perform the bulk of neural network computations, while CPU processes are tasked with asynchronously preparing data batches to be fed to the GPU.

Optimizing memory use may serve several purposes: running on a smaller less-expensive hardware, expanding the model size for higher expressivity, or increasing the batch size for faster training and better generalization. For instance, mixed precision training [221] reduces the memory footprint of a model by performing less precision sensitive operations on float16 rather than float32. Investigating memory efficiency is key to processing batches of large 3D point clouds whose size can easily exceed the memory capacity of a single GPU.

Compute Efficiency. We refer to the number of operations required to run an algorithm as its computational efficiency. In the context of deep learning, minimizing CPU and GPU operations will usually permit running the model faster, which can benefit time-sensitive applications, or simply reduce experimentation time. However, a more subtle relationship links compute and memory when training neural networks. In fact, the dominant method for training neural networks is the backpropagation algorithm [201], which requires storing the output of each operation in memory for the gradient calculation. For this reason, minimizing the number of operations in a neural network will also reduce memory usage during training. Similarly to memory efficiency, compute efficiency is pivotal to extracting meaningful local and contextual information from large 3D point clouds within reasonable time.

Hardware Efficiency. The hardware requirements of an algorithm. This can characterize the number of GPUs to run a model or the type of sensors necessary to produce the input data. The trend to increase the number of parameters in deep learning models in the hope of slightly increased performance comes with a need for larger and more numerous GPUs, making it harder for laboratories with limited resources to use these models. Minding hardware efficiency can contribute to making an algorithm more accessible,

reducing the cost of data acquisition, or leveraging readily available data. In the case of this thesis on 3D computer vision, we advocate robust solutions capable of processing data generated from a variety of acquisition techniques (see Section 1.1.4), with minimal hardware requirements.

Data Efficiency. Data efficiency characterizes how much training data a machine learning algorithm needs to achieve a given performance. Training a deep learning model requires especially large amounts of data [10, 159]. Unfortunately, data acquisition and annotation are costly and time-consuming, a fortiori for 3D point clouds. Hence, we prefer strategies that minimize the amount of annotated data needed to train a model. 3D acquisition platforms frequently produce both point clouds and images, which carry complementary representations of a scene. In Chapter 5, we will see that exploiting models trained on large image datasets is a data-efficient strategy for 3D point cloud analysis.

Human Efficiency. Several steps of a machine learning project may require human intervention: data acquisition, data annotation, experimentation, hyperparameter tuning, and model selection. We consider human time to be costly and precious, and generally favor human efficient strategies which aim at reducing the needed amount of human labor or time. As an example, transfer learning [30, 29] can be considered more human-efficient than training a model from scratch, as it reduces the amount of human annotations required to train a model, and tends to reduce the training time until convergence.

These categories are not mutually exclusive. Often, improving one aspect of the efficiency of a method will also impact another. For example, a technique to train with fewer data while maintaining performance will be both data efficient and human efficient. In this thesis, our proposed approaches explicitly

focus on memory, compute, and data efficiency.

In a nutshell, this work is motivated by the development of 3D deep learning techniques capable of processing real-world large-scale point clouds, while aiming for efficiency in terms of memory, compute, hardware, data, or human labor. Next, Section 1.3 will detail the challenges relative to this goal.

1.3 Challenges

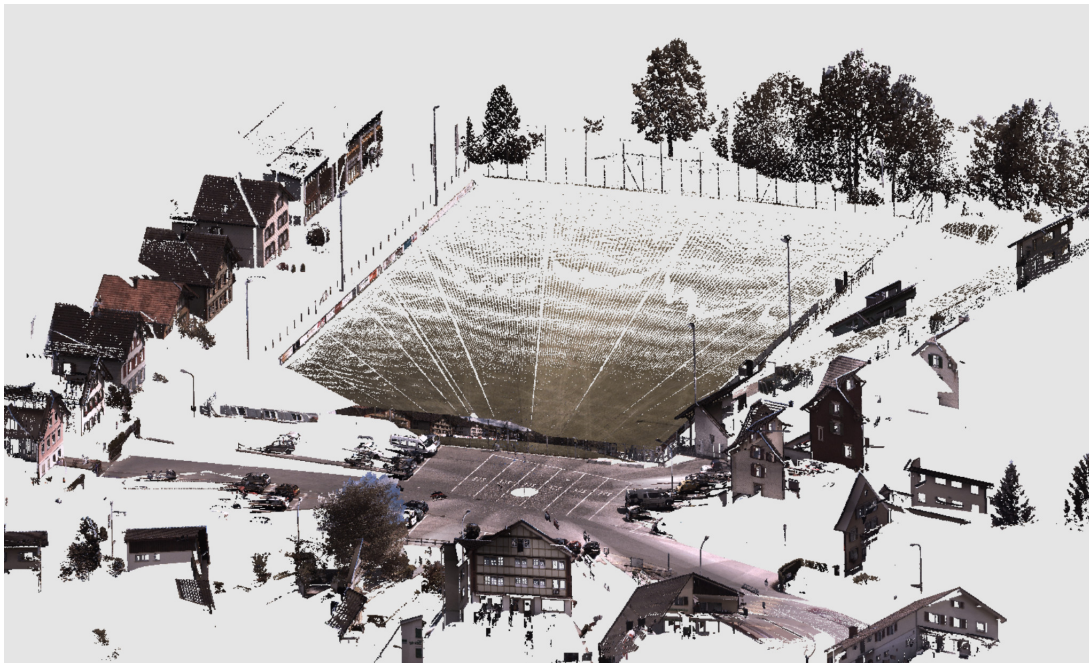


Figure 1.7 – **Challenges of 3D Point Cloud Analysis.** Some point clouds acquisition characteristics make 3D processing challenging. In this sample from the Semantic3D [111] dataset, we notice the absence of connectivity between points, occlusions, missing data, and radial sampling density.

Processing 3D point clouds comes with a set of challenges that we detail in this section. Point clouds imperfectly characterize 3D scenes.

Unlike meshes, which explicitly describe a surface, point clouds only represent a (potentially noisy) sampling of a real surface. Sparsely sampled regions may discard fine-grained geometric details [282]. The variety of acquisition techniques results in point clouds with distinct characteristics.

For example, photogrammetric point clouds [303] exhibit distance-dependent precision and present artifacts resulting from erroneous matching. Meanwhile, LiDAR sensors [150] are subject to reflections producing outliers. Common to all 3D acquisition methods, the constant angular resolution of the sensor induces a radial distribution of points; objects farther away from the sensor are sampled with fewer points. This phenomenon can be observed in Figure 1.7. Consequently, extracting meaningful information from point clouds calls for methods robust to uneven density, noisy point coordinates, occlusions, and acquisition artifacts.

The above “generic” challenges are inherent to 3D point clouds and are shared by all point cloud processing methods. In this thesis, we tackle three important 3D computer vision challenges, specific to our ambition to efficiently process large 3D scenes: extracting information at multiple scales, enriching point cloud representations with additional modalities, and 3D processing implementation.

1.3.1 Efficient Multiscale Reasoning

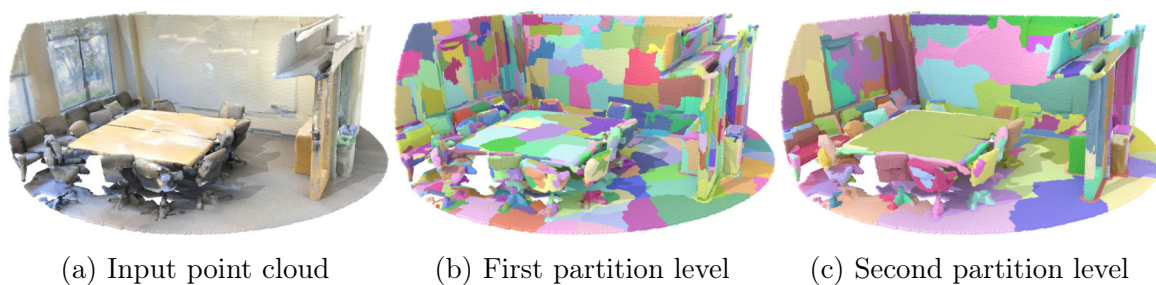


Figure 1.8 – **Hierarchical Representation.** Our method takes as input a point cloud (a) and computes its hierarchical partition into geometrically homogeneous superpoints at multiple scales: (b) and (c).

In image processing, the extraction of features at multiple scales provides rich representations of the content of an image [35, 36, 215]. High-resolution (high-frequency) features capture local details, while lower-resolution (low-

frequency) features capture contextual, long-range interactions in the image. Similarly, point cloud processing methods rely on features captured at different scales to characterize 3D shapes [253, 58, 301]. *state-of-the-art* 3D deep learning methods either rely on a hierarchy of arbitrary point [255, 175] or voxel [58] samplings of decreasing resolutions to capture local and contextual information. However, these methods are compute- and memory-intensive and do not scale to large scenes. For instance, KPConv [301] can only consume 2 m-radius crops of 3D scenes at once, and Stratified Transformer [175] requires four GPUs to process a single room of the S3DIS dataset [13]. Such approaches do not scale to scenarios that require both local geometric details and long-range interactions to be captured. For example, in an urban mapping [197] scenario, objects of interest may span a wide range of scales: from the traffic sign characterized by small geometric patterns and its global location on the street, to the large multi-story building, characterized by the aggregation of numerous smaller shapes grouped into one large concept.

This challenge has led us to develop in this thesis a hierarchical data representation, shown in Figure 1.8, that adapts to the geometric complexity of the scene, allowing for compute- and memory-efficient multiscale reasoning.

1.3.2 Efficient Multimodal Fusion

Fusing information from additional modalities such as images can improve the performance of 3D point cloud analysis. Indeed, these two modalities carry complementary information, as point clouds capture the geometry, while images capture the texture and context. Figure 1.9 illustrates this synergy between 3D point clouds and images.

One way of exploiting this complementarity is to colorize point clouds. Yet, such colorization is often a blackbox preprocessing [188, 83, 306] which discards a large portion of the dense textural and contextual information

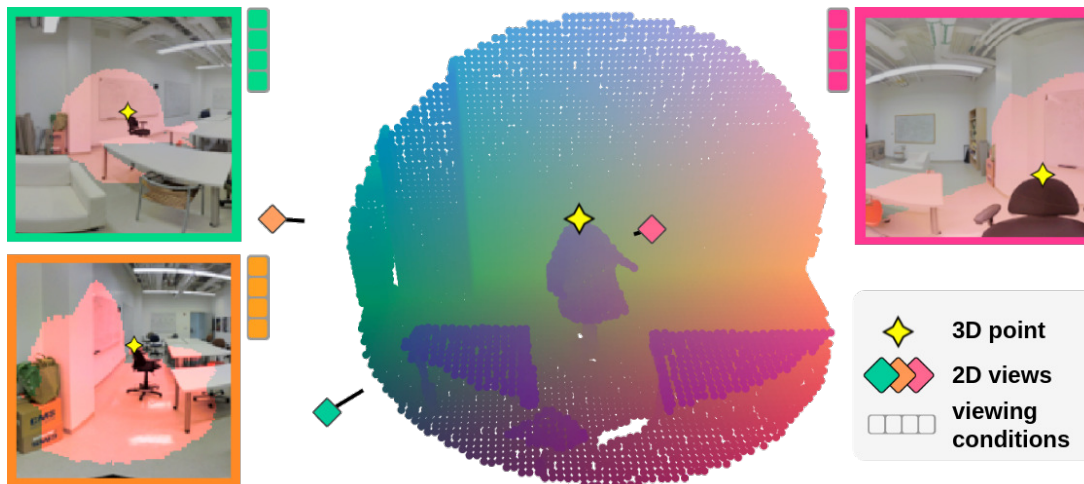


Figure 1.9 – **Multimodal Fusion.** Point clouds capture the geometry of the scene, while images capture textures and context. The pink halo in the images illustrates which pixels “see” a 3D point from the sample. This mapping is necessary to both point cloud colorization and 2D-3D learning. We seek methods capable of building such mappings and fusing information from both point clouds and arbitrarily-posed images, with minimal sensor and preprocessing requirements.

carried by the images. Methods capable of directly processing point clouds and images should have access to more information about the scene.

Recent multimodal 2D-3D approaches [65, 149, 135] propose using a 3D network to process point clouds, and a 2D network to extract image features that are projected onto the point cloud, outperforming methods operating on colorized point clouds. However, we identify two limitations of these approaches. First, their reliance on specific depth cameras or compute-intensive meshing operations to map points to pixels does not align with our search for hardware and compute-efficient solutions. Second, they aggregate features from all views of the same object, without taking their observation conditions into account.

In this thesis, we propose a method for efficiently extracting and fusing features from point clouds and an arbitrary number of images in the wild. Our method does not require any specific sensor or meshing, only raw point clouds, images, and their poses.

1.3.3 Efficient Implementation

Efficiently processing 3D point clouds requires careful implementation. Unlike 2D images whose fixed-size matrix format conveniently adapts to hardware and tensor reasoning, 3D point clouds do not capture local connectivity, are often unordered, and exhibit varying sizes. Basic operations such as nearest-neighbor search, trivial for image pixels, can quickly become compute or memory bottlenecks in 3D. As a result, although not typically advertised in the literature, efficient 3D processing necessitates a non-negligible implementation effort. For example, by developing an elegant sparse voxel convolution library, MinkowskiNet [58] paved the way for scaling voxel-based methods. Such endeavor was necessary to bypass the cubic complexity of dense voxel convolutions, but entailed the fastidious development of an entire library for sparse tensor calculus. Similarly, algorithmic engineering represents an important part of the work presented in this thesis.

To summarize, this section highlights the challenges to efficiently process large 3D point clouds. These challenges encompass efficient multiscale feature extraction, fusing information from point clouds and images, and implementation. Coming next, Section 1.4 details how our work addresses these challenges.

1.4 Contributions

This thesis introduces two main contributions for the efficient analysis of large-scale 3D scenes.

Lightweight 3D Analysis Our first main contribution is an efficient framework for point cloud semantic analysis, which can match and even surpass state-of-

the-art methods while benefitting from one to two orders of magnitude fewer parameters and significantly faster training and inference times. This achievement is based on two key ideas. First, different from competing methods operating on arbitrary point or voxel samplings, we partition point clouds into geometrically and radiometrically homogeneous *superpoints* organized in a hierarchical structure. Second, we use a streamlined transformer architecture to propagate information between adjacent superpoints and reason on the scene at each level of hierarchy.

By confining point- or grid-based reasoning to local feature extraction, our data structure drastically reduces the problem complexity, leading to significant gains in compute and memory efficiency. With limited training and inference times on a single consumer-grade GPU, our algorithm makes experimentation on 3D point clouds fast and accessible, proving to be both hardware and human-efficient. We validate our approach for semantic [267] and panoptic [268] segmentation, across multiple datasets, sensor technologies, and acquisition environments. Our method proves to be suitable for capturing long-range dependencies and scales to very large scenes without performance loss. In particular, our semantic segmentation architecture matches or outperforms competing methods on all benchmarks, with up to 200 times fewer parameters and 70 times faster training. For panoptic segmentation, we set a new state-of-the-art on all benchmarks and process of unprecedented size in a few seconds scenes on a single GPU.

Multimodal 2D-3D Analysis Our second main contribution is an architecture capable of jointly extracting information from both 3D point clouds and 2D images. In particular, our method learns to aggregate information from an arbitrary number of views of the same point, based on their viewing conditions, such as angle of view or distance to the sensor.

Unlike similar methods which rely on mesh reconstruction or depth cameras to map points with pixels, we only use raw point clouds and images with poses. Our sparse, parallelized implementation ensures that we can efficiently scale to large 3D scenes with an arbitrary number of images. In addition, by leveraging the knowledge learned from larger image datasets to improve 3D scene understanding without additional 3D annotations, our approach is data and human efficient.

To summarize, this thesis exploits structures in 3D point clouds and multi-modal data to devise deep learning architectures achieving high performance, while being efficient and scalable.

1.5 Publications and Research Activities





International Conferences. All the works presented in this thesis were published in three international computer vision conferences.

- Damien Robert, Bruno Vallet, Loic Landrieu. “Learning Multi-View Aggregation in the Wild for Large-Scale 3D Semantic Segmentation”. In: *CVPR* (2022), **Shortlisted for the Best Paper award**, top 0.4% submissions.
- Damien Robert, Hugo Raguet, Loic Landrieu. “Efficient 3D Semantic Segmentation With Superpoint Transformer”. In: *ICCV* (2023), top 26.8% submissions.
- Damien Robert, Hugo Raguet, Loic Landrieu. “Scalable 3D Panoptic Segmentation As Superpoint Graph Clustering”. In: *3DV* (2024), **Oral**, *pending acceptance rate*.

International Workshops and Schools. Some of the above publications were also presented at international workshops or schools.

- [Damien Robert](#), Bruno Vallet, Loic Landrieu. “Learning Multi-View Aggregation in the Wild for Large-Scale 3D Semantic Segmentation”. In: *ISPRS congress* (2022).
- [Damien Robert](#), Bruno Vallet, Loic Landrieu. “Learning Multi-View Aggregation in the Wild for Large-Scale 3D Semantic Segmentation”. In: *International Computer Vision Summer School* (2022).
- [Damien Robert](#), Hugo Raguet, Loic Landrieu. “Efficient 3D Semantic Segmentation With Superpoint Transformer”. In: *4th Visual Inductive Priors for Data-Efficient Deep Learning Workshop, ICCV* (2023).

Open-Source Code. All code produced during this thesis has been publicly released on GitHub. These repositories allow the reproduction of the results communicated in our papers, provide tools for 3D vision, or research in general.

 drprojects/DeepViewAgg	205 ★ 23 📄
 drprojects/superpoint_transformer	202 ★ 36 📄
 drprojects/point_geometric_features	21 ★ 3 📄
 drprojects/nora	15 ★ 0 📄

Dissemination. I was invited to present my work to the following research groups.

2021 May	French Mapping Agency - IGN
2022 Jan	GDR ISIS
2022 May	École Polytechnique - LIX lab
2022 June	École des Ponts - IMAGINE lab
2022 Nov	Federal Agency for Cartography and Geodesy - BKG
2023 May	Samp

- 2023 May** [Valeo.ai](#)
- 2023 May** [University of Zurich - EcoVision lab](#)
- 2023 June** [ENGIE Lab CRIGEN - CSAI](#)
- 2023 Sept** [ETH Zurich - Photogrammetry and Remote Sensing group](#)
- 2023 Sept** [ETH Zurich - Vision and Learning Group](#)
- 2023 Oct** [National Land Survey of Finland - Maanmittauslaitos](#)
- 2023 Nov** [École des Ponts - IMAGINE lab](#)

Teaching. During this thesis, I carried out the following teaching missions.

- 2020** Course on *Deep Learning for Computer Vision* at [École Polytechnique](#) for 1st year Master students (12 hours)
- 2022** Course on *Deep Learning for Remote Sensing* at [ENSG](#) for 2nd year Master students (9 hours)
- 2022** Tutorial on *3D Deep Learning* at [ENGIE Lab CRIGEN - CSAI](#) for researchers (1 day)
- 2022** Tutorial on *3D Deep Learning for Remote Sensing* at the [XXIV ISPRS congress](#) for researchers (1 day)
- 2023** Course on *Deep Learning for Remote Sensing* at [ENSG](#) for 2nd year Master students (13 hours)

1.6 Outline

This thesis is organized as follows.

Chapter 1: Introduction. We start by placing this thesis in the larger context of deep learning for 2D and 3D computer vision. Then we discuss our motivations behind this work and the challenges that stand in our way to

achieve our goals. Finally, we present the contributions of this thesis and outline the structure of the document.

Chapter 2: Related Work. We start by introducing the main families of 3D deep learning models. Then, we review existing strategies for efficient deep learning on images and point clouds. Next, we expand on superpoint-based methods for efficient 3D processing. Finally, we overview approaches for multimodal learning with point clouds.

Chapter 3: Efficient and Scalable 3D Semantic Segmentation. We present our superpoint-based transformer architecture to efficiently perform semantic segmentation on large-scale 3D scenes. This method relies on a fast algorithm for partitioning point clouds into a hierarchical superpoint structure, as well as a self-attention mechanism to learn the relationships between superpoints at multiple scales. We demonstrate *state-of-the-art* performance on three 3D semantic segmentation benchmarks with up to $200\times$ fewer parameters and up to $70\times$ faster training, compared to competing approaches.

Chapter 4: Efficient and Scalable 3D Panoptic Segmentation. We cast the 3D panoptic segmentation task as a scalable graph partitioning problem, which a small model can be trained to address based on local objectives only. Our framework circumvents several limitations of competing methods, such as computing the segmentation at training time, matching predicted and target instances, or hard-coded priors on the minimum or maximum number of objects in a scene. It can naturally be extended to the superpoint paradigm presented in Chapter 3, which allows for efficiently scaling to vast 3D scenes. We reach *state-of-the-art* performance on four 3D panoptic segmentation benchmarks and demonstrate the efficiency of our method in training and inference time.

Chapter 5: Learning From Point Clouds and Images in the Wild. We propose an end-to-end multi-view aggregation method for 3D semantic segmentation from images and point clouds. We reach *state-of-the-art* performance on two 3D semantic segmentation benchmarks without requiring point cloud colorization, meshing, or depth sensors: only point clouds, images, and their poses.

Chapter 6: Conclusion. We summarize the contributions of this thesis and discuss directions for future work on efficient 3D computer vision.

Chapter 2

Related Work

Our contributions for the efficient processing of large point clouds build upon several bodies of literature which we detail in the present chapter. First, we introduce in Section 2.1 the main families of 3D deep learning models. We then review in Section 2.2 existing strategies for efficient deep learning, with a final focus on point-cloud-specific methods. This leads us to expand, in Section 2.3, on the superpoint-based research directions that inspired our work in Chapter 3 and Chapter 4. Finally, Section 2.4 provides an overview of the approaches related to the multimodal learning framework proposed in Chapter 5.

2.1 3D Deep Learning

3D Deep Learning architectures can be broadly categorized according to the data representation they operate on: voxel-based, image-based, point-based. This section gives an overview of these categories and how each relates to 2D deep learning. Figure 2.1 provides a visual summary of the presented categories. For a more in-depth review of 3D deep learning literature, we refer the reader to [110].

Formally, in our setting, a *3D point cloud* refers to a collection of points defined by their (generally Cartesian) coordinates in three-dimensional space. These points are assumed to form a discrete, unordered, and potentially noisy sampling of a real 3D surface. Optionally, such points may be endowed with additional information such as color, normal vector, or intensity.

Image-Based Methods. Image-based or view-based approaches project 3D point clouds into multiple 2D views and analyze the resulting images with 2D neural networks. First introduced for shape classification by MVCNN [292], this strategy was extended to 3D object detection [52] and semantic segmentation of large scenes [25]. Image-based methods conveniently leverage

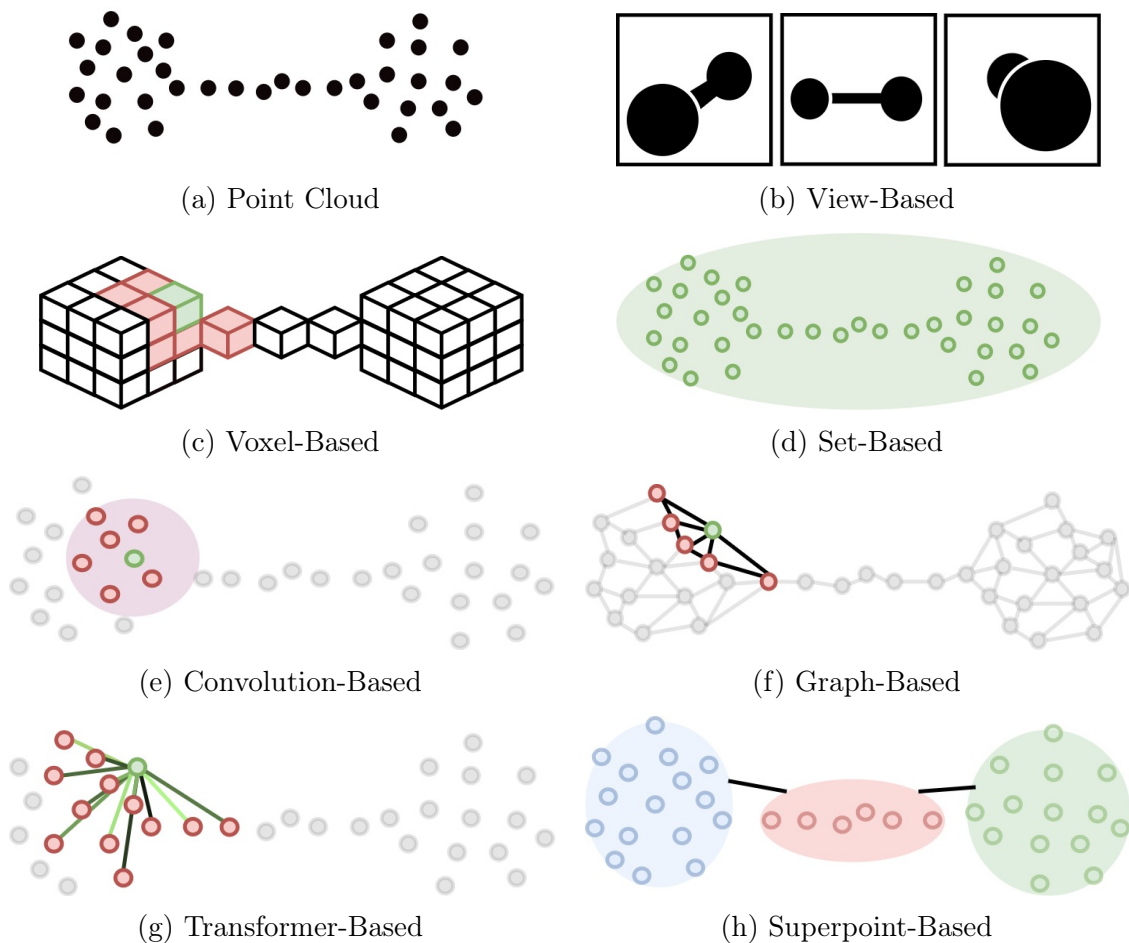


Figure 2.1 – **3D Deep Learning Methods**. A variety of methods have been proposed to extract features from point clouds 2.1a. Some approaches discard information by rendering 2D views of the points 2.1b, or discretizing the cloud in a voxel grid 2.1c. Other methods directly operate on unordered sets of points 2.1d, or generalize discrete 2D convolutions to continuous 3D space 2.1e. Alternatives use graph neural networks 2.1f or transformers 2.1g to reason on a local neighborhood. Closely related to our work, superpoint-based approaches 2.1h reason on a partition of the scene.

readily-available architectures and easier-to-annotate datasets from the 2D computer vision community. However, some limitations hinder the wide adoption of these methods: rendering occlusions when projecting points necessitates a costly surface reconstruction [25], aggregating information from multiple views of the same object is nontrivial [278, 318], and the projection operation inevitably discards information.

Voxel-Based Methods. Voxel-based (or volumetric) methods convert unstructured 3D point clouds to 3D voxel grids. This straightforward representation adapts successful 2D computer vision methods operating on pixel grids to 3D. VoxNet [217] introduces a 3D convolutional neural network for processing dense 3D grids, and OctNet [266] uses an octree to mitigate the memory footprint of dense voxel representations. SEGCloud [299] circumvents the memory problem by using convolutions on large voxels and a graphical model for subvoxel segmentation. Motivated by the sparsity of voxelized point clouds, MinkowskiNet [58] designs a library for sparse 3D convolutions on GPU, making it possible to process large scenes with volumetric methods. Recently, hybrid approaches [207, 297, 352] address the small-scale information loss inherent to voxelization. Although conceptually simple, voxel-based methods tend to be compute- and memory-intensive, and their three-dimensional kernels require more parameters than their 2D counterparts.

Point-based methods operate directly on the point cloud, without introducing explicit information loss by discretization or projection. This category can be further divided into set-based, convolution-based, graph-based, transformer-based, and superpoint-based methods.

Set-Based Methods. Unlike image-based and voxel-based methods that draw their inspiration from 2D computer vision to address point cloud processing, set-based methods are specifically designed to operate on unordered data such as point clouds. Deep Set [350] and PointNet [252] proposed similar pioneer models for encoding permutation-invariant sets of points. Inspired by multiscale encoders from image processing, PointNet++ [253] introduces an architecture for hierarchical reasoning on sets of increasing coarseness. Different from the max-pooling operation commonly used for downsampling

pixel grids, this model combines farthest point sampling (FPS), nearest neighbor search, and an aggregation function to progressively coarsen the point sets. Although set-based methods were among the first 3D deep learning algorithms in the literature, PointNeXt [255] recently achieved *state-of-the-art* performance by scaling PointNet++ with modern training recipes.

Convolution-Based Methods. Convolution-based methods seek to generalize discrete 2D convolutions to continuous 3D space [193, 24]. Contrary to pixels or voxels, where convolution kernels align with grid-based neighborhoods of constant size and shape, defining a convolution operator for points localized in a continuous space is not straightforward [31], due to the varying size and distribution of the neighborhood of points. KPConv [301] and ConvPoint [24], for instance, learn convolution kernels which can be placed at any position in a continuous space. Although mathematically elegant, convolution-based methods involve sensitive parameterization, and their computational cost hinders their scalability.

Graph-Based Methods. Graph-based methods treat individual points as graph vertices, connected based on their spatial proximity. Graph processing models are then used to extract features in the spatial [287, 327] or spectral [300] domain. The adjacency graph structure presents a sensible representation for analyzing point clouds and their underlying surface geometry. Nevertheless, graph-based methods are challenged by their restricted connectivity, sequential treatment, limited expressivity, and small receptive fields.

Transformer-Based Methods. The success of the Transformer [313] architecture in natural language processing and its adaptation to image analysis [76, 206] have inspired the development of transformer-based methods for point

clouds. These models can be seen as a generalization of graph-based approaches to dynamic graphs, where edge weights evolve depending on the node features, allowing for the adaptive aggregation of neighborhood information. 3D vision transformers have demonstrated strong performance on various point cloud analysis tasks [359] and the ability to capture long-range interactions [175]. Yet, the quadratic complexity of their self-attention scheme leads to high memory consumption when applied on dense point clouds, which limits their ability to process large scenes.

Superpoint-Based Methods. Scenes with many points can often be broken down into a much smaller set of simple shapes. This especially applies to anthropic objects that exhibit smooth, regular shapes. The surface of a house, for instance, is largely made of planes making up its walls, floor, ceiling, and roof. In addition, adjacent points sharing similar geometry and color often share the same semantic meaning. Hence, instead of reasoning on individual points, 3D scene analysis could be simplified to reasoning on much sparser shapes. Following this train of thought, superpoint-based methods partition point clouds into *superpoints*: sets of adjacent points with similar geometric and radiometric properties. Closely related to our work, Superpoint Graph [180] processes 3D scenes as graphs of superpoints with a graph neural network. Operating on superpoint partitions reduces the size of the point cloud parsing problem by several orders of magnitude, leading to efficient models with few parameters capable of processing extensive scenes. However, superpoint-based approaches rely on graph neural networks which, as previously mentioned, tend to lack in expressivity. A deeper introduction of these methods will be provided in Section 2.3.

The work presented in this thesis spans all three categories. Chapter 3 presents

an efficient and scalable model that inherits from transformer- and superpoint-based methods. Meanwhile, Chapter 5 addresses 2D-3D multimodal learning with inspiration from image- and voxel-based approaches.

2.2 Efficient Point Cloud Analysis

As 3D scans of real-world scenes can contain hundreds of millions of points, the efficiency of 3D analysis is critical. In this section, we first provide a high-level overview of research directions for efficient deep learning. Then, we introduce some works specific to efficient 3D point cloud analysis.

2.2.1 Efficient Deep Learning

Inspired by Menghani [220], we broadly categorize research directions pursuing efficient deep learning as follows: compression, training, AutoML, implementation, and architecture.

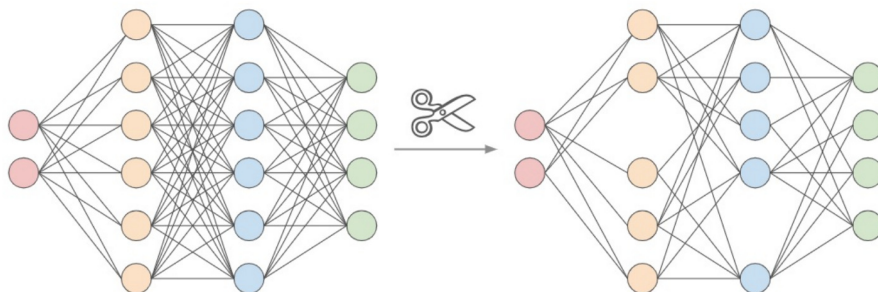


Figure 2.2 – **Pruning**. A simplified illustration of pruning weights (connections) and neurons (nodes) in a neural network comprising fully connected layers. Source: [220]

Compression. Compression techniques aim to reduce the size of a neural network, with a potential trade-off in performance. As illustrated in Figure 2.2, pruning methods search for non-essential neurons to sparsify the network [185, 115]. Quantization methods reduce the memory cost of neural networks by encoding their activations and weights with low-precision datatypes. This can

be done at inference time, by mapping high-precision values to finite sets of discrete values [167, 146, 57]. Alternatively, quantized networks can be trained directly with low precision [139, 190]. Meanwhile, low-rank compression [141] approximates the weight matrix of each layer by the product of two smaller matrices.

Training. Efficiency can be gained by improving the training process itself. Data augmentation can be seen as an efficient strategy to artificially increase the size of the data set [169, 293]. Distillation techniques [124] use a large, pretrained teacher model to supervise a smaller student network to be deployed in downstream applications. Low-rank adaptation [132] makes the fine-tuning of large models easier by using a low-rank approximation of the network weights. Departing from traditional supervised learning, weakly supervised [364] makes use of inexpensive or partial annotations. For instance, for the subfield of semi-supervised learning [345] only a fraction of data points have annotations, while multi-instance learning [42] only provides labels for bags of data points. Pushing even further, self-supervised learning [97, 51, 120, 121] relies on pretext tasks to learn useful representations from data without labels. Typical pretext tasks involve reconstructing the input from a corrupted version of itself [121], or pairing transformed versions of the same input [51].

AutoML. AutoML techniques propose to minimize human intervention in the model search process. Hyperparameter optimization [18, 224, 6] automates the exploration of hyperparameters for a given neural network architecture, while neural architecture search [78, 296] explores the design space of neural network architectures to find the most efficient one for a given task.

Implementation. The implementation of deep learning algorithms may lead to savings in computation or memory. For example, mixed precision training [221] only uses high-precision datatypes for sensitive operations, thus reducing the memory footprint of the network. By accumulating the gradients computed on smaller batches, microbatching [137] can train large models on limited memory. Differently, gradient checkpointing [50, 34] saves memory at training time by discarding intermediate activations in the forward pass and recomputing them during backpropagation.

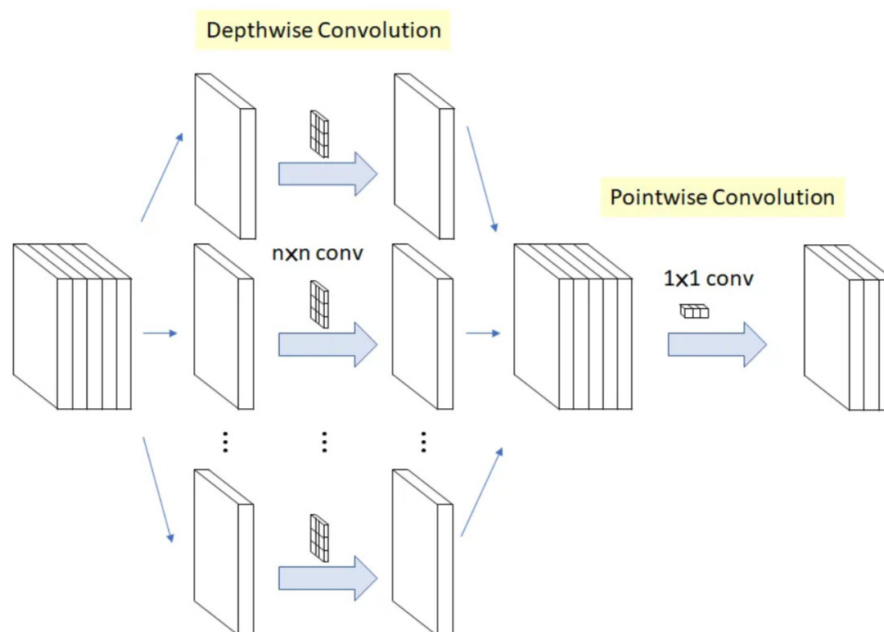


Figure 2.3 – **Depth-Separable Convolution.** By separating the traditional convolution operation into a per-channel convolution followed by a 1×1 channel-mixing operation, depth-separable convolution [56] reduce the number of parameters per convolutional layer. Source: [307]

Architecture. Designing new building blocks for neural networks can improve model efficiency. Often, introducing inductive priors in the architecture may reduce the number of parameters or operations. For example, by sharing parameters to enforce translation equivariance, convolutional neural networks [186] are more efficient than fully-connected networks for image processing. Similarly, atrous convolutions [48] assume some local regularity in

the features to increase the receptive field of convolutions without increasing the number of parameters. As shown in Figure 2.3, the depth-separable convolutional layers [56, 273] disentangle spatial and channel-wise correlations by factorizing standard convolutions, with fewer parameters. With the recent popularity of transformer architecture, multiple works have proposed efficient variants [298, 163, 325], while ConvNeXt [208] has shown that convolutional networks can still outperform their less efficient transformer counterparts when trained with similar recipes.

While the combination of several of these strategies certainly holds potential, our contribution mainly lies in proposing new architectures for efficient 3D deep learning.

2.2.2 Efficient Deep Learning on Point Clouds

Various architectures for efficiently processing 3D point clouds can be found in the literature. The seminal PointNet [252] and PointNet++ [253] models provide fast, parameter-efficient 3D analysis baselines. RandLANet [133] demonstrates that efficient sampling strategies can yield excellent results, by replacing the costly Farthest Point Sampling of PointNet++ with simple random sampling. Akin to ConvNeXt [208] mentioned above for images, PointNeXt [255] achieves state-of-the-art performance by updating PointNet++ [253] with modern training recipes, outperforming less efficient transformer-based architectures [359].

Other works explore efficient data representations for point cloud analysis. To circumvent the prohibitive memory and compute footprint of 3D convolutions on dense voxel grids, SparseConvNet [104] and MinkowskiNet [58] operate on sparse representations, conducting voxel-based models capable of processing large scenes while maintaining high resolution. More recently,

hybrid point cloud representations [207, 209] have also helped reduce memory cost, while capturing both fine-grained local details and contextual information.

However, by leveraging the local similarity of dense point clouds to construct geometry-informed data structures, superpoint-based methods can achieve an input reduction of several orders of magnitude, resulting in unparalleled efficiency. The next section provides a more in-depth introduction to these methods.

2.3 Superpoint-Based Learning

Part of our work draws inspiration from superpoint-based methods to develop efficient and scalable approaches for 3D semantic segmentation and panoptic segmentation. This section provides an overview of image partition methods and these more recently inspired 3D point cloud partition. We then focus on existing superpoint-based approaches for semantic segmentation, which serve as the basis for our methods in Chapter 3 and Chapter 4.

2.3.1 Superpixel Partitioning

Partitioning images into superpixels has been extensively studied to simplify image analysis, both before [265, 3] and after [308, 148] the widespread use of deep learning. By decomposing images into meaningful parts, superpixels offer a computationally efficient representation, which has been applied to various tasks such as object detection [284, 342], semantic segmentation [91, 283], and depth estimation [310].

Superpixel algorithms can be categorized into graph-based and clustering-based approaches. Graph-based methods formulate superpixel segmentation as a graph partitioning problem, with nodes corresponding to pixels and edges representing the connectivity between adjacent pixels. Popular algorithms

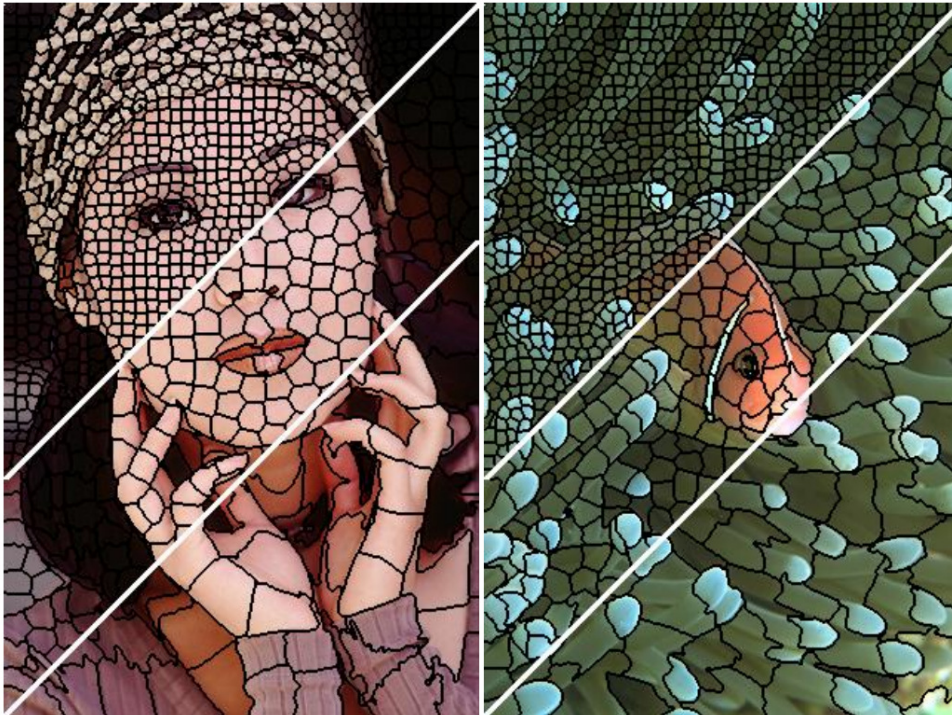


Figure 2.4 – **SLIC Superpixel Partition.** SLIC [3] is an image partitioning algorithm that segments images into compact, uniform superpixels. Here, two images are partitioned using SLIC with different resolutions. Source: [3]

in this category include normalized cuts [265], Felzenszwalb and Huttenlocher [85], energy minimization [26], and entropy rate superpixels [203]. Clustering-based methods embed pixels into a space where clustering techniques can be leveraged to produce a superpixel partition. Central to this category, SLIC [3] embeds pixels with their XY position and Lab color features, and uses a k-means variant to mitigate the computational cost. See Figure 2.4 for examples of superpixel partitions produced by SLIC. Subsequent works propose different features [195, 205], make SLIC faster [2], or differentiable [148].

2.3.2 Superpoint Partitioning

Partitioning large 3D point clouds into groups of adjacent and homogeneous points, called superpoints, is also an active area of research. Superpoint partitioning has been used successfully applied for point cloud oversegmentation.

tation [243, 200, 177], semantic segmentation [180, 140], and object detection [113, 80].

Akin to superpixels, superpoint methods may be clustering-based or graph-based. Clustering approaches such as VCCS [243] draw inspiration from SLIC [3] and use k-means on point features, under local adjacency constraints. However, k-means-based methods rely on a fixed number of randomly initialized clusters, proscribing the processing of point clouds of arbitrary size and geometric complexity. On the other hand, Landrieu *et al.* [180] cast point cloud oversegmentation as a structured optimization problem and uses the cut-pursuit [178] algorithm to generate superpoints. This method does not make any assumption on the number of superpoints and produces a partition whose granularity adapts to the 3D geometry.

2.3.3 Superpoints for Semantic Segmentation

Superpoint Graph (SPG) [180] proposes learning the relationship between superpoints using graph convolutions [287] for semantic segmentation, as detailed in Figure 2.5. While this method trains fast, its preprocessing is slow and its expressivity and range are limited, as it operates on a single partition. Recent works have proposed ways of learning superpoints themselves [177, 140, 302], which yields improved results but at the cost of an extra training step or a large point-based backbone [158].

Hierarchical partitions are used for image processing [12, 340, 357] and 3D analysis tasks, such as point cloud compression [82] and object detection [49, 196]. Hierarchical approaches for semantic segmentation use octrees with fixed grids [229, 266]. On the contrary, our model Superpoint Transformer (SPT), introduced in Chapter 3, uses a multiscale hierarchical structure that adapts to the local geometry of the data. This hierarchical partition conforms more closely to semantic boundaries than grid-based structures, enabling the

network to model the interactions between objects or object parts.

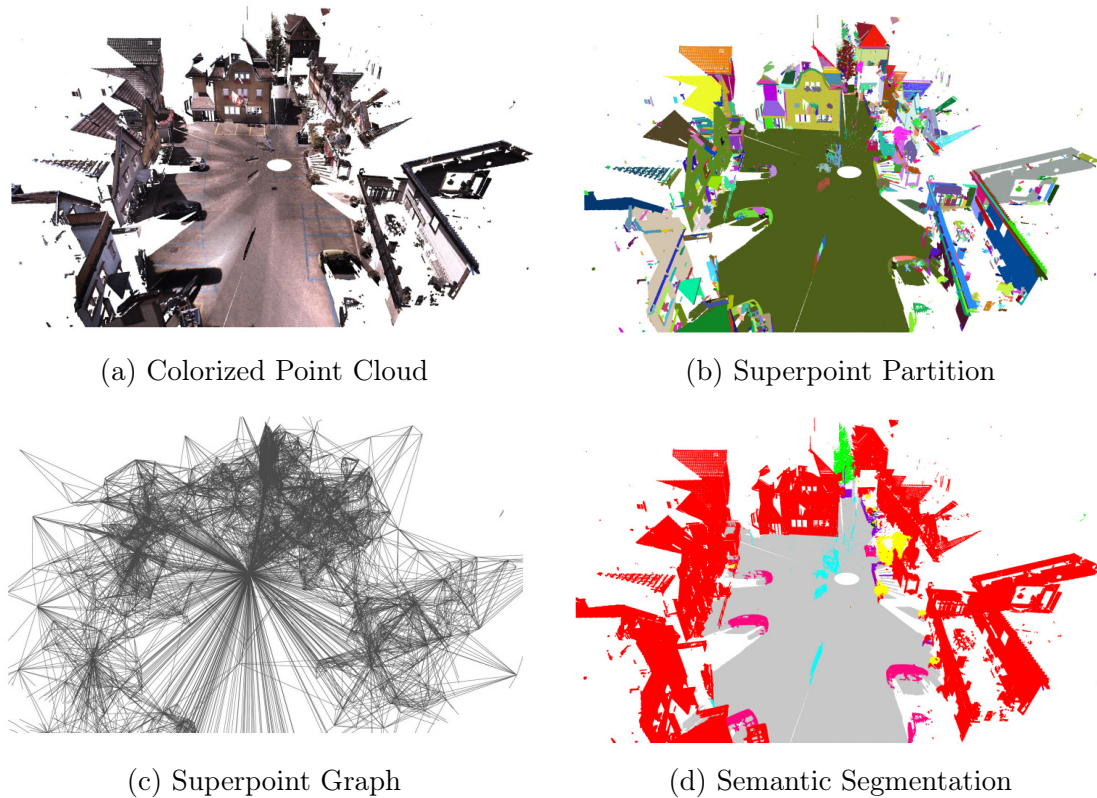


Figure 2.5 – **Superpoint Graph**. Superpoint Graph [180] partitions an input point cloud 2.5a into geometrically simple superpoints 2.5b. A superpoint graph 2.5c is then constructed by linking nearby superpoints. A network is trained to transform superpoints into compact embeddings then processed with graph convolutions, and finally classified into semantic labels 2.5d. Source: [180]

2.3.4 Superpoints for Panoptic Segmentation

The panoptic segmentation task is related to semantic and instance segmentation [162]. Like semantic segmentation, all points must be assigned a single semantic label. But unlike instance segmentation, each point must also belong to exactly one predicted instance. This setting makes partition-based methods naturally suitable for panoptic segmentation. We further elaborate on this idea and the related literature in Section 4.2.

The superpoint paradigm is central to our work. In Chapter 3, we get

inspiration from SPG [180] to develop an efficient 3D semantic segmentation architecture that relies on expressive transformer blocks to reason on a hierarchical partition of the scene. In Chapter 4, we propose a fast and scalable superpoint graph clustering approach to address panoptic segmentation of large 3D point clouds.

2.4 Leveraging Images for 3D Understanding

Point cloud understanding may be enriched by information from other modalities, such as images. In doing so, data-efficient methods can leverage readily-available knowledge from 2D models pretrained on large 2D datasets. In Chapter 5, we propose a method that learns to aggregate information from an arbitrarily-posed views of the same 3D object, based on their viewing conditions. This section contextualizes our approach. A deeper review of the literature on this topic can be found in Section 5.2.

2.4.1 Point Cloud Colorization

One way to exploit images to improve 3D understanding is to colorize point clouds. Unlike photogrammetry-based [303] acquisition techniques which naturally produce colorized points, active sensors such as LiDAR [150] or time-of-flight cameras [240] do not. In practice, these clouds can be colorized through a nontrivial heuristics-based preprocessing that requires localized RGB images and their camera parameters [154]. Colorized point cloud datasets [13, 111, 66, 197] are frequently used to compare 3D deep learning methods [252, 301, 58], which consistently perform better when radiometric information is available [255]. In short, point cloud colorization assumes either a specific sensor or heuristics-based preprocessing, and discards dense, contextual, multi-view information carried by images. Hence, 3D analysis methods capable of directly processing raw point clouds and localized images

would be less hardware-dependent and more data-efficient.

2.4.2 Learning to Fuse Points and Images

Advances in image and point cloud analysis using deep neural networks naturally invite to devise architectures capable of jointly extracting features from both modalities. We further present this active field of research in Section 5.2.

The work presented in Chapter 5 is a novel method for learning to extract and fuse information from arbitrarily-posed images and large point clouds in an end-to-end fashion.

Chapter 3

Efficient and Scalable 3D Semantic Segmentation

Abstract

We introduce a novel superpoint-based transformer architecture for efficient semantic segmentation of large-scale 3D scenes. Our method incorporates a fast algorithm to partition point clouds into a hierarchical superpoint structure, which makes our preprocessing 7 times faster than existing superpoint-based approaches. Additionally, we leverage a self-attention mechanism to capture the relationships between superpoints at multiple scales, leading to state-of-the-art performance on three challenging benchmark datasets: S3DIS (76.0% mIoU 6-fold validation), KITTI-360 (63.5% on Val), and DALES (79.6%). With only 212k parameters, our approach is up to 200 times more compact than other state-of-the-art models while maintaining similar performance. Furthermore, our model can be trained on a single GPU in 3 hours for a fold of the S3DIS dataset, which is $7\times$ to $70\times$ fewer GPU-hours than the best-performing methods. Our code and models are accessible at https://github.com/drprojects/superpoint_transformer.

This chapter’s work was initially presented in: Damien Robert, Hugo Raguet, Loic Landrieu, “Efficient 3D Semantic Segmentation With Superpoint Transformer”, ICCV, 2023.

3.1 Introduction

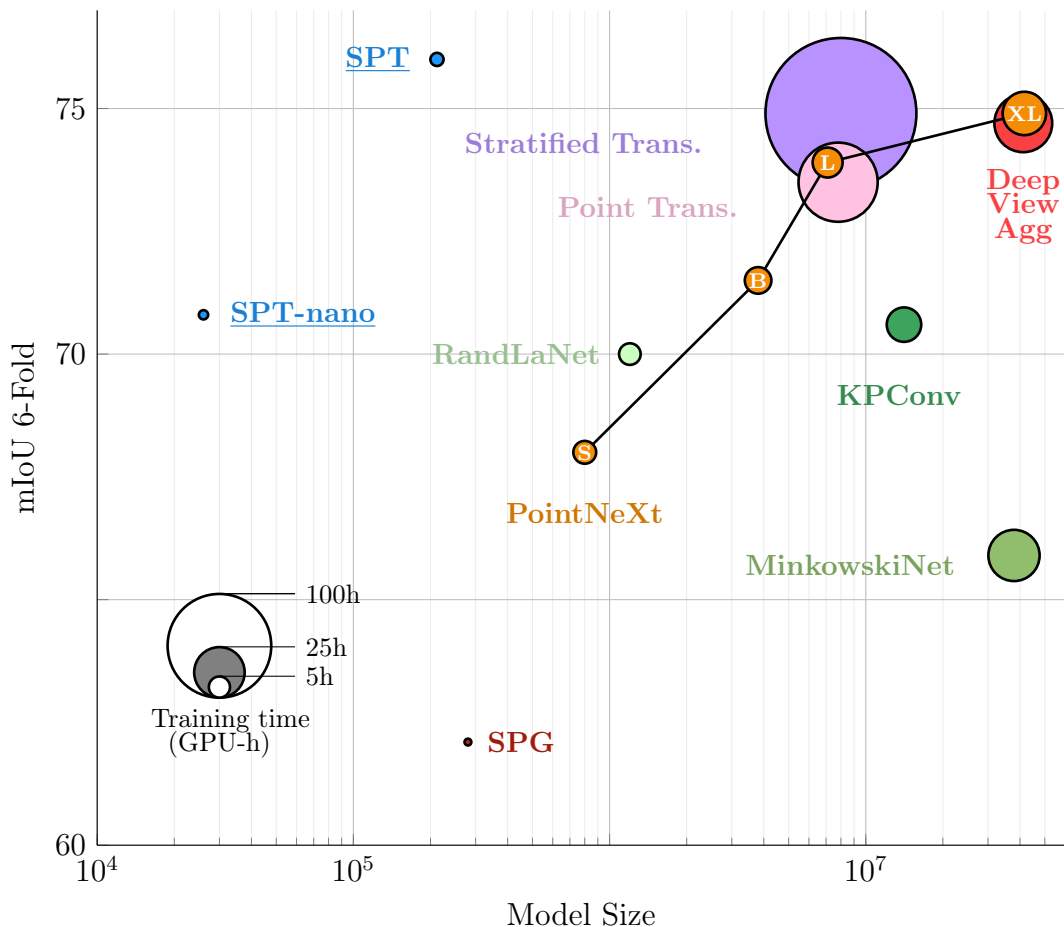


Figure 3.1 – **Model Size vs. Performance.** We visualize the performance of different methods on the S3DIS dataset (6-fold validation) in relation to their model size in log-scale. The area of the markers indicates the GPU-time to train on a single fold. Our proposed method Superpoint Transformer (SPT) achieves state-of-the-art with a reduction of up to 200-fold in model size and 70-fold in training time (in GPU-h) compared to recent methods. The even smaller SPT-nano model achieves a fair performance with 26k parameters only.

As the expressivity of deep learning models increases rapidly, so do their complexity and resource requirements [96]. In particular, vision transformers have demonstrated remarkable results for 3D point cloud semantic segmentation [359, 244, 109, 175, 209], but their high computational requirements make them challenging to train effectively. Additionally, these models rely on regular grids or point samplings, which do not adapt to the varying complexity of 3D data: the same computational effort is allocated everywhere, regardless

of the local geometry or radiometry of the point cloud. This issue leads to needlessly high memory consumption, limits the number of points that can be processed simultaneously, and hinders the modeling of long-range interactions.

Superpoint-based methods [180, 177, 140, 256] address the limitation of regular grids by partitioning large point clouds into sets of points—superpoints—which adapt to the local complexity. By directly learning the interaction between superpoints instead of individual points, these methods enable the analysis of large scenes with compact and parsimonious models that can be trained faster than standard approaches. However, superpoint-based methods often require a costly preprocessing, and their range and expressivity are limited by their use of local graph-convolution schemes [287].

In this chapter, we propose a novel superpoint-based transformer architecture that overcomes the limitations of both approaches, see Figure 3.1. Our method starts by partitioning a 3D point cloud into a hierarchical superpoint structure that adapts to the local properties of the acquisition at multiple scales simultaneously. To compute this partition efficiently, we propose a new algorithm that is an order of magnitude faster than existing superpoint preprocessing algorithms. Next, we introduce the Superpoint Transformer (SPT) architecture, which uses a sparse self-attention scheme to learn relationships between superpoints at multiple scales. By viewing the semantic segmentation of large point clouds as the classification of a small number of superpoints, our model can accurately classify millions of 3D points simultaneously without relying on sliding windows. SPT achieves near state-of-the-art accuracy on various open benchmarks while being significantly more compact and able to train much quicker than common approaches. The main contributions of this work are as follows:

— **Efficient Superpoint Computation:** We propose a new method to compute a hierarchical superpoint structure for large point clouds, which

is more than 7 times faster than existing superpoint-based methods. Our preprocessing time is also comparable or faster than standard approaches, addressing a significant drawback of superpoint methods.

— **State-of-the-Art Performance:** Our model reaches performance at or close to the state-of-the-art for three open benchmarks with distinct settings: S3DIS for indoor scanning [13], KITTI-360 for outdoor mobile acquisitions [197], and DALES for city-scale aerial LiDAR [311, 288].

— **Resource-Efficient Models:** SPT is particularly resource-efficient as it only has 212k parameters for S3DIS and DALES, a 200-fold reduction compared to other state-of-the-art models such as PointNeXt [255] and takes 70 times fewer GPU-h to train than Stratified Transformer [175]. The even more compact SPT-nano reaches 70.8% 6-Fold mIoU on S3DIS with only 26k parameters, making it the smallest model to reach above 70% by a factor of almost 300.

3.2 Method

Our method has two key components. First, we use an efficient algorithm to segment an input point cloud into a compact multiscale hierarchical structure. Second, a transformer-based network leverages this structure to classify the elements of the finest scale.

3.2.1 Efficient Hierarchical Superpoint Partition

We consider a point cloud \mathcal{C} with positional and radiometric information. To learn multiscale interactions, we compute a hierarchical partition of \mathcal{C} into geometrically-homogeneous superpoints of increasing coarseness; see Figure 3.2. We first define the concept of hierarchical partitions.

Definition 1 Hierarchical Partitions. A partition of a set \mathcal{X} is a collection of subsets of \mathcal{X} such that each element of \mathcal{X} is in one and only one of such

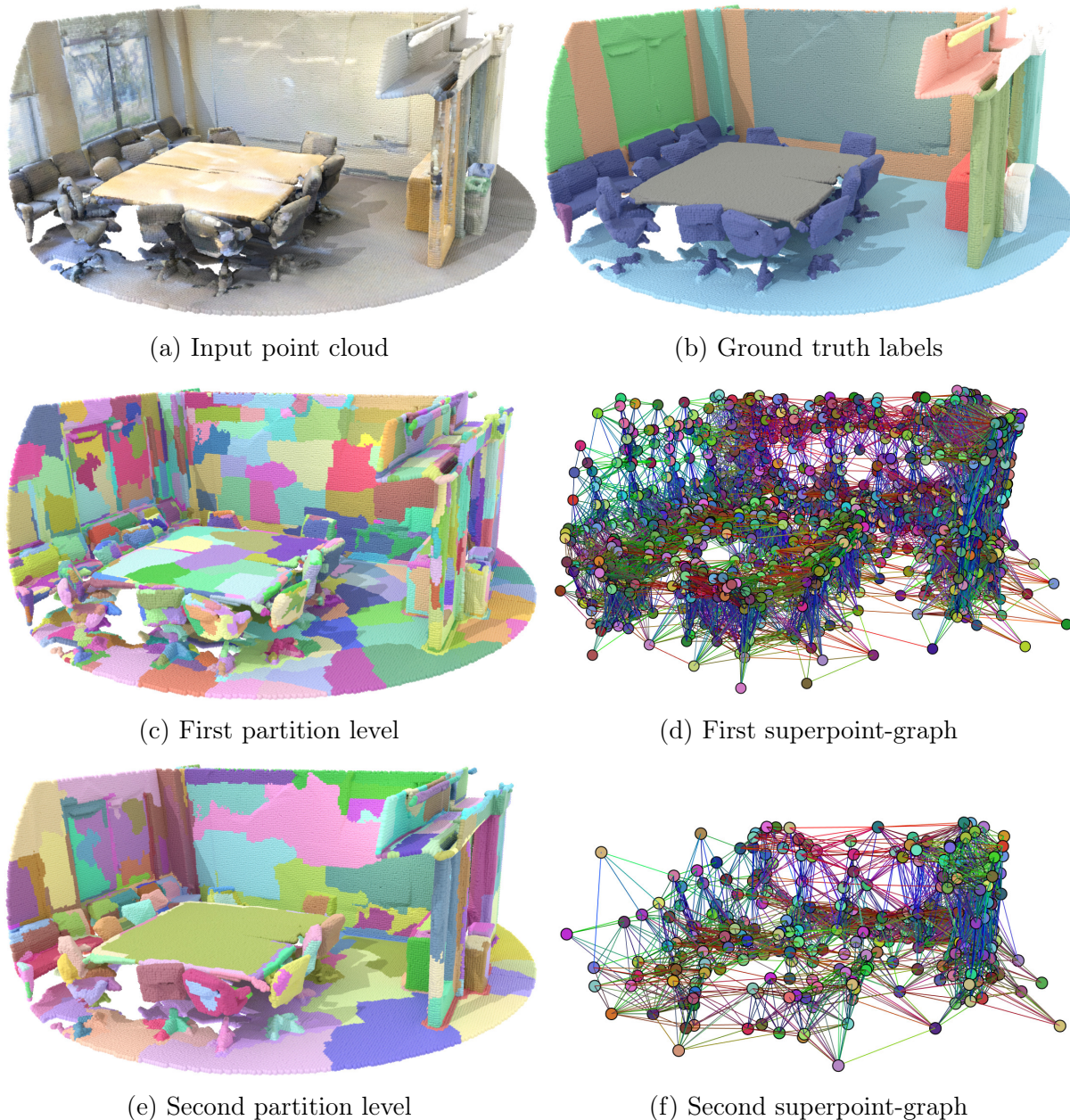


Figure 3.2 – **Superpoint Transformer**. Our method takes as input a point cloud (a) and computes its hierarchical partition into geometrically homogeneous superpoints at multiple scales: (c) and (e). For all partition levels, we construct superpoint adjacency graphs (d) and (f), which are used by an attention-based network to classify the finest superpoints.

subsets. $\mathcal{P} := [\mathcal{P}_0, \dots, \mathcal{P}_I]$ is a hierarchical partition of \mathcal{X} if $\mathcal{P}_0 = \mathcal{X}$, and \mathcal{P}_{i+1} is a partition of \mathcal{P}_i for $i \in [0, I - 1]$.

Throughout this chapter, all functions or tensors related to a specific partition level i are denoted with an exponent i .

Hierarchical Superpoint Partitions. We propose an efficient approach for constructing hierarchical partitions of large point clouds. First, we associate each point c of \mathcal{C} with features f_c representing its local geometric and radiometric information. These features can be handcrafted [108] or learned [177, 140]. See Section 3.2.4 for more details on point features. We also define a graph \mathcal{G} encoding the adjacency between points usually based on spatial proximity, *e.g.* k -nearest neighbors.

We view the features f_c for all c of \mathcal{C} as a signal f defined on the nodes of the graph \mathcal{G} . Following the ideas of SuperPoint Graph [180], we compute an approximation of f into constant components by solving an energy minimization problem penalized with a graph-based notion of *simplicity*. The resulting constant components form a partition whose granularity is determined by a regularization strength $\lambda > 0$: higher values yield fewer and coarser components.

For each component of the partition, we can compute the mean position (centroid) and feature of its elements, defining a coarser point cloud on which we can repeat the partitioning process. We can now compute a hierarchical partition $\mathcal{P} := [\mathcal{P}_0, \dots, \mathcal{P}_I]$ of \mathcal{C} from a list of regularization strengths $\lambda_1, \dots, \lambda_I$. First, we set \mathcal{P}_0 as the point cloud \mathcal{C} and f^0 as the point features f . Then, for $i = 1$ to I , we compute (i) a partition \mathcal{P}_i of f^{i-1} penalized with λ_i ; (ii) the mean signal f^i for all components of \mathcal{P}_i . The coarseness of the resulting partitions $[\mathcal{P}_0, \dots, \mathcal{P}_I]$ is thus strictly increasing. See Section B-4 for a more detailed description of this process, and Section B-5 for our parameterization recipe.

Hierarchical Graph Structure. A hierarchical partition defines a polytree structure across the different levels. Let p be an element of \mathcal{P}_i . If $i \in [0, I - 1]$, $\text{parent}(p)$ is the component of \mathcal{P}_{i+1} which contains p . If $i \in [1, I]$, $\text{children}(p)$

is the set of components of \mathcal{P}_{i-1} whose parent is p .

Superpoints also share adjacency relationships with superpoints *of the same partition level*. For each level $i \geq 1$, we build a *superpoint-graph* \mathcal{G}_i by connecting adjacent components of \mathcal{P}_i , *i.e.* superpoints whose closest points are within a distance gap $\epsilon_i > 0$. For $p \in \mathcal{P}_i$, we denote $\mathcal{N}(p) \subset \mathcal{P}_i$ the set of neighbors of p in the graph \mathcal{G}_i . See Section 3.2.5 for more details on the superpoint-graph construction.

Hierarchical Parallel ℓ_0 -Cut Pursuit. Computing the hierarchical components involves solving a recursive sequence of non-convex, non-differentiable optimization problems on large graphs. We propose an adaptation of the ℓ_0 -cut pursuit algorithm [179] to solve this problem. To improve efficiency, we adapt the graph-cut parallelization strategy initially introduced by Raguet *et al.* [261] in the convex setting.

3.2.2 Superpoint Transformer

Our proposed SPT architecture draws inspiration from the popular U-Net [270, 93]. However, instead of using grid, point, or graph subsampling, our approach derives its different resolution levels from the hierarchical partition \mathcal{P} .

General Architecture. As represented in Figure 3.3, SPT comprises an encoder with I stages and a decoder with $I - 1$ stages: the prediction takes place at the level \mathcal{P}_1 and not on individual points. We start by computing the relative positions x of all points and superpoints with respect to their parent. For a superpoint $p \in \mathcal{P}_i$, we define x_p^i as the position of the centroid of p relative to its parent's. The coarsest superpoints of \mathcal{P}_I have no parent and use the center of the scene as a reference centroid. We then normalize these values so that the sets $\{x_p^i | p \in \text{children}(q)\}$ have a radius of 1 for all

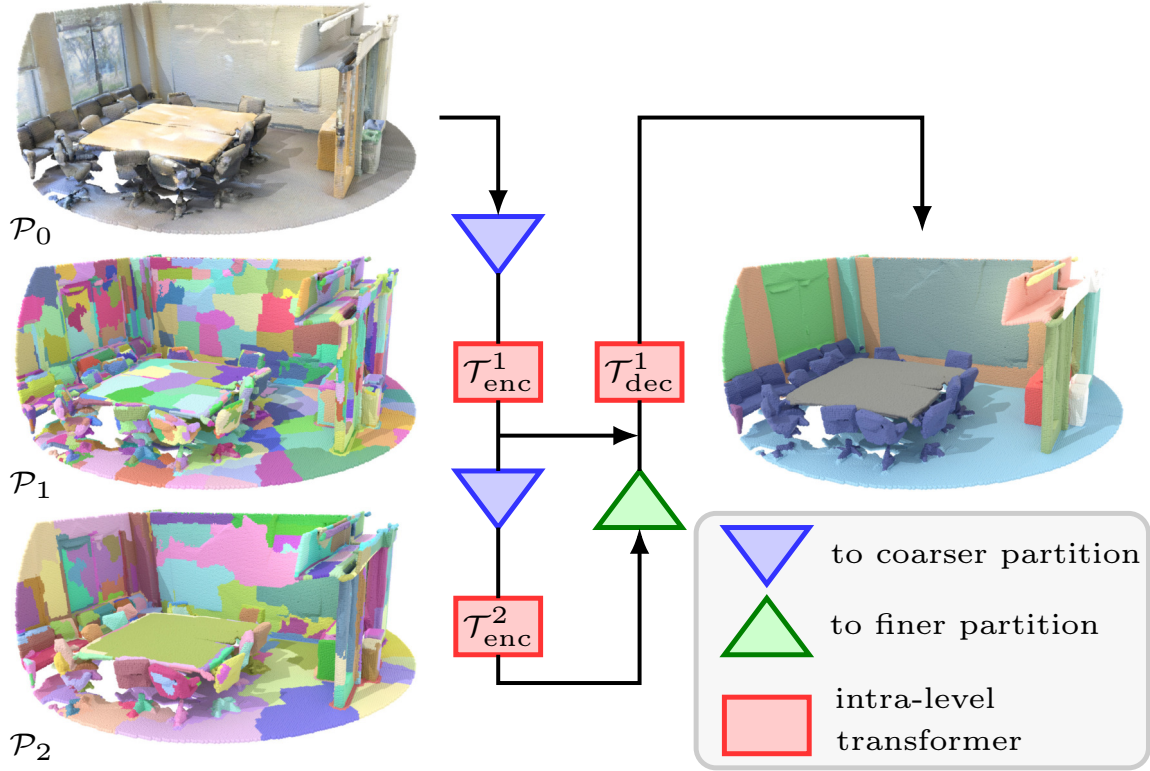


Figure 3.3 – **Superpoint Transformer**. We represent our proposed architecture with two partitions levels \mathcal{P}_1 and \mathcal{P}_2 . We use a transformer-based module to leverage the context at different scales, leading to large receptive fields. We only classify the superpoints of the partition \mathcal{P}_1 and not individual 3D points, allowing fast training and inference.

$q \in \mathcal{P}_{i+1}$. We compute features for each 3D point by using a multi-layer perceptron (MLP) to mix their relative positions and handcrafted features: $g^0 := \phi_{\text{enc}}^0([x^0, f^0])$, with $[\cdot, \cdot]$ the channelwise concatenation operator.

Each level $i \geq 1$ of the encoder maxpools the features of the finer partition level $i - 1$, adds relative positions x^i and propagates information between neighboring superpoints in \mathcal{G}_i . For a superpoint p in \mathcal{P}_i , this translates as:

$$g_p^i = \mathcal{T}_{\text{enc}}^i \circ \phi_{\text{enc}}^i \left(\left[x_p^i, \max_{q \in \text{children}(p)} (g_q^{i-1}) \right] \right) \quad (3.1)$$

with ϕ_{enc}^i an MLP and $\mathcal{T}_{\text{enc}}^i$ a transformer module explained below. By avoiding communication between the 3D points of \mathcal{P}_0 , we bypass a potential computational bottleneck.

The decoder passes information from the coarser partition level $i + 1$ to the finer level i . It uses the relative positions x^i and the encoder features g^i to improve the spatial resolution of its feature maps h^i [270]. For a superpoint p in partition \mathcal{P}_i with $1 \leq i < I - 1$, this can be expressed as:

$$h_p^i = \mathcal{T}_{\text{dec}}^i \circ \phi_{\text{dec}}^i \left(\left[x_p^i, g_p^i, h_{\text{parent}(p)}^{i+1} \right] \right) \quad (3.2)$$

with $h^I = g^I$, ϕ_{dec}^i an MLP, and $\mathcal{T}_{\text{dec}}^i$ an attention-based module similar to $\mathcal{T}_{\text{enc}}^i$.

Self-Attention Between Superpoints. We propose a variation of graph-attention networks [314] to propagate information between neighboring superpoints of the same partition level. For each level of the encoder and decoder, we associate to superpoint $p \in \mathcal{P}_i$ a triplet of key, query, value vectors K_p, Q_p, V_p of size $D_{\text{key}}, D_{\text{key}}$ and D_{val} . These values are obtained by applying a linear layer to the corresponding feature map m after GraphNorm normalization [39].

We then characterize the relationship between two superpoints p, q of \mathcal{P}_i adjacent in \mathcal{G}_i by a triplet of features $a_{p,q}^{\text{key}}, a_{p,q}^{\text{que}}, a_{p,q}^{\text{val}}$ of dimensions $D_{\text{key}}, D_{\text{key}}$ and D_{val} , and whose computation is detailed in the next section. Given a superpoint p , we stack the vectors $a_{p,q}^{\text{key}}, a_{p,q}^{\text{que}}, a_{p,q}^{\text{val}}$ for $q \in \mathcal{N}(p)$ in matrices $A_p^{\text{key}}, A_p^{\text{que}}, A_p^{\text{val}}$ of dimensions $|\mathcal{N}(p)| \times D_{\text{key}}$ or $|\mathcal{N}(p)| \times D_{\text{val}}$. The modules $\mathcal{T}_{\text{enc}}^i$ and $\mathcal{T}_{\text{dec}}^i$ gather contextual information as follows:

$$[\mathcal{T}(m)]_p^{\pm} = \text{att}(Q_p^{\top} \oplus A_p^{\text{que}}, K_{\mathcal{N}(p)} + A_p^{\text{key}}, V_{\mathcal{N}(p)} + A_p^{\text{val}}), \quad (3.3)$$

with \pm a residual connection [118], \oplus the addition operator with broadcasting on the first dimension, and $K_{\mathcal{N}(p)}$ the matrix of stacked vectors K_q for $q \in \mathcal{N}(p)$. The attention mechanism writes as follows:

$$\text{att}(Q, K, V) := V^{\top} \text{softmax} \left(\frac{Q \odot K \mathbf{1}}{\sqrt{|\mathcal{N}(p)|}} \right), \quad (3.4)$$

with \odot the Hadamard termwise product and $\mathbf{1}$ a column-vector with D_{key} ones. Our proposed scheme is similar to classic attention schemes with two differences: (i) the queries adapt to each neighbor, and (ii) we normalize the softmax with the neighborhood size instead of the key dimension. In practice, we use multiple independent attention modules in parallel (multi-head attention) and several consecutive attention blocks.

3.2.3 Leveraging the Hierarchical Graph Structure

The hierarchical superpoint partition \mathcal{P} can be used for more than guidance for graph pooling operations. Indeed, we can learn expressive adjacency encodings capturing the complex adjacency relationships between superpoints and employ powerful supervision and augmentation strategies based on the hierarchical partitions.

Adjacency Encoding. While the adjacency between two 3D points is entirely defined by their distance vector, the relationships between superpoints are governed by additional factors such as their alignment, proximity, and difference in sizes or shapes. We characterize the adjacency of pairs of adjacent superpoints of the same partition level using a set of handcrafted features whose description is provided in Section 3.2.4.

For each pair of superpoints (p, q) adjacent in \mathcal{G}_i , we jointly compute the concatenated $a_{p,q}^{\text{key}}, a_{p,q}^{\text{que}}, a_{p,q}^{\text{val}}$ by applying an MLP ϕ_{adj}^i to the handcrafted adjacency features defined above. Further details on the superpoint-graph construction are provided in Section 3.2.5.

Hierarchical Supervision. We propose to take advantage of the nested structure of the hierarchical partition \mathcal{P} into the supervision of our model. We can naturally associate the superpoints of any level $i \geq 1$ with a set of 3D points in \mathcal{P}_0 . The superpoints at the finest level $i = 1$ are almost semantically

pure (see Figure 3.7), while the superpoints at coarser levels $i > 1$ typically encompass multiple objects. Therefore, we use a dual learning objective: (i) we predict the most frequent label within the superpoints of \mathcal{P}_1 , and (ii) we predict the label distribution for the superpoints of \mathcal{P}_i with $i > 1$. We supervise both predictions with the cross-entropy loss.

Let y_p^i denote the true label distribution of the 3D points within a superpoint $p \in \mathcal{P}_i$, and \hat{y}_p^i a one-hot-encoding of its most frequent label. We use a dedicated linear layer at each partition level to map the decoder feature g_p^i to a predicted label distribution z_p^i . Our objective function can be formulated as follows:

$$\mathcal{L} = \sum_{p \in \mathcal{P}_1} \frac{-N_p^1}{|\mathcal{C}|} H(\hat{y}_p^1, z_p^1) + \sum_{i=2}^I \sum_{p \in \mathcal{P}_i} \frac{\mu^i N_p^i}{|\mathcal{C}|} H(y_p^i, z_p^i), \quad (3.5)$$

where μ^2, \dots, μ^I are positive weights, N_p^i represents the number of points within a superpoint $p \in \mathcal{P}_i$, and $|\mathcal{C}|$ is the total number of points in the point cloud, and $H(y, z) = -\sum_{k \in \mathcal{K}} y_k \log(z_k)$ and \mathcal{K} the class set.

Superpoint-Based Augmentations. Although our approach classifies superpoints rather than individual 3D points, we still need to load the points of \mathcal{P}_0 in memory to embed the superpoints from \mathcal{P}_1 . However, since superpoints are designed to be geometrically simple, only a subset of their points is needed to characterize their shape. Therefore, when computing the feature g_p^1 of a superpoint p of \mathcal{P}_1 containing n points with Equation 3.1, we sample only a portion $\tanh(n/n_{\max})$ of its points, with a minimum of n_{\min} . This sampling strategy reduces the memory load and acts as a powerful data augmentation. The lightweight version of our model SPT-nano goes even further. It ignores the points entirely and only use handcrafted features to embed the superpoints of \mathcal{P}_1 , thus avoiding entirely the complexity associated with the size of the input point cloud \mathcal{P}_0 .

To further augment the data, we exploit the geometric consistency of superpoints and their hierarchical arrangement. During the batch construction, we randomly drop each superpoint with a given probability at all levels. Dropping superpoints at the fine levels removes random objects or object parts, while dropping superpoints at the coarser levels removes entire structures such as walls, buildings, or portions of roads, for example.

3.2.4 Handcrafted Features

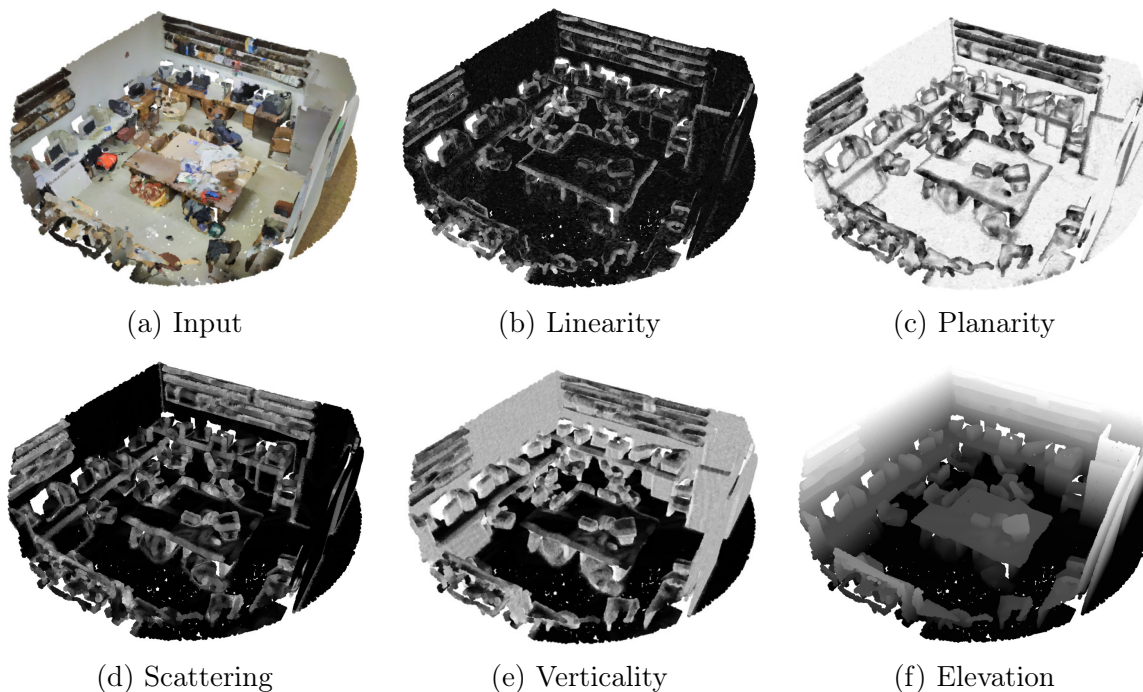


Figure 3.4 – **Point Geometric Features.** Given an input cloud (a), the computed PCA-based geometric features (b, c, d, e) and distance to the ground (f) offer a simple characterization of the local geometry around each point.

Similar to SPG [180], our method relies on simple handcrafted features to build the hierarchical partition and learn meaningful points and adjacency relationships. In this section, we provide further details on the definition of these features and how to compute them. It is important to note that these features are only computed once during preprocessing, and thanks to our optimized implementation, this step only takes a few minutes.

Point Features. We can associate each 3D point with a set of 8 easy-to-compute handcrafted features, described below.

- *Radiometric features* (3 or 1): RGB colors are available for S3DIS and KITTI-360, and intensity values for DALES. These radiometric features are normalized to $[0, 1]$ at preprocessing time. For KITTI-360, we find that using the HSV color model yields better results.
- *Geometric features* (5): We use PCA-based features: *linearity*, *planarity*, *scattering*, [70] and *verticality* [108], computed on the set of 50-nearest neighbors of each point. This neighbor search is only computed once during preprocessing and is also necessary to build the graph \mathcal{G} . We also define *elevation* as the distance between a point and the ground below it. Since the ground is neither necessarily flat nor horizontal, we use the RANSAC algorithm [88] on a coarse subsampling of the scene to find a ground plane. We normalize the elevation by dividing it by 4 for S3DIS and 20 for DALES and KITTI-360.

At preprocessing time, we only use radiometric and geometric features to compute the hierarchical partition. At training time, as mentioned in Section 3.2.2, SPT computes point embeddings by mapping all available point features, along with the normalized point position, to a vector of size D_{point} with a dedicated MLP ϕ_{enc}^0 .

We provide an illustration of the geometric point features in Figure 3.4, to help the reader apprehend these simple geometric descriptors.

Adjacency Features. The relationship between adjacent superpoints provides crucial information to leverage their context. For each edge of the superpoint-graph, we compute the 18 following features:

- *Interface features* (7): All adjacent superpoints share an *interface*, *i.e.*

pairs of points from each superpoint that are close and share a line of sight. SPG [180] uses the Delaunay triangulation of the entire point cloud to compute such interfaces, while we propose a faster heuristic approach in Section 3.2.5 called the *Approximate Superpoint Gap algorithm*. Each pair of points of an interface defines an offset, *i.e.* a vector pointing from one superpoint to its neighbor. We compute the mean offset (dim 3), the mean offset length (dim 1), and the standard deviation of the offset in each canonical direction (dim 3).

- *Ratio features (4)*: As defined in [180], we characterize each pair of adjacent superpoints with the ratio of their *lengths*, *surfaces*, *volumes*, and *point counts*.
- *Pose features (7)*: For each superpoint, we define a normal vector as its principal component with the smallest eigenvalue. We then characterize the relative position between two superpoints with the cosine of the angle between the superpoint normal vectors (dim: 1) and between each of the two superpoints’ normal and the mean offset direction (dim: 2). Additionally, the offset between the centroids of the superpoints is used to compute the centroid distance (dim: 1) and the unit-normalized centroid offset direction (dim: 3).

Note that the mean offset and the ratio features are not symmetric and imply that the edges of the superpoint-graphs are oriented.

As mentioned in Section Section 3.2.3, a network ϕ_{adj}^i maps these hand-crafted features to a vector of size $D_{\text{key}} + D_{\text{que}} + D_{\text{val}}$, for each level $i \geq 1$ of the encoder and the decoder.

3.2.5 Superpoint-Graphs Computation

The Superpoint Graph method by Landrieu and Simonovsky [180] builds a superpoint-graph from a point cloud using Delaunay triangulation, which can

take a long time for large point clouds. In contrast, our approach connects two superpoints in \mathcal{P}_i , where $i \geq 1$ if their closest points are within a distance gap $\epsilon_i > 0$. However, computing pairwise distances for all points is computationally expensive. We propose a heuristic to approximately find the closest pair of points for two superpoints, see Algorithm 1. We also accelerate the computation of adjacent superpoints by approximating only for superpoints with centroids closer than the sum of their radii plus the gap distance. This approximation helps to reduce the number of computations required for adjacency computation, which leads to faster processing times. All steps involved in the computation of our superpoint-graph are implemented on the GPU to further enhance computational efficiency.

Algorithm 1 Approximate Superpoint Gap

Input: superpoints p_1 and p_2 , num_steps
 $c_1 \leftarrow \text{centroid}(p_1)$
 $c_2 \leftarrow \text{centroid}(p_2)$
for $s \in \text{num_steps}$ **do**
 $c_2 \leftarrow \arg \min_{p \in p_2} \|c_1 - p\|$
 $c_1 \leftarrow \arg \min_{p \in p_1} \|c_2 - p\|$
end for
 return $\|c_1 - c_2\|$

Recovering the interface between two adjacent superpoints as evoked in Section 3.2.4 involves a notion of visibility: we connect points from each superpoint which are *facing* each other. This can be a challenging and ambiguous problem, which SuperPoint Graph [178] tackles using a Delaunay triangulation of the points. However, this method is impractical for large point clouds. To address this issue, we propose a heuristic approach with the following steps: (i) first, we use the Approximate Superpoint Gap algorithm to compute the approximate nearest points for each superpoint. Then, we restrict the search to only consider points within a certain distance of the nearest points. Finally, we match the points by sorting them along the principal component of the selected points.

3.3 Experiments

We evaluate our model on three diverse datasets described in Section 3.3.1. In Section 3.3.2, we evaluate our approach in terms of precision, but also quantify the gains in terms of preprocessing, training, and inference times. Finally, we propose an extensive ablation study in Section 3.3.3.

3.3.1 Datasets and Models

Datasets. To demonstrate its versatility, we evaluate SPT on three large-scale datasets of different natures.

S3DIS [13]. This indoor dataset of office buildings contains over 274 million points across 6 building floors—or areas. The dataset is organized by individual rooms, but can also be processed by considering entire areas at once.

KITTI-360 [197]. This outdoor dataset contains more than 100 k laser scans acquired in various urban settings on a mobile platform. We use the *accumulated point clouds* format, which consists of large scenes with around 3 million points. There are 239 training scenes and 61 for validation.

DALES [311, 288]. This 10 km² aerial LiDAR dataset contains 500 millions of points across 40 urban and rural scenes, including 12 for evaluation.

We subsample the datasets using a 3 cm grid for S3DIS, and 10 cm for KITTI-360 and DALES. All accuracy metrics are reported for the full, unsampled point clouds. We use a two-level partition ($I = 2$) with $\mu^2 = 50$ for all datasets and select the partition parameters to obtain a 30-fold reduction between \mathcal{P}_1 and \mathcal{P}_0 and a further 5-fold reduction for \mathcal{P}_2 . See Table 3.1 for more details.

Models. We use the same model configuration for all three datasets with minimal adaptations. All transformer modules have a shared width D_{val} ,

Table 3.1 – **Partition Configuration.** We report the point count of different datasets before and after subsampling, as well as the size of the partitions.

Dataset	Points	Subsampled	$ \mathcal{P}_1 $	$ \mathcal{P}_2 $
S3DIS [13]	273m	32m	979k	292k
DALES [311, 288]	492m	449m	14.8m	2.56m
KITTI-360 [197]	919m	432m	16.2m	2.98m

a small key space of dimension $D_{\text{key}} = 4$, 16 heads, with 3 blocks in the encoder and 1 in the decoder. We set $D_{\text{val}} = 64$ for S3DIS and DALES (210k parameters), and $D_{\text{val}} = 128$ (777k parameters) for KITTI360. See the Section B-6 and our open repository for the detailed configuration of all modules.

We also propose SPT-nano, a lightweight version of our model that does not compute point-level features but operates directly on the first partition level \mathcal{P}_1 . The value of the maxpool over points in Equation 3.1 for $i = 1$ is replaced by f^1 , the aggregated handcrafted point features at the level 1 of the partition. This model never considers the full point cloud \mathcal{P}_0 but only operates on the partitions. For this model, we set $D_{\text{val}} = 16$ for S3DIS and DALES (26k parameters), and $D_{\text{val}} = 32$ for KITTI360 (70k parameters).

Batch Construction. Batches are sampled from large *tiles*: entire building floors for S3DIS, and full scenes for KITTI-360 or DALES. Each batch is composed of 4 randomly sampled portions of the partition with a radius of 7 m for S3DIS and 50 m for KITTI-360 and DALES, allowing us to model long-range interactions. During training, we apply a superpoint dropout rate of 0.2 for each superpoint at all hierarchy levels, as well as random rotation, tilting, point jitter and handcrafted features dropout. When sampling points within each superpoint, we set $n_{\text{min}} = 32$ and $n_{\text{max}} = 128$.

Optimization. We use the ADAMW optimizer [212] with default parameters, a weight decay of 10^{-4} , a learning rate of 10^{-2} for DALES and KITTI-360 on and 10^{-1} for S3DIS. The learning rate for the attention modules is 10 times smaller than for other weights. Learning rates are warmed up from 10^{-6} for 20 epochs and progressively reduced to 10^{-6} with cosine annealing [213].

3.3.2 Quantitative Evaluation

Table 3.2 – **Performance Evaluation.** We report the Mean Intersection-over-Union of different methods on three different datasets. SPT performs on par or better than recent methods with significantly fewer parameters. † superpoint-based. ★/* model with 777k/70k parameters.

Model	Size $\times 10^6$	S3DIS		KITTI	DALES
		6-Fold	Area 5	360 val	
PointNet++ [253]	3.0	56.7	-	-	68.3
† SPG [180]	0.28	62.1	58.0	-	60.6
ConvPoint [24]	4.7	68.2	-	-	67.4
† SPG + SSP [177]	0.29	68.4	61.7	-	-
† SPNet [140]	0.32	68.7	-	-	-
MinkowskiNet [58, 46]	37.9	69.1	65.4	58.3	-
RandLANet [133]	1.2	70.0	-	-	-
KPConv [301]	14.1	70.6	67.1	-	81.1
Point Trans.[359]	7.8	73.5	70.4	-	-
RepSurf-U [263]	0.97	74.3	68.9	-	-
DeepViewAgg [269]	41.2	74.7	67.2	62.1	-
Strat. Trans. [175, 323]	8.0	74.9	72.0	-	-
PointNeXt-XL [255]	41.6	74.9	71.1	-	-
† SPT (ours)	0.21	76.0	68.9	63.5*	79.6
† SPT-nano (ours)	0.026	70.8	64.9	57.2*	75.2

Performance Evaluation. As seen in Table 3.2, SPT performs at the state-of-the-art on two of three datasets despite being a significantly smaller model. On S3DIS, SPT beats PointNeXt-XL with $196\times$ fewer parameters. On

KITTI-360, SPT outperforms MinkowskiNet despite a size ratio of 49, and surpasses the performance of the even larger multimodal point-image model DeepViewAgg. On DALES, SPT outperforms ConvPoint by more than 12 points with over 21 times fewer parameters. Although SPT is 1.5 points behind KPConv on this dataset, it achieves these results with 67 times fewer parameters. SPT achieves significant performance improvements over all superpoint-based methods on all datasets, ranging from 7 to 14 points. SPT overtakes the SSP and SPNet superpoint methods that *learn* the partition in a two-stage training setup, leading to preprocessing times of several hours.

Interestingly, the lightweight SPT-nano model matches KPConv and MinkowskiNet with only 26k parameters.

See Figure 3.5 for qualitative illustrations.

Preprocessing Speed. As reported in Table 3.3, our implementation of the preprocessing is highly efficient. We can compute partitions, superpoint-graphs, and handcrafted features, and perform I/O operations quickly: 12.4 min for S3DIS, 117 for KITTI-360, and 148 for DALES using a server with a 48-core CPU. An 8-core workstation can preprocess S3DIS in 26.6 min. Our preprocessing time is as fast or faster than point-level methods and $7\times$ faster than SuperPoint Graph’s, thus alleviating one of the main drawbacks of superpoint-based methods.

Training Speed. We trained several state-of-the-art methods from scratch and report in Figure 3.6 the evolution of test performance as a function of training time. We used the official training logs for the multi-GPU Point Transformer and Stratified Transformer. SPT can train much faster than all methods not based on superpoints while attaining similar performance. Although Superpoint Graph trains even faster, its performance saturates earlier, 6.0 mIoU points below SPT. We also report the inference time of our method

Table 3.3 – **Efficiency Analysis.** We report the preprocessing time for the entire S3DIS dataset and the training and inference time for Area 5. SPT and SPT-nano shows significant speedups in preprocessing, training, and inference times.

	Preprocessing in min	Training in GPU-h	Inference in s
PointNet++ [253]	8.0	6.3	42
KPConv [301]	23.1	14.1	162
MinkowskiNet [58]	20.7	28.8	83
Stratified Trans. [175]	8.0	216.4	30
Superpoint Graph [180]	89.9	1.3	16
SPT (ours)	12.4	3.0	2
SPT-nano (ours)	12.4	1.9	1

in Table 3.3, which is significantly lower than competing approaches, with a speed-up factor ranging from 8 to 80. All speed measurements were conducted on a single-GPU server (48 cores, 512Go RAM, A40 GPU). Nevertheless, our model can be trained on a standard workstation (8 cores, 64Go, 2080Ti) with smaller batches, taking only 1.5 times longer and with comparable results.

SPT performs on par or better than complex models with up to two orders of magnitude more parameters and significantly longer training times. Such efficiency and compactness are beneficial for real-world scenarios where hardware, time, or energy may be limited.

3.3.3 Ablation Study

We evaluate the impact of several design choices in Table 3.4 and reports here our observations.

a) Handcrafted features. Without handcrafted point features, our model performs worse on all datasets. This observation is in line with other works which also remarked the positive impact of well-designed handcrafted features on the performance of smaller models [131, 263].

A more in-depth investigation of the influence of the handcrafted features can be found in Section B-3.

b) Influence of Edges. Removing the adjacency encoding between superpoints leads to a significant drop of 6.3 points on S3DIS; characterizing the relative position and relationship between superpoints appears crucial to exploiting their context. We also find that pruning the 50% longest edges of each superpoint results in a systematic performance drop, highlighting the importance of modeling long relationships.

Table 3.4 – **Ablation Study.** Impact of some of our design choices on the mIoU for all tested datasets.

Experiment	S3DIS 6-Fold	KITTI 360 Val	DALES
Best Model	76.0	63.5	79.6
a) No handcrafted features	-0.7	-4.1	-1.4
b) No adjacency encoding	-6.3	-5.4	-3.0
b) Fewer edges	-3.5	-1.1	-0.3
c) No point sampling	-1.3	-0.9	-0.5
c) No superpoint sampling	-2.7	-2.5	-0.7
c) Only 1 partition level	-8.4	-5.1	-0.9

c) Partition-Based Improvements. We assess the impact of several improvements made possible by using hierarchical superpoints. First, we find that using all available points when embedding the superpoints of \mathcal{P}_1 instead of random sampling resulted in a small performance drop. Second, setting the superpoint dropout rate to 0 worsens the performance by over 2.5 points on S3DIS and KITTI-360.

While we did not observe better results with three or more partition levels, only using one level leads to a significant loss of performance for all datasets.

d) Influence of Partition Purity. In Figure 3.7, we plot the performance of the “oracle” model which associates each superpoint of \mathcal{P}_1 with its most frequent true label. This model acts as an upper bound on the achievable performance with a given partition. Our proposed partition has significantly higher semantic purity than a regular voxel grid with as many nonempty voxels as superpoints. This implies that our superpoints adhere better to semantic boundaries between objects.

We also report the performance of our model for different partitions of varying coarseness, measured as the number of superpoints in \mathcal{P}_1 . Using, respectively, $\times 1.5$ ($\times 3$) fewer superpoints leads to a performance drop of 2.2 (4.7) mIoU points, but reduce the training time to 2.4 (1.6) hours. The performance of SPT is more than 20 points below the oracle, suggesting that the partition does not strongly limit its performance.

3.3.4 Model Scalability

We study the scalability of SPT by comparing models with different parameter counts on each dataset. It is important to note that the superpoint approach drastically compresses the training set, which can lead to overfitting, see Section 3.3.6. For example, as illustrated in Table 3.5, SPT-128 with $D_{\text{val}} = 128$ (777k param.) performs 1.4 points below $D_{\text{val}} = 64$ on S3DIS.

We report a similar behavior for other hyperparameters: in Table 3.6, $D_{\text{key}} = 8$ instead of 4 incurs a drop of 1.0, while in Table 3.7, $N_{\text{heads}} = 32$ instead of 16 a drop of 0.1 point. For the larger KITTI-360 dataset (13M nodes), $D_{\text{val}} = 128$ performs 1.9 points above $D_{\text{val}} = 64$, but 5.4 points above $D_{\text{val}} = 256$ (2.7m param.).

Table 3.5 – **Impact of Model Scaling.** Impact of model size for each dataset.

Model	Size $\times 10^6$	S3DIS 6-Fold	KITTI 360 Val	DALES
SPT-32	0.14	74.5	60.6	78.7
SPT-64	0.21	76.0	61.6	79.6
SPT-128	0.77	74.6	63.5	78.8
SPT-256	1.80	74.0	58.1	77.6

Table 3.6 – **Impact of Query-Key Dimension.** Impact of D_{key} on S3DIS 6-Fold.

D_{key}	2	4	8	16
SPT-64	75.6	76.0	75.0	74.7

Table 3.7 – **Impact of Heads Count.** Impact of the number of heads N_{head} on the S3DIS 6-Fold performance.

N_{head}	4	8	16	32
SPT-64	74.3	75.2	76.0	75.9

3.3.5 Hierarchical Supervision

We explore, in Table 3.8, alternatives to our hierarchical supervision introduced in Section Section 3.2.3 : predicting the most frequent label for \mathcal{P}_1 and the distribution for \mathcal{P}_2 . We use “freq- \mathcal{P}_i ” to refer to the prediction of the most frequent label applied the \mathcal{P}_i partition. Similarly, “dist- \mathcal{P}_i ” denotes the prediction of the distribution of labels within each superpoint of the partition \mathcal{P}_i .

We observe a consistent improvement across all datasets by adding the dist- \mathcal{P}_i supervision. This illustrates the benefits of supervising higher-level partitions, despite their lower purity. Moreover, supervising \mathcal{P}_1 with the distribution rather than the most frequent label leads to a further performance

drop. This validates our choice to consider \mathcal{P}_1 superpoints as sufficiently pure to be supervised using their dominant label.

Table 3.8 – **Ablation on Supervision.** Impact of our hierarchical supervision for each dataset.

Loss	S3DIS	KITTI	DALES
	6-Fold	360 Val	
freq- \mathcal{P}_i - \mathcal{P}_1 dist- \mathcal{P}_i - \mathcal{P}_2	76.0	63.5	79.6
freq- \mathcal{P}_1	-0.2	-0.8	-0.8
dist- \mathcal{P}_i - \mathcal{P}_1	-0.8	-1.3	-0.8

3.3.6 Limitations

Our model provides significant advantages in terms of speed and compactness but also comes with its own set of limitations.

Overfitting and Scaling. The superpoint approach drastically simplifies and compresses the training sets: the 274M 3D points of S3DIS are captured by a geometry-driven multilevel graph structure with fewer than 1.25M nodes. While this simplification favors the compactness and speed of the training of the model, this can lead to overfitting when using SPT configurations with more parameters, as shown in Section 3.3.4. Scaling our model to millions of parameters may only yield better results for training sets that are sufficiently large, diverse, and complex.

Errors in the Partition. Object boundaries lacking obvious discontinuities, such as curbs vs. roads or whiteboards vs. walls, are not well recovered by our partition. As partition errors cannot be corrected with our approach, this may lead to classification errors. To improve this, we could replace our handcrafted point descriptors (Section 3.2.4) with features directly learned for partitioning [177, 140]. However, such methods significantly increase the

preprocessing time, contradicting our current focus on efficiency. In line with [131, 263], we use easy-to-compute yet expressive handcrafted features. Our model SPT-nano without point encoder relies purely on such features and reaches 70.8 mIoU on S3DIS 6-Fold with only 27k param, illustrating this expressivity.

Learning Through the Partition. The idea of learning point and adjacency features directly end-to-end is a promising research direction to improve our model. However, this implies efficiently backpropagating through superpoint hard assignments, which remains an open problem. Furthermore, such a method would consider individual 3D points during training, which would necessitate to perform the partitioning step multiple times during training, which may negate the efficiency of our method

Predictions. Finally, our method predicts labels at the superpoint level P_1 and not individual 3D points. Since this may limit the maximum performance achievable by our approach, we could consider adding an upsampling layer to make point-level predictions. However, this does not appear to us as the most profitable research direction. Indeed, this may negate some of the efficiency of our method. Furthermore, as shown in the ablation study Section 3.3.3 of the present chapter, the “oracle” model outperforms ours by a large margin. This may indicate that performance improvements should primarily be searched in superpoint classification rather than in improving the partition.

Our model also learns features for superpoints and not individual 3D points. This may limit downstream tasks requiring 3D point features, such as surface reconstruction or panoptic segmentation. However, we argue that specific adaptations could be explored to perform these tasks at the superpoint level.

3.4 Conclusion

We have introduced the Superpoint Transformer approach for semantic segmentation of large point clouds, combining superpoints and transformers to achieve state-of-the-art results with significantly reduced training time, inference time, and model size. This approach particularly benefits large-scale applications and computing with limited resources. More broadly, we argue that small, tailored models can offer a more flexible and sustainable alternative to large, generic models for 3D learning. With training times of a few hours on a single GPU, our approach allows practitioners to easily customize the models to their specific needs, enhancing the overall usability and accessibility of 3D learning.

Acknowledgements. This work was funded by ENGIE Lab CRIGEN. This work was supported by ANR project READY3D ANR-19-CE23-0007, and was granted access to the HPC resources of IDRIS under the allocation AD011013388R1 made by GENCI. We thank Bruno Vallet, Romain Loiseau and Ewelina Rupnik for inspiring discussions and valuable feedback.

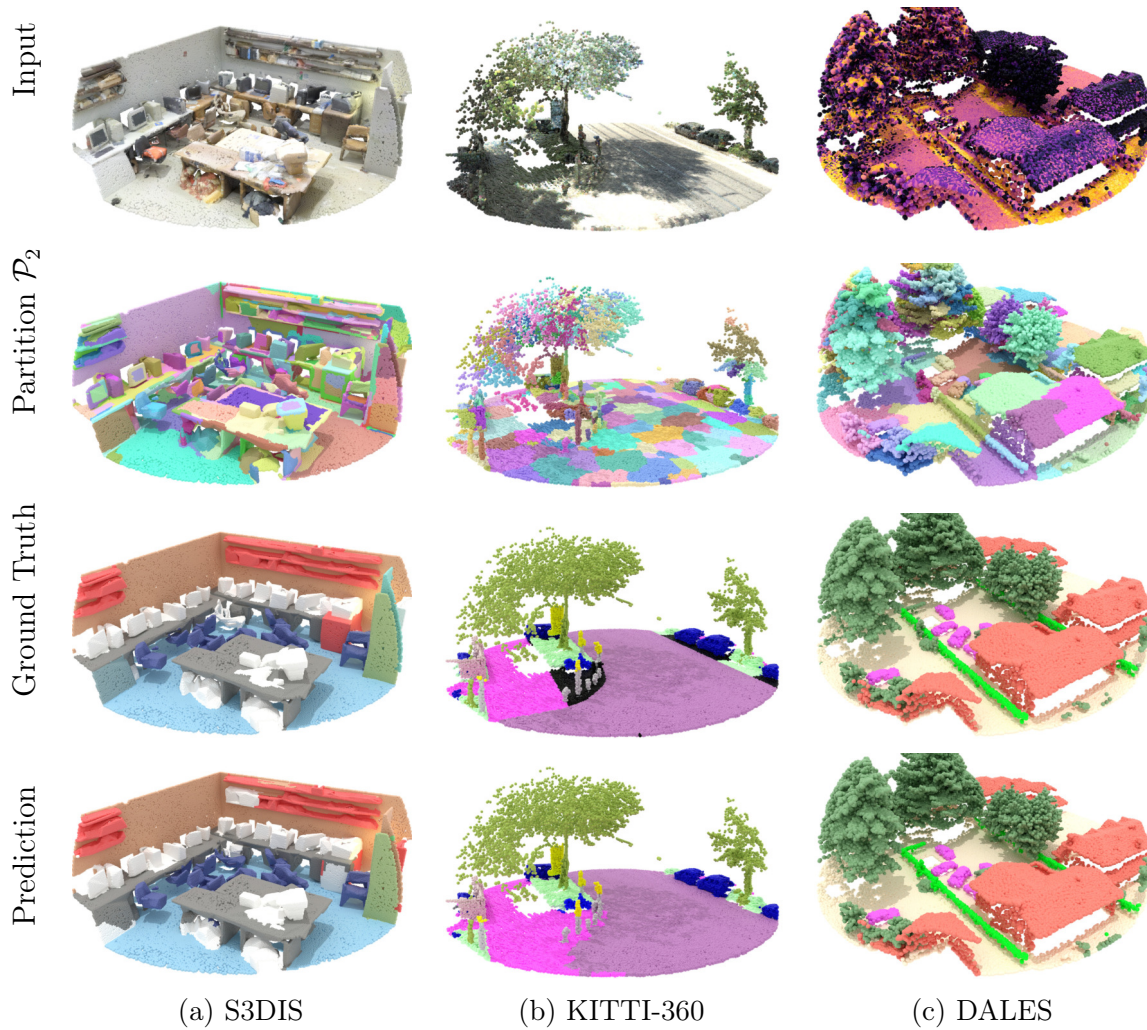


Figure 3.5 – **Qualitative Results.** We represent input samples (with color or intensity) of our approach and its predictions for all three datasets. Additionally, we show the coarsest partition level and demonstrate how superpoints can accurately capture the contours of complex objects and classify them accordingly. Black points are unlabeled in the ground truth. Color legend given in Section A-1.

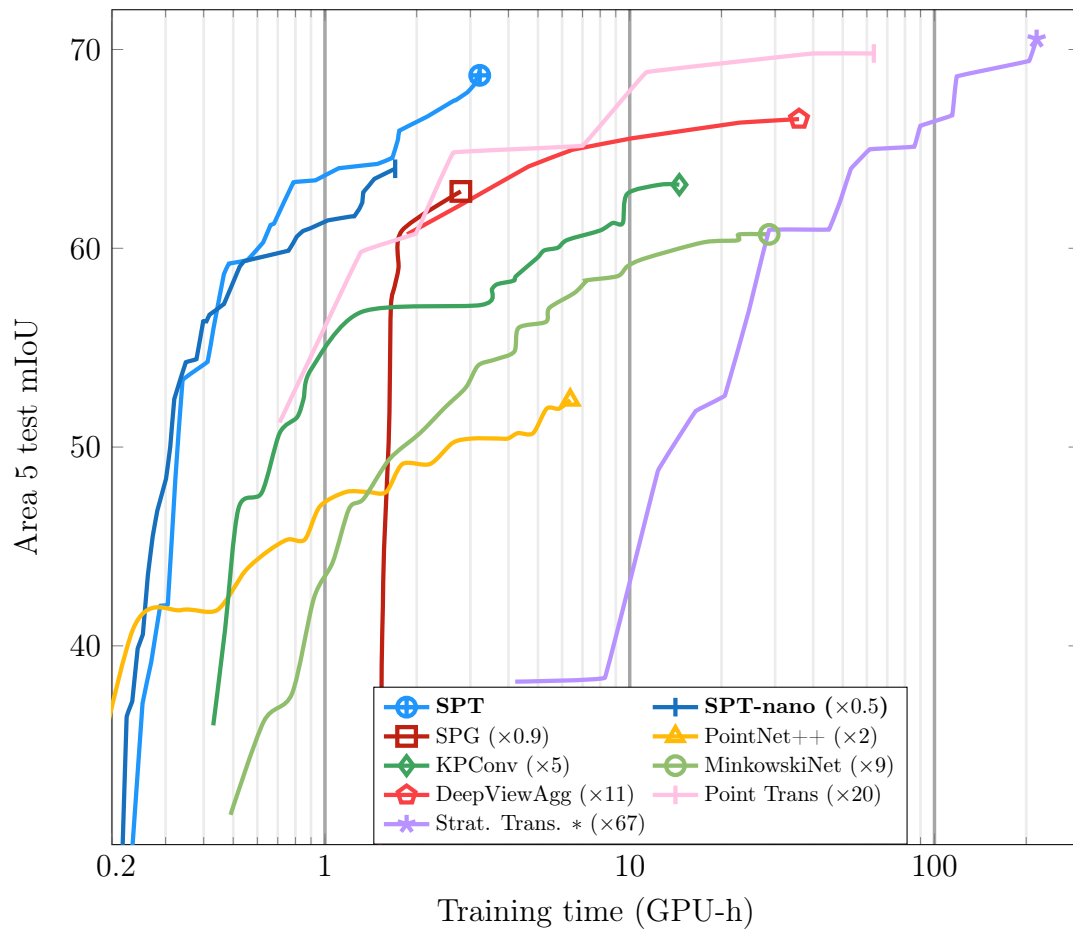


Figure 3.6 – **Training Speed.** We report the evolution of the test mIoU for S3DIS Area 5 for different methods *until the best epoch is reached*. The curves are shifted right according to the preprocessing time. We report in parenthesis the time ratio compared to SPT.

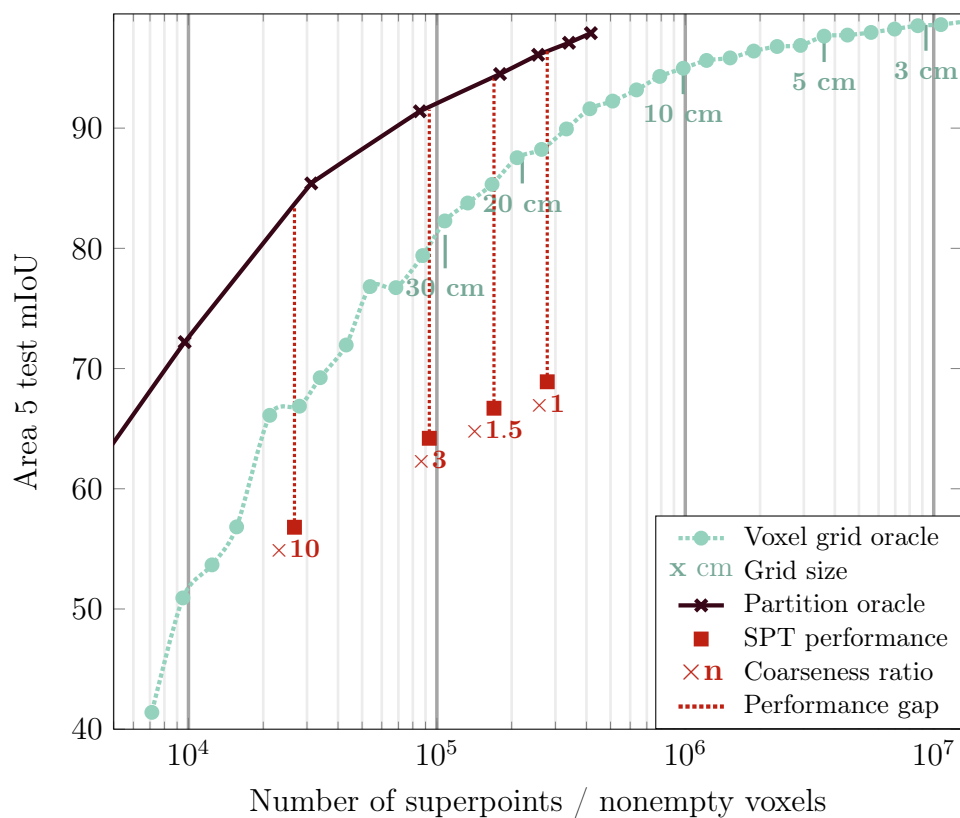


Figure 3.7 – **Partition Purity**. We plot the highest achievable “oracle” prediction for our partitions and a regular voxel grid. We also show the performance of SPT for 4 partitions with a coarseness ratio from $\times 1$ to $\times 10$. The significant performance gap between SPT and the “oracle” suggests model design is more limiting than partition purity.

Chapter 4

Efficient and Scalable 3D Panoptic Segmentation

Abstract

We introduce a highly efficient method for panoptic segmentation of large 3D point clouds by redefining this task as a scalable graph clustering problem. This approach can be trained using only local auxiliary tasks, thereby eliminating the resource-intensive instance-matching step during training. Moreover, our formulation can easily be adapted to the superpoint paradigm, further increasing its efficiency. This allows our model to process scenes with millions of points and thousands of objects in a single inference. Our method, called SuperCluster, achieves a new state-of-the-art panoptic segmentation performance for two indoor scanning datasets: 50.1 PQ (+7.8) for S3DIS Area 5, and 58.7 PQ (+25.2) for ScanNetV2. We also set the first state-of-the-art for two large-scale mobile mapping benchmarks: KITTI-360 and DALES. With only 209k parameters, our model is over 30 times smaller than the best-competing method and trains up to 15 times faster. Our code and pretrained models are available at github.com/drprojects/super_cluster.

This chapter’s work was initially presented in: Damien Robert, Hugo Raguet, Loic Landrieu, “Scalable 3D Panoptic Segmentation As Superpoint Graph Clustering”, 3DV, 2024.

4.1 Introduction

Understanding large-scale 3D environments is pivotal for numerous high-impact applications such as the creation of “digital twins” of extensive industrial facilities [257, 238, 153] or even the digitization of entire cities [173, 341, 239]. Extensive and comprehensive 3D analysis methods also benefit large-scale geospatial analysis, *e.g.* for land [343, 289] or forest surveys [339, 116], as well as building inventory [320] for country-scale mapping. These problems call for scalable models that can process large point clouds with millions of 3D points, accurately predict the semantics of each point, and recover all instances of specific objects, a task referred to as 3D panoptic segmentation [162].

Most existing 3D panoptic segmentation methods focus on sparse LiDAR scans for autonomous navigation [15, 89, 365]. Given the relevance of large-scale analysis for industry and practitioners, there is surprisingly little work on large-scale 3D panoptic segmentation [337]. Although they contain non-overlapping instance labels, S3DIS [13] and ScanNet [66] only have a few panoptic segmentation entries, and KITTI-360 [197] and DALES [311, 288] currently have none.

Large-scale 3D panoptic segmentation is particularly challenging due to the sheer scale of the scenes, often featuring millions of 3D points, and the diversity in objects—ranging from a few to thousands and with extreme size variability. Current methods typically rely on large backbone networks with millions of parameters, restricting their analysis to small scenes or portions of scenes due to their high memory consumption. Furthermore, training these models requires resource-intensive procedures, such as non-maximum suppression and instance matching. These costly operations prevent the analysis of large scenes with many points or objects. Most methods also require a pre-set limit



Figure 4.1 – **Large-Scale Panoptic Segmentation.** We present the results of Super-Cluster for the entire Area 5 of S3DIS [14] (ceiling removed for visualization) with 9.2M points (78M pre-subsampling) and 1863 true “things” objects. Our model can process such large scan in one inference on a single V100-32GB GPU in 3.3 seconds and reach a state-of-the-art PQ of 46.3.

on the number of detectable objects, introducing unnecessary complexity and the risk of missing objects in large scenes. Although recent mask-based instance segmentation methods [280] have demonstrated high performance and versatility, they fail to scale effectively to large scenes, as they predict a binary mask that covers the entire scene for each proposed instance.

To address these limitations, we present Super-Cluster, a novel approach for large-scale and efficient 3D panoptic segmentation. Our model differs from existing methods in three main ways:

- **Scalable graph clustering:** We view the panoptic segmentation task as a scalable graph clustering problem, which can be resolved efficiently at a large scale without setting the number of predicted objects in advance.

- **Local supervision:** We use a neural network to predict the parameters of the graph clustering problem and supervise with auxiliary losses that do not require an actual segmentation. This allows us to avoid resource-intensive non-maximum suppression or instance-matching steps.
- **Superpoint-only segmentation:** Our approach can easily be adapted to a superpoint-based approach. Feature computation, supervision, and prediction are entirely conducted at the superpoint level and never individual points, starkly decreasing their complexity.

These features make SuperCluster particularly resource-efficient, fast, and scalable, while ensuring high precision, as shown in Figure 4.1. Our primary contributions are:

- **Large-scale panoptic segmentation:** SuperCluster significantly improves the panoptic segmentation state-of-the-art for two indoor scanning datasets: 50.1 PQ (+7.8) on S3DIS Fold5 [14], and 58.7 PQ (+25.2) on ScanNetV2 [66]. We also set the first panoptic state-of-the-art for S3DIS 6-fold and two large-scale benchmarks (KITTI-360 [197] and DALES [311, 288]).
- **Fast and scalable segmentation:** SuperCluster contains only 209k trainable parameters (205k in the backbone), yet outperforms networks over 30 times larger. SuperCluster inference is also as fast as the fastest instance segmentation methods and trains up to 15-times faster: 4 h for one S3DIS fold and 6 h for ScanNet.

4.2 Related Work

The panoptic segmentation of point clouds with millions of points has received little attention from the 3D computer vision community. In this chapter, we aim to address this gap.

Over the last few years, deep learning approaches for 3D point clouds have garnered considerable interest [110]. Autonomous driving, in particular, has been the focus of numerous studies, resulting in multiple proposed approaches for object detection [351, 7], as well as semantic [366, 209, 355], instance [361, 363], and panoptic segmentation [15, 89, 365, 225]. However, these methods consider sequences of sparse LiDAR acquisition, and focus on a small set of classes (pedestrians, cars).

For the panoptic segmentation of dense LiDAR point clouds, the volume of research is surprisingly small [337]. A limited number of studies have addressed the panoptic segmentation of indoor spaces using RGB-D images [333, 231]. Dense scans have primarily been used in the context of instance segmentation [280, 234, 122, 344, 152, 317]. However, while this task is related to panoptic segmentation, these methods often adopt specific strategies to maximize instance segmentation metrics [337, 54]. Moreover, many methods require specifying the maximum number of predicted instances in advance, a constraint that proves inefficient for small scenes and results in missing objects in large scenes. Additionally, when implementing a sliding-window strategy, the predicted instances must be stitched together using either heuristic techniques or resource-intensive post-processing. Lastly, the best-performing methods [280, 234] rely on a computationally expensive matching step between the predicted and true instances [44, 344, 151]. This process often depends on the Hungarian algorithm, which has cubic complexity in the number of objects and, therefore, cannot scale to large scenes.

The strategy of partitioning large 3D point clouds into groups of adjacent and homogeneous points, called superpoints, has been used successfully for point cloud oversegmentation [177, 243, 200], semantic segmentation [180, 267, 140], and object detection [113, 80]. Our approach shares similarities with some superpoint-based approaches for 3D instance segmentation [294,

[196]. However, these methods are limited in scalability due to their reliance on point-wise encoders. Furthermore, the work by Sun *et al.* [294] employs a Hungarian-type instance matching scheme and allocates a binary mask to each predicted instance, covering the entire scene and drastically limiting the number of detected instances. Liang *et al.* [196] resort to quadratic-complexity agglomerative clustering to merge superpoints, and heavy post-processing for refining and scoring superpoints. In contrast, our method employs a fast graph clustering approach [179, 165], which does not require any instance matching or post-processing steps.

4.3 Method

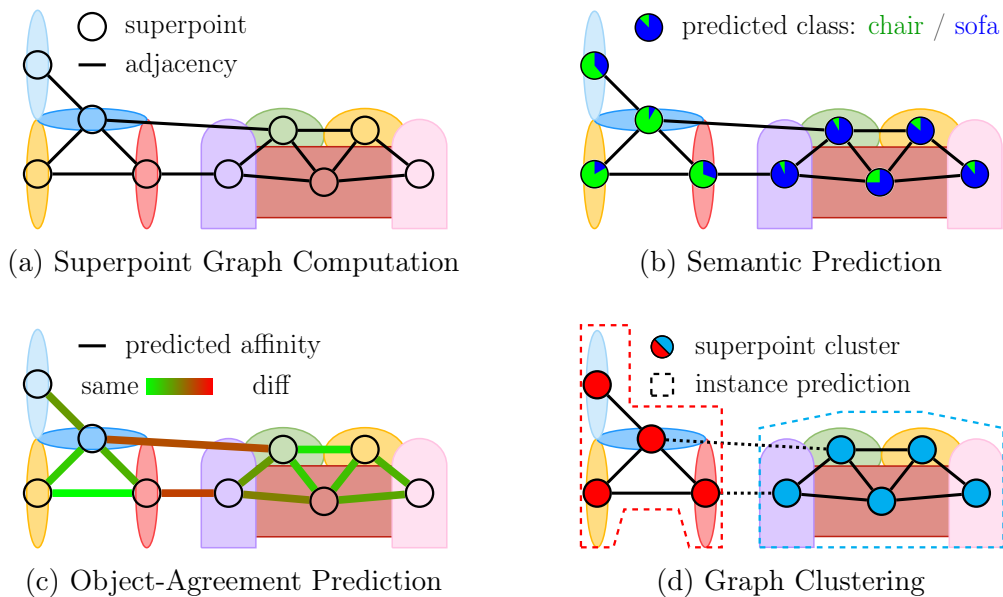


Figure 4.2 – **SuperCluster**. We illustrate the sequence of operations of SuperCluster for a simplified scene with two objects: a chair and a sofa. (a) showcases the first stage of our process, where the point cloud is partitioned into connected *superpoints* with simple geometric shapes. In (b), we predict a semantic class distribution for each superpoint. In (c), we predict the object agreement for each pair of adjacent superpoints, indicating the likelihood that they belong to the same object. Finally, (d) showcases the output of a graph clustering problem which merges superpoints with compatible class distribution and object agreement while cutting edges at the transition between objects. The resulting superpoint clusters define the instances of a panoptic 3D segmentation

Our objective is to perform panoptic segmentation of a large 3D point

cloud \mathcal{P} with potentially numerous and broad objects. For clarity, we first present our graph clustering formulation at the point level. We then explain how our approach can be supervised purely with local objectives, making its training particularly efficient. Finally, we detail how our method can be easily generalized to superpoints to further increase its scalability. Our final pipeline is illustrated in Figure 4.2.

Problem Statement. Consistently with the image panoptic segmentation setup [162], each point $p \in \mathcal{P}$ is associated with its position, a semantic label $\text{cls}(p) \in [1, C]$ with C the total number of classes, and an object index $\text{obj}(p) \in \mathbb{N}$. Points identified with a “thing” label (*e.g.* chair, car) are given an index uniquely identifying this object. Conversely, points with a “stuff” label (*e.g.* road, wall) are assigned an index *shared by all points with the same class* within \mathcal{P} . Our goal is to recover the class and object index of all points in \mathcal{P} .

4.3.1 Panoptic Segmentation as Graph Clustering

We propose viewing the panoptic segmentation task as grouping adjacent points with compatible class and object predictions. We formulate this task as an optimization problem structured by a graph. Specifically, we connect the points of \mathcal{P} to their K nearest neighbors, forming a graph $\mathcal{G} = (\mathcal{P}, \mathcal{E})$ where $\mathcal{E} \subset \mathcal{P} \times \mathcal{P}$ denotes these connections.

Spatial-Semantic Regularization. We use a neural network to associate each point p with a probabilistic class prediction $x_p^{\text{class}} \in [0, 1]^C$. The architecture and supervision of this network are detailed in Section 4.3.2. A simple way to obtain a panoptic segmentation would be to group spatially adjacent points with the same class prediction $\arg \max_c x_{p,c}^{\text{class}}$. However, this approach ignores the structure of objects, and thus would lead to two types of issues: adjacent

but distinct objects of the same class might be erroneously merged, and the probabilistic nature of the prediction x^{class} may lead to unwanted object fragmentation.

To tackle this last issue, we aim to enforce the spatial consistency of the object prediction. We introduce the signal x , defined for each point p as the channelwise concatenation of its position x_p^{pos} and its semantic prediction: $x_p = [x_p^{\text{class}}, x_p^{\text{pos}}]$. We propose to compute a piecewise-constant approximation y^* of x with an energy minimization problem regularized by the graph cut [27] between its constant components [181]. This approach aligns with well-established practices in 2D [182, 227] and 3D [172] analyses, and leads to the following optimization problem:

$$y^* = \arg \min_{y \in \mathbb{R}^{(C+3) \times |\mathcal{P}|}} \sum_{p \in \mathcal{P}} d(x_p, y_p) + \lambda \sum_{(p,q) \in \mathcal{E}} w_{p,q} [y_p \neq y_q], \quad (4.1)$$

where $[a \neq b] := 0$ if $a = b$ and 1 otherwise, $\lambda > 0$ is a parameter controlling the regularization strength, and $w_{p,q}$ is a nonnegative weight associated with edge (p, q) , see below.

The dissimilarity function d takes into account both the spatial and semantic nature of x :

$$d(x_p, y_p) = H(y_p^{\text{class}}, x_p^{\text{class}}) + \eta \|x_p^{\text{pos}} - y_p^{\text{pos}}\|^2, \quad (4.2)$$

where y_p^{class} is the first C coordinates of y_p and y_p^{pos} the last 3, and $\eta \geq 0$ a parameter. The term $H(x, y)$ denotes the cross-entropy between two distributions: $H(x, y) = -\sum_{c=1}^C y_c \log(x_c)$.

Object-Guided Edge Weights. The edge weight $w_{p,q}$ determines the cost of predicting an object transition between p and q . Designing appropriate edge weights is critical to differentiate between objects of the same class that are spatially adjacent, such as rows of chairs or cars in traffic. Edge weights

should encourage cuts along probable object transitions and prevent cuts within objects.

To facilitate this, we propose to train a neural network to predict an object agreement $a_{p,q} \in [0, 1]$ for each edge (p, q) in \mathcal{E} . This value represents the probability that both points belong to the same object. We then determine the edge weight $w_{p,q} \in [0, \infty]$ as follows:

$$w_{p,q} = a_{p,q} / (1 - a_{p,q} + \epsilon) , \quad (4.3)$$

with $\epsilon > 0$ a fixed parameter. High values of $w_{p,q}$ discourage cuts between points p and q that are confidently predicted to belong to the same object. Conversely, a smaller $w_{p,q}$ means that cuts between edges with a probable transition $a_{p,q}$ are not heavily penalized.

Graph Clustering. The constant components of the solution y^* of Equation 4.1 define a clustering \mathcal{K} of \mathcal{P} . The clusters \mathcal{K} contain spatially adjacent points with compatible semantics, and their contours should follow predicted object transitions.

Converting to a Panoptic Segmentation. We can derive a panoptic segmentation from the clusters \mathcal{K} . For each cluster, we calculate the average point distribution of its constituent points and select the class with the highest probability. We then associate a unique object index to each cluster k predicted as a “thing” class. Likewise, we assign to each cluster classified as “stuff” an index shared by all clusters predicted as the same class. Finally, each individual point is labeled with the class and object index of its respective cluster.

Optimization. The optimization problem expressed in Equation 4.1 is widely explored within graph optimization literature. Referred to as the *generalized*

minimal partition problem [179], this problem is related to the Potts models [251] and image partitioning techniques [182, 227]. We adapt the parallel ℓ_0 -cut pursuit algorithm [261, 181] to the dual spatial-semantic nature of the regularized signal. The resulting algorithm is particularly scalable and can handle graphs with hundreds of millions of edges on a standard workstation. This allows us to process large point clouds in one inference without the need for tiling and instance stitching post-processing.

4.3.2 Local Supervision

A major benefit of our approach is that it can be entirely supervised with local auxiliary tasks: all losses described in this section are sums of simple functions depending on one or two points at the time. In particular, we bypass the computationally expensive step of matching true instances with their predicted counterparts.

Recall from Section 4.3.1 that we can obtain a panoptic segmentation by predicting the parameters of a graph clustering problem: the semantic predictions x_p^{class} and the object agreements $a_{p,q}$. These quantities are both derived from a common pointwise embeddings $\{e_p\}_{p \in \mathcal{P}}$, computed by a neural network.

Predicting Semantics. We predict the class distribution $x_p^{\text{class}} = \text{softmax}(\phi^{\text{class}}(e_p))$ with ϕ^{class} a Multi-Layer Perceptron (MLP). This distribution is supervised by its cross-entropy against the true class $\text{cls}(p)$:

$$\mathcal{L}_p^{\text{class}} = H(x_p^{\text{class}}, \mathbf{1}(\text{cls}(p))) , \quad (4.4)$$

with $\mathbf{1}(c) \in \{0, 1\}^C$ the one-hot embedding of class c .

Predicting Object Agreement. To predict the object agreement $a_{p,q}$ between two adjacent points $(p, q) \in \mathcal{E}$, we employ an MLP ϕ^{object} whose input is a

symmetric combination of the points' embedding vectors:

$$a_{p,q} = \text{sigmoid} \left(\phi^{\text{object}} \left(\left[\frac{1}{2}(e_p + e_q), |e_p - e_q| \right] \right) \right), \quad (4.5)$$

where $|\cdot|$ refers to the termwise absolute value. The true object agreement $\hat{a}_{p,q}$ is assigned the value of 1 if $\text{obj}(p) = \text{obj}(q)$ and 0 otherwise. The prediction of $a_{s,t}$ can be seen as a *binary edge classification problem* as inter- and intra-object edges [177], and is supervised with the cross-entropy between true and predicted object agreements:

$$\mathcal{L}_{p,q}^{\text{object}} = H(\text{Bern}(a_{p,q}), \text{Bern}(\hat{a}_{p,q})) , \quad (4.6)$$

where $\text{Bern}(a)$ denote the Bernoulli distribution parametrized by $a \in [0, 1]$.

Loss Function. We combine the two losses above into a single objective \mathcal{L} :

$$\mathcal{L} = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \mathcal{L}_p^{\text{class}} + \frac{1}{|\mathcal{E}|} \sum_{(p,q) \in \mathcal{E}} \mathcal{L}_{p,q}^{\text{object}}, \quad (4.7)$$

with $|\mathcal{E}|$ and $|\mathcal{P}|$ the total number of edges and 3D points, respectively.

4.3.3 Extension to Superpoints

In this section, we discuss the extension of our method to a superpoint-based approach for enhanced scalability.

Motivation. We aim to design a panoptic segmentation method that can scale to large 3D point clouds. While the formulation presented in the previous section is advantageous, it still requires computing embeddings and predictions for each individual point, which can be memory-intensive and limits the volume of data that can be processed simultaneously. We propose to group adjacent points with similar local geometry and color into *superpoints*,

and to only compute embeddings and predictions for superpoints and not individual points. By doing so, we drastically reduce the computational and memory requirements of our method, enabling it to handle larger 3D point clouds at once. Our pipeline is illustrated in Figure 4.2.

Computing Superpoints. We partition the point cloud \mathcal{P} into a set of non-overlapping superpoints \mathcal{S} . We use the superpoint partition method implemented by Robert *et al.* in SPT [267], which defines the superpoints as the constant components of a low-surface piecewise constant approximation of geometric and radiometric point features.

Although the superpoints \mathcal{S} form a high-purity oversegmentation of \mathcal{P} , some superpoints can span multiple objects. To account for this, we associate each superpoint s with its *majority-object* $\text{obj}(s)$ defined as the most common object index within its points: $\text{obj}(s) = \text{mode}\{\text{obj}(p) \mid p \in s\}$. Likewise, we define $\text{cls}(s) = \text{mode}\{\text{cls}(p) \mid p \in s\}$.

Adapting Graph Clustering. Our clustering step can be directly adapted by substituting the point set \mathcal{P} with the superpoint set \mathcal{S} , and defining the graph \mathcal{G} by connecting superpoints with adjacent points following the approach of SPT [267]. We replace the point position x_p^{pos} by the coordinates of the superpoints' centroids x_s^{pos} . All other steps are unchanged.

Superpoint Embedding. We use a superpoint-embedding network to assign compute the superpoint features e_s for $s \in \mathcal{S}$. We employ the SuperPoint Transformer model [267] for its efficiency and ability to leverage large spatial context.

Superpoint Semantic Supervision. We supervise the semantic superpoint prediction x_s^{class} with Equation 4.4 where we replace $\text{cls}(p)$ with $\text{cls}(s)$.

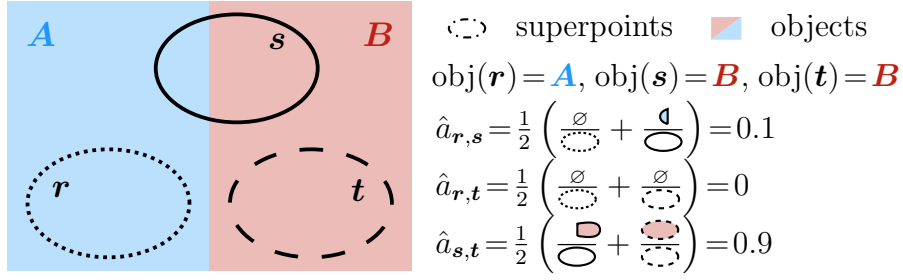


Figure 4.3 – **Superpoint Object Agreement.** We compute for each pair of adjacent superpoint (s, t) an object agreement score $\hat{a}_{s,t}$. This value is defined by the average overlap ratio between s and t and their majority-objects $\text{obj}(t)$ and $\text{obj}(s)$, see Equation 4.8.

Superpoint Object Agreement Supervision. While the true object agreement $\hat{a}_{p,q}$ between two points is binary, the agreement between superpoints spans a continuum. As illustrated in Figure 4.3, we quantify this agreement as:

$$\hat{a}_{s,t} = \frac{1}{2} \left(\frac{|s \cap \mathcal{P}_{|\text{obj}(t)|}|}{|s|} + \frac{|t \cap \mathcal{P}_{|\text{obj}(s)|}|}{|t|} \right), \quad (4.8)$$

where $\mathcal{P}_{|o|} := \{p \in \mathcal{P} \mid \text{obj}(p) = o\}$ is the set of points of \mathcal{P} with the object index o , and $|s|$ is the count of 3D points in s . We can now supervise the predicted object agreement $a_{s,t}$ with Equation 4.6 unchanged.

4.4 Experiments

We first present the datasets and metrics used for evaluation in Section 4.4.1, then our main results and their analysis in Section 4.4.2, and finally an ablation study in Section 4.4.3.

4.4.1 Datasets and Metrics

Datasets. We present the four datasets used in this chapter.

- **S3DIS** [14]: This indoor scanning dataset consists of 274 million points distributed across 271 rooms in 6 building floors—or areas. We do not use the provided room partition, as they require significant manual processing and may not translate well to other environments such as

open offices, industrial sites, or mobile mapping. Instead, we merge all rooms in the same area and treat each floor as one single large-scale acquisition [301, 46].

We follow the standard evaluation protocol, using the area 5 as a test set and implementing 6-fold cross-validation. In line with Xiang *et al.*'s [337] proposal, we treat all 13 classes as “thing”. However, certain classes, such as *walls*, *ceiling*, and *floors*, are susceptible to arbitrary division due to room splitting, making their evaluation somewhat inconsistent. As a result, we also present panoptic metrics where these three classes are considered as “stuff”.

- **ScanNet**. [66] This dataset consists of 237M 3D points organized in 1501 medium-scale indoor scenes. We evaluate SuperCluster on ScanNet’s open test set, as the hidden test set is not evaluated for panoptic segmentation. We use for “things” the class evaluated in the instance segmentation setting: *bathtub*, *bed*, *bookshelf*, *cabinet*, *chair*, *counter*, *curtain*, *desk*, *door*, *other furniture*, *picture*, *refrigerator*, *shower curtain*, *sink*, *sofa*, *table*, *toilet*, and *window*. The *walls* and *floor* class are designated as “stuff”.
- **KITTI-360** [197]: Containing over 100k mobile mapping laser scans from an outdoor urban environment, we utilize the *accumulated point clouds* format, which aggregates multiple sensor rotations to form 300 extensive scenes with an average of more than 3 million points. We train on 239 scenes and evaluate it on the remaining 61. *Building* and *cars* classes are treated as “thing” while the remaining 13 are classified as “stuff”.
- **DALES** [311, 288]. This large-scale aerial scan data set spans 10 km² and contains 500 millions of 3D points organized along 40 urban and rural scenes, of which we use 12 for evaluation. The “thing” classes

are *buildings, cars, trucks, power lines, fences, and poles*. *Ground and vegetation* evaluated as “stuff”.

Evaluation Metrics. Recognition Quality (RQ) assesses a model’s capacity to identify and classify objects. Segmentation Quality (SQ) evaluates the alignment between an object’s true and predicted segmentation. Panoptic Quality (PQ) combines both measures by computing their product. We also compute the semantic segmentation performance of our method by associating each point with the class of the superpoint to which it belongs and measuring the class-averaged Intersection over Union (mIoU).

Model Parameterization. Our backbone for the S3DIS and DALES datasets is a small SPT-64 model [267] with 205k parameters. We use a larger SPT-128 (791k parameters) for KITTI-360 and a slightly modified model for ScanNet (1M) parameters. SuperCluster adds two small MLP ϕ^{class} and ϕ^{object} for a total of 4.4k parameters for S3DIS and DALES, and 8.8k parameters for KITTI-360 and ScanNet.

Our training batches are composed of 4 randomly sampled cylinders with a radius of 7 m for S3DIS, and 50 m for KITTI-360 and DALES, and entire scenes for ScanNet. The partition parameters are adjusted so that $\mathcal{S}/\mathcal{P} \sim 30$ for S3DIS, DALES, and KITTI-360, and 20 for ScanNet.

Graph Clustering. We can tune the graph clustering parameters post-training to optimize the PQ on the training set: λ in Equation 4.1, η in Equation 4.2, and ϵ in Equation 4.3. As the clustering step is particularly efficient, we can evaluate tens of values in a few minutes. We detail in Table 4.1 the tuned parameters for each dataset.

Table 4.1 – **Graph Clustering Parameters.** We provide the graph clustering parameters used for each dataset.

Dataset	λ	η	ϵ
S3DIS	10	$5 \cdot 10^{-2}$	10^{-4}
S3DIS - no “stuff”	20	$5 \cdot 10^{-2}$	10^{-4}
ScanNet	20	$5 \cdot 10^{-2}$	10^{-4}
KITTI-360	10	$5 \cdot 10^{-2}$	10^{-4}
DALES	20	$5 \cdot 10^{-2}$	10^{-4}

4.4.2 Results and Analysis

We compare our method quantitatively with state-of-the-art models in Table 4.2 to 4.6. We also report a runtime analysis in Table 4.7 and qualitative illustrations in Figure 4.4.

S3DIS. We report in Table 4.2 the performance of our algorithm evaluated for Area 5 of the S3DIS dataset. Compared to several baselines for panoptic segmentation, our model shows a notable improvement with a PQ boost of +7.8 points and a mIoU increase of +3.2 points. Remarkably, our model is more than 33 times smaller than the highest performing model. Furthermore, we compute panoptic metrics by treating *wall*, *ceiling*, and *floor* as “stuff” classes to account for their arbitrary boundaries.

In Table 4.3, we present the same metrics evaluated with 6-Fold cross-validation. This classic setting is typically used to evaluate semantic segmentation; however, we are the first to report panoptic results in this context.

Despite its smaller size, our model achieves high semantic segmentation performance, improving the Area 5 performance of SPT by 1 point and reaching near state-of-the-art performance on the 6-fold evaluation.

ScanNet. As shown in Table 4.4, SuperCluster significantly improves the state-of-the-art of panoptic segmentation by 25.2 PQ points. Our model does

Table 4.2 – **S3DIS Area 5**. We report the semantic (*SS*) and panoptic segmentation results of the top-performing semantic segmentation methods on the fifth area of S3DIS, as well as panoptic segmentation approaches implemented by Xiang *et al.* [337]. We provide two panoptic metrics by considering all classes as “things” (*PS - no “stuff”*) and with *wall*, *ceiling* and *floor* as “stuff” (*PS*).

	size	SS	PS - no “stuff”			PS		
	$\times 10^6$	mIoU	PQ	RQ	SQ	PQ	RQ	SQ
Semantic segmentation models								
SPT [267]	0.21	68.9	-	-	-	-	-	-
Point Trans.[359]	7.8	70.4	-	-	-	-	-	-
PointNeXt-XL [255]	41.6	71.1	-	-	-	-	-	-
Strat. Trans. [175, 323]	8.0	72.0	-	-	-	-	-	-
Panoptic segmentation models								
Xiang <i>et al.</i> [337]	0.13							
+ PointNet++ [253]	+3.0	58.7	24.6	32.6	68.2	-	-	-
+ Minkowski [58]	+37.9	63.8	39.2	48.0	74.9	-	-	-
+ KPConv [301]	+14.1	65.3	41.8	51.5	74.7	-	-	-
PointGroup [207] in [337]	7.7	64.9	42.3	52.0	74.7	-	-	-
SuperCluster (ours)	0.21	68.1	50.1	60.1	76.6	58.4	68.4	77.8

not perform as well as large networks designed for semantic segmentation but provides decent results with a small backbone of just 1M parameters.

DALES and KITTI-360. To the best of our knowledge, SuperCluster is the first method capable of processing the large tiles of the DALES and KITTI-360 data sets at once, thus establishing the first panoptic state-of-the-art for these datasets given in Table 4.5 and Table 4.6.

Inference and Training Speed. In Table 4.7, we compare the inference speed of our approach with state-of-the-art instance and panoptic segmentation algorithms. Although we used our smallest GPU (a 1080Ti) to replicate the setting used to measure most of the approaches’ speed (a Titan-X), the

Table 4.3 – **S3DIS 6-Fold**. We report the 6-Fold cross-validated semantic and panoptic segmentation results on S3DIS. No panoptic methods were evaluated in this setting to the best of our knowledge.

	size	SS	PS - no “stuff”			PS		
	$\times 10^6$	mIoU	PQ	RQ	SQ	PQ	RQ	SQ
Semantic segmentation models								
DeepViewAgg [269]	41.2	74.7	-	-	-	-	-	-
Strat. Trans. [175, 323]	8.0	74.9	-	-	-	-	-	-
PointNeXt-XL [255]	41.6	74.9	-	-	-	-	-	-
SPT [267]	0.21	76.0	-	-	-	-	-	-
Panoptic segmentation models								
SuperCluster (ours)	0.21	75.3	55.9	66.3	83.8	62.7	73.2	84.8

Table 4.4 – **ScanNetv2 Val**. We report the Semantic Segmentation (*SS*) and Panoptic Segmentation (*PS*) performance for various methods on the open test set of ScanNetv2. † code and models unavailable.

	size	SS	PS		
	$\times 10^6$	mIoU	PQ	RQ	SQ
Semantic segmentation models					
KPConv [301]	14.1	69.2	-	-	-
Point Trans [359]	7.8	70.6	-	-	-
Point Trans. v2[334]	11.3	75.4	-	-	-
OctFormer [322]	44.0	75.7	-	-	-
Panoptic segmentation models					
SceneGraphFusion [333, 319]	2.9	-	31.5	42.2	72.9
Panoptic Fusion [231]	†	-	33.5	45.3	73.0
SuperCluster (ours)	1.0	66.1	58.7	69.1	84.1

values are not entirely comparable. Still, our model is on par with the fastest methods and offers superior scalability.

None of the reported runtimes include the method’s preprocessing times.

Table 4.5 – **KITTI-360** We report the Semantic Segmentation (*SS*) and Panoptic Segmentation (*PS*) performance for various methods on the open test set of KITTI-360. No panoptic methods were evaluated on this dataset to the best of our knowledge.

	size	SS	PS		
	$\times 10^6$	mIoU	PQ	RQ	SQ
Semantic segmentation models					
Minkowski [58]	37.9	58.3	-	-	-
DeepViewAgg [359]	41.2	62.1	-	-	-
SPT [267]	0.78	63.5	-	-	-
Panoptic segmentation models					
SuperCluster (ours)	0.79	62.1	48.3	58.4	75.1

Table 4.6 – **DALES** We report the Semantic Segmentation (*SS*) and Panoptic Segmentation (*PS*) performance for various methods on the open test set of DALES. No panoptic methods were evaluated on this dataset to the best of our knowledge.

	size	SS	PS		
	$\times 10^6$	mIoU	PQ	RQ	SQ
Semantic segmentation models					
ConvPoint [24]	4.7	67.4	-	-	-
PointNet++ [253]	3.0	68.3	-	-	-
SPT [267]	0.21	79.6	-	-	-
KPConv [301]	14.1	81.1	-	-	-
Panoptic segmentation models					
SuperCluster (ours)	0.21	77.3	61.2	68.6	87.1

Thanks to SPT’s efficient implementation, our entire preprocessing, including the superpoint partition, is faster or equivalent to all existing 3D segmentation methods. For instance, preprocessing the entirety of S3DIS (271 rooms) takes only 12 minutes with an A40 GPU [267].

Our model can be trained in an amount of time comparable to its backbone SPT for semantic segmentation [269]. One fold of S3DIS takes just under

4 hours, which is substantially quicker than most existing semantic, instance, or panoptic segmentation models. For instance, PointTransformer [359] trains for 63h and Stratified Transformer [175] 216 GPU-h. SuperCluster trains on 6 h on ScanNet, compared to 78 h for Mask3D [280] and 20 h for ISBNet [234].

Table 4.7 – **Runtime.** We compare the speed of our model to various instance and panoptic segmentation models. We report the time spent in the backbone network (first number) and performing panoptic segmentation (second number) on ScanNet Val. scans. ★ optional CRF post-processing.

	hardware	runtime in ms
Instance segmentation methods		average per scan
PointGroup [152]	Titan X	452 = 128 + 324
SoftGroup [317]	Titan X	345 = 152 + 148
HAIS [49]	Titan X	339 = 154 + 185
Mask3D [280]	Titan X	339
ISBNet [234]	Titan X	237 = 152 + 85
SuperCluster (ours)	1080Ti	238 = 193 + 45
Panoptic segmentation methods		scan scene0645_01
PanopticFusion [231]	2×1080Ti	485 = 317 + 168 (+ 4500★)
SuperCluster (ours)	1080Ti	482 = 376 + 106

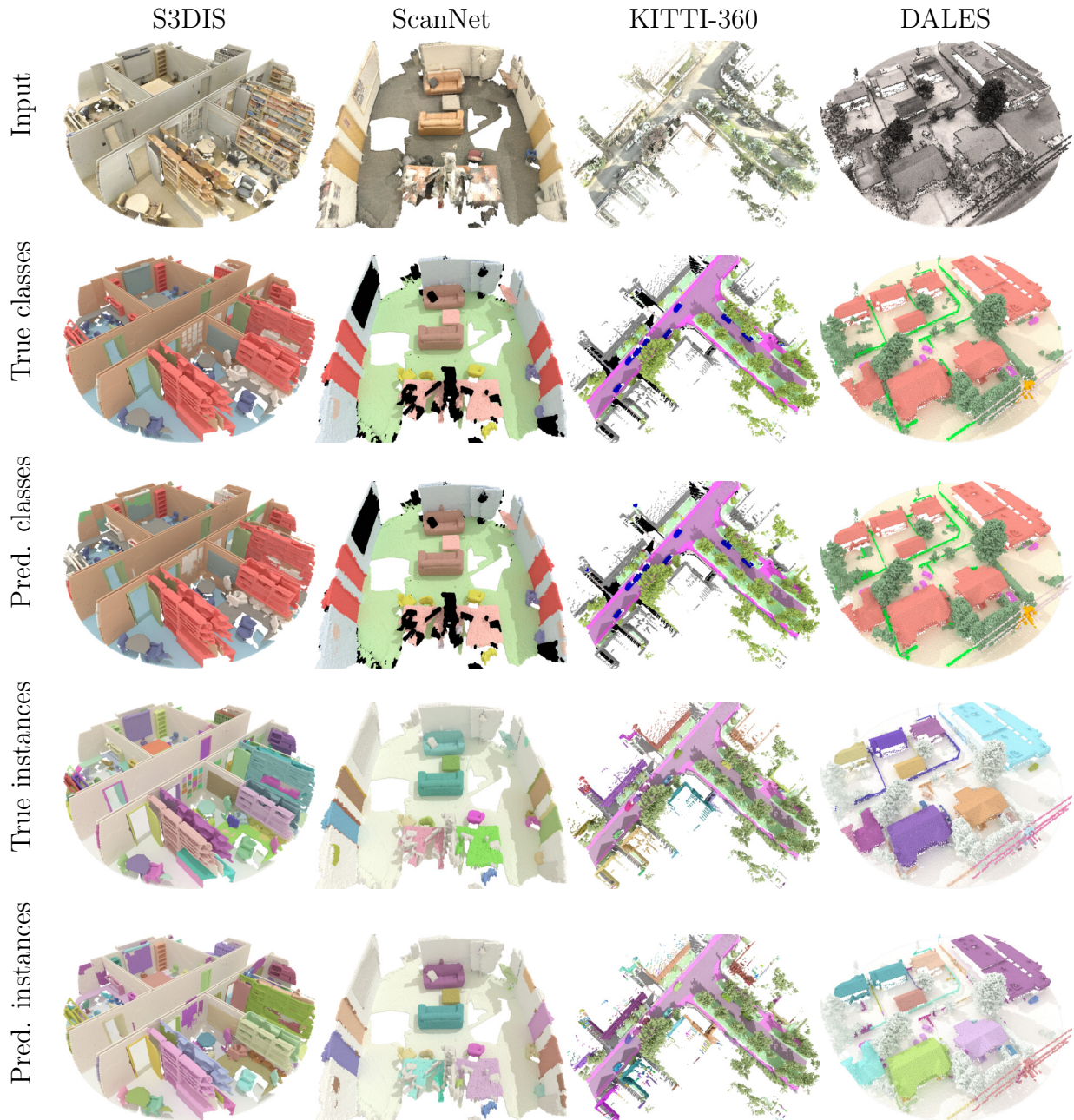


Figure 4.4 – **Qualitative Results.** We present the panoptic predictions of our model for the four considered datasets. The scenes’ size corresponds to a single batch item during training. “Stuff” classes are represented with a lower opacity. Color legend given in Section A-1.

4.4.3 Ablation Study

We evaluate the impact of our design choice by performing several experiments whose results are given in Table 4.8.

Constant Edge Weights. Replacing all edge weights with a constant value of 1 yields a drop of 4.2 PQ points. This experiment shows the benefit of learning object transitions.

Offset Prediction. Several *bottom-up* [123] segmentation approaches [152, 337, 174, 113] propose clustering points by shifting their positions towards the predicted position of the object centroid. To reproduce this strategy, we adjust the position of x_s^{pos} in x along a vector that predicts the center of the majority object. We supervise this prediction with the L1 loss, as it produced the best results among several alternatives that we examined. Despite our efforts, this approach did not improve the results: -1.3 PQ points. We attribute this to the size diversity of objects observed in large-scale scenes (corridors, buildings), resulting in an unstable prediction.

Smaller Superpoints. To demonstrate the benefits of using superpoints, we consider a finer partition with $\mathcal{S}/\mathcal{P} \sim 15$ instead of 30. This requires training with smaller 3 m cylinders instead of 7, decreasing the performance by -1.8 PQ points. This result illustrates that the superpoint paradigm is central to our approach.

Superpoint Oracle. Using superpoints greatly improves the efficiency and scalability of SuperCluster. However, since the predictions are made at the superpoint level and never for individual 3D points, the semantic and object purity of the superpoints can restrict the model’s performance. To evaluate this impact, we define the superpoint oracle, which assigns each superpoint s

the class and index of its majority object $\text{obj}(s)$. The resulting performance provides an upper bound of what our model could potentially achieve. The high performance of this oracle (93.4 PQ) indicates that very little precision is lost by working with superpoints.

Clustering Oracle. In a similar vein, we calculate the upper bound of our model by computing the results of the graph clustering with perfect network predictions: x^{class} is set as the one-hot-encoding of the class of the majority object, and the object agreement is set to its true value: $a_{p,q} = \hat{a}_{p,q}$. The performance of this oracle (83.6 PQ) shows that our scalable clustering formulation does not significantly compromise the model’s precision in its current regime.

Table 4.8 – **Ablation Study.** We report the performance of different experiments on S3DIS Area 5 with *wall*, *ceiling* and *floor* as “stuff”.

Experiment	PS		
	PQ	RQ	SQ
Best Model	58.4	68.4	77.8
Constant Edge Weights	54.2	64.2	76.6
Offset Prediction	57.1	65.2	77.1
Smaller Superpoints	56.6	64.6	78.6
Superpoint Oracle	93.4	99.7	93.7
Clustering Oracle	83.6	91.7	90.8

4.4.4 Large-Scale Inference

Our method can process large 3D point clouds with just one inference. In this section, we represent the largest portion of each dataset that SuperCluster can handle in one inference with an A40 GPU (48G of VRAM). Results for each dataset are presented in Figure 4.5, Figure 4.6, Figure 4.7, and Figure 4.8.

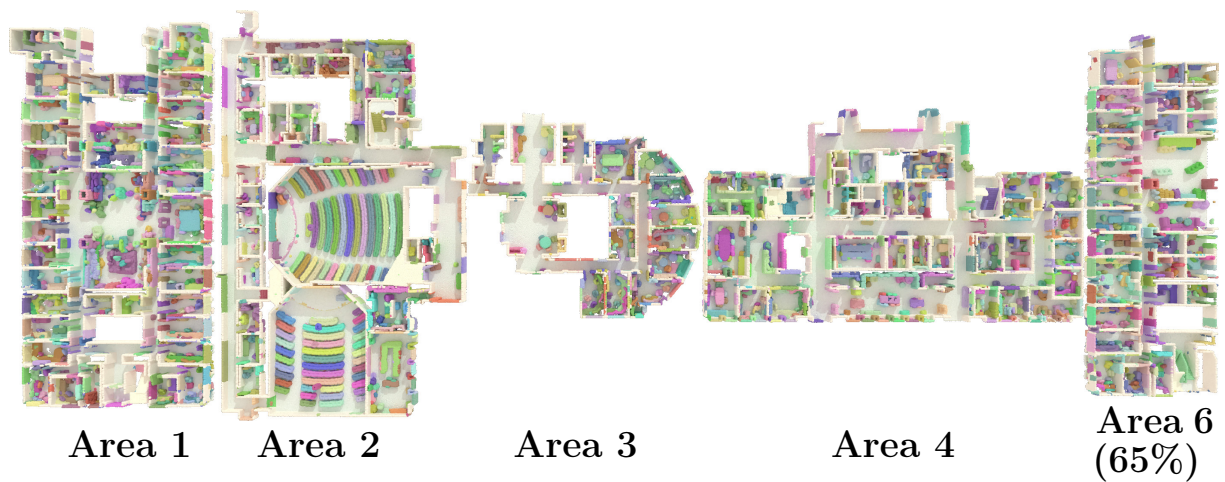


Figure 4.5 – **Large-Scale Inference on S3DIS**. Largest scan that SuperCluster can segment in one inference on an A40 GPU: **4.6** areas, **21.3m** points, **646k** superpoints, **5298** target objects, and **4565** predicted objects. Inference takes **7.4** seconds.



Figure 4.6 – **Large-Scale Inference on ScanNet**. Largest number of scans that SuperCluster can segment in one inference on an A40 GPU: **105** scans , **10.9m** points, **398k** superpoints , **1683** target objects, and **2148** predicted objects. The inference takes **6.8** seconds.



Figure 4.7 – **Large-Scale Inference on DALES.** Largest scan that SuperCluster can segment in one inference on an A40 GPU: **15.3** tiles, **7.8km²**, **18.0m** points, **589k** superpoints, **1727** target objects, and **1559** predicted objects. Inference takes **10.1** seconds.



Figure 4.8 – **Large-Scale Inference on KITTI-360.** Largest scan that SuperCluster can segment in one inference on an A40 GPU: **7.5 tiles**, **11.0m points**, **414k superpoints**, **602 target objects**, and **1947 predicted objects**. Inference takes **6.6 seconds**.

4.4.5 Limitations

Our approach, while efficient, is not devoid of constraints. The functional minimized in Equation 4.1 is noncontinuous and nondifferentiable. This hinders the computation of gradients and the possibility of learning the panoptic segmentation end-to-end. Nevertheless, this aspect lends itself to the speed and simplicity of our training process. Although our approach can run on diverse acquisition setups, the superpoint partition is sensitive to low point density and may fail for sparse scans as can be observed on the edge of some KITTI-360 acquisitions.

We use a lightweight SPT network to ensure maximum scalability. This network, while expressive, is not the most powerful existing architecture. There is a potential for improved results using more resource-intensive networks, especially for medium-scale datasets like ScanNet and S3DIS.

Local panoptic supervision does not translate into higher semantic segmentation performance in our experiments.

4.5 Conclusion

In this paper, we introduced SuperCluster, a novel approach for 3D panoptic segmentation of large-scale point clouds. We propose a new formulation of this task as a scalable graph clustering problem, bypassing some of the most compute-intensive steps of current panoptic segmentation methods. Our results across multiple benchmarks, including S3DIS, ScanNet, KITTI-360, and DALES, demonstrate that our model achieves state-of-the-art performance while being significantly smaller, scalable, and easier to train.

Despite the considerable industrial applications, large-scale panoptic segmentation has been relatively unexplored by the 3D computer vision community. We hope that our positive results and the state-of-the-art we established

on new datasets and settings will encourage the development of future panoptic approaches for large-scale 3D scans.

Acknowledgements. This work was funded by ENGIE Lab CRIGEN. This work was supported by ANR project READY3D ANR-19-CE23-0007, and was granted access to the HPC resources of IDRIS under the allocation AD011013388R1 made by GENCI. We thank Mathieu Brédif and Bruno Vallet for inspiring discussions and valuable feedback.

Chapter 5

Learning From Point Clouds and Images in the Wild

Abstract

Recent works on 3D semantic segmentation propose to exploit the synergy between images and point clouds by processing each modality with a dedicated network and projecting learned 2D features onto 3D points. Merging large-scale point clouds and images raises several challenges, such as constructing a mapping between points and pixels, and aggregating features between multiple views. Current methods require mesh reconstruction or specialized sensors to recover occlusions, and use heuristics to select and aggregate available images. In contrast, we propose an end-to-end trainable multi-view aggregation model leveraging the viewing conditions of 3D points to merge features from images taken at arbitrary positions. Our method can combine standard 2D and 3D networks and outperforms both 3D models operating on colorized point clouds and hybrid 2D/3D networks without requiring colorization, meshing, or true depth maps. We set a new state-of-the-art for large-scale indoor/outdoor semantic segmentation on S3DIS (74.7 mIoU 6-Fold) and on KITTI-360 (58.3 mIoU). Our full pipeline only requires raw 3D scans and a set of images and poses. Our code is publicly accessible at

<https://github.com/drprojects/DeepViewAgg>.

This chapter’s work was initially presented in: Damien Robert, Bruno Vallet, Loic Landrieu, “Learning Multi-View Aggregation in the Wild for Large-Scale 3D Semantic Segmentation”, CVPR, 2022.

5.1 Introduction



Figure 5.1 – **Combining 2D and 3D Information.** We propose to merge the complementary information between point clouds and a set of co-registered images. Using a simple visibility model, we can project 2D features onto the 3D points and use viewing conditions to select features from the most relevant images. We represent images at their position with the symbol \bullet and color the 3D points according to the image they are seen in.

The fast-paced development of dedicated neural architectures for 3D data has led to significant improvements in the automated analysis of large 3D scenes [110]. All top-performing methods operate on colorized point clouds, which requires either specialized sensors [331], or running a colorization step which is often closed-source [306, 83, 188] and sensor-dependent [154]. However, while colorized point clouds carry some radiometric information, images combined with dedicated 2D architectures are better suited for learning textural and contextual cues. A promising line of work sets out to leverage the complementarity between 3D point clouds and images by projecting onto 3D

points the 2D features learned from real [65, 135, 149] or virtual images [171, 55]

Combining point clouds and images with arbitrary poses (*i.e. in the wild*) as represented in Figure 5.1, involves recovering occlusions and computing a point-pixel mapping, which is typically done using accurate depth maps from specialized sensors [309, 40] or a potentially costly meshing step [25]. Furthermore, when a point is seen in different images simultaneously, the 2D features must be merged in a meaningful way. In the mesh texturation literature, multi-view aggregation is typically addressed by selecting images for each triangle based on their viewing conditions, *e.g.* distance, viewing angle, or occlusion [9, 189, 318]. Hybrid 2D/3D methods for large-scale point cloud analysis usually rely on heuristics to select a fixed number of images per point and pool their features uniformly without considering viewing conditions. Multi-view aggregation has also been extensively studied for shape recognition [86, 292, 329], albeit in a controlled and synthetic setting not entirely applicable to the analysis of large scenes.

In this work, we propose to learn to merge features from multiple images with a dedicated attention-based scheme. For each 3D point, the information from relevant images is aggregated based on the point’s viewing condition. Thanks to our GPU-based implementation, we can efficiently compute a point-pixel mapping without mesh or true depth maps, and without sacrificing precision. Our model can handle large-scale scenes with an arbitrary number of images per point taken at any position (with camera pose information), which corresponds to a standard industrial operational setting [128, 316, 248]. Using only standard 2D and 3D backbone networks, we set a new state-of-the-art for the S3DIS and KITTI-360 datasets. Our method improves on both standard and hybrid 2D/3D approaches without requiring point cloud colorization, mesh reconstruction, or depth sensors. In this work, we present

a novel and modular multi-view aggregation method for semantizing hybrid 2D/3D data based on the viewing conditions of 3D points in images. Our approach combines the following advantages:

- We set a new state-of-the-art for S3DIS 6-fold (74.7 mIoU), and KITTI-360 Test (58.3 mIoU) without using points’ colorization.
- Our point-pixel mapping operates directly on 3D point clouds and images without requiring depth maps, meshing, colorization, or virtual view generation.
- Our efficient GPU-based implementation handles arbitrary numbers of 2D views and large 3D point clouds.

5.2 Related Work

Point Cloud Colorization. One way of exploiting the complementarity between 3D point clouds and images is to colorize the points. Unlike photogrammetry-based [303] acquisition techniques which naturally produce colorized points, active sensors such as LiDAR [150] or time-of-flight cameras [240] do not. In practice, these clouds can be colorized through a nontrivial heuristics-based preprocessing requiring localized RGB images and their camera parameters [154]. Colorized point cloud datasets [13, 111, 66, 197] are frequently used for comparing 3D deep learning methods [252, 301, 58], which consistently perform better when radiometric information is available [255]. In short, point cloud colorization assumes either a specific sensor or heuristics-based preprocessing, and discards dense, contextual, multi-view information carried by images. Hence, 3D analysis methods capable of directly processing raw point clouds and localized images would be less hardware-dependent and more data efficient.

Attention-Based Modality Fusion. Methods using attention mechanisms to learn multi-modal representation have attracted a lot of attention, in particular for combining textual and visual information [38, 138, 107] as well as videos [129, 211]. Closer to our setting, Lu *et al.* [214] use an attention scheme to select the most relevant parts of an image for visual question answering. Li *et al.* [192] define a two-branch attention-based modality fusion network merging 2D semantic and 3D occupancy for scene completion. Such work confirms the relevance of using attention for learning multi-modal representations.

2D/3D Scene Analysis with Deep Learning. Over the last few years, deep networks specifically designed to handle the 3D modality have reached impressive degrees of performance and maturity, see the review of Guo *et al.* [110]. Recent work [65, 149, 135] propose to use a dedicated 3D network for processing point clouds, while a 2D convolutional network extracts radiometric features which are projected to the point cloud. These methods require the true depth of each pixel to compute the point-pixel mapping, which makes them less applicable in a real-world setting. SnapNet [25], as well as more recent work [171, 55] generate virtual views processed by a 2D network and whose predictions are then projected back to the point cloud. These approaches, while performing well, require a costly mesh reconstruction preprocessing to generate meaningful images. Some approaches [117, 168] fuse RGB and range images, which requires dedicated sensors and can not handle multiple views with occlusions. Existing hybrid 2D/3D methods rely on a fixed number of images per point chosen with heuristics such as the maximization of unseen points [65, 149, 171]. Then, the different views are merged using pooling operations (max [292, 65] or sum-pool [149]) or based on the 2D features' content [135]. To the best of our knowledge, no method has yet been proposed

to leverage the viewing conditions for multi-view aggregation for the semantic segmentation of large scenes.

The problem of selecting and merging the best images for a 3D scene has been extensively studied for surface reconstruction and texturing. Images are typically chosen according to the viewing angle with the surface normal [189, 20], proximity and resolution [9, 33], geometric and visibility priors [278], as well as *crispness* [92], and consistency with respect to occlusions [318]. While most of these criteria do not directly apply to point clouds, they illustrate the importance of camera pose information for selecting relevant images.

Related to our setting is the *Next Best View* selection problem [281], which consists in planning the camera position giving the *most information* about an object of interest [61]. This criterion takes different meanings according to the setting, such as the number of unseen voxels [312], diversity [226], information-theoretic measures of uncertainty [144], or can be directly learned end-to-end [219, 335]. Our setting differs in that the images have already been acquired, and the task is to choose which one contains the most relevant information for each point. We draw inspiration from the end-to-end approaches demonstrating that a neural network can assess the quality of information contained in an image from pose information.

The problem of view selection is also addressed in the literature on shape recognition [292]. Features from different images can be merged based on their similarity [321], *discriminativity* [86], or using patch matching schemes [344, 349] or graph-neural networks [329]. Some methods use 3D features [346] or camera position [157, 112] to select the best views, but no technique yet makes explicit use of the viewing configuration. Furthermore, these methods operate on synthetic views of artificial shapes, which differs from our goal of analyzing large scenes with images in arbitrary poses.

Closer to our problem, Armeni *et al.* [14] aggregate views using handcrafted

heuristics. Bozic *et al.* [28] use a distance-aware attentive view aggregation for 3D reconstruction, but disregard other viewing conditions.

5.3 Method

Let P be a set of 3D points and I a collection of co-registered images, all acquired from the same scene. We characterize points by their position in space, and images by their pixels' RGB values along with intrinsic and extrinsic camera parameters. Our goal is to exploit the correspondence between points and image pixels to perform 3D point cloud semantic segmentation with features learned from both modalities.

Our method starts by computing an occlusion-aware mapping between 3D points and pixels, then uses viewing conditions through an attention scheme to aggregate relevant image features for each 3D point. This approach can be easily integrated into a standard 3D network architecture, allowing us to learn from both point clouds and images simultaneously in an end-to-end fashion.

5.3.1 Point-Image Mapping

We start by efficiently computing a mapping between the images of I and the points of P . We say that a point-image pair $(p, i) \in P \times I$ is *compatible* if p is visible in i , *i.e.* p is in the frustum of i and not occluded. For such a pair, we define the re-projection $\text{pix}(p, i)$ as the pixel of i in which p is *visible*. Note that as points are zero-dimensional objects (zero-volume), $\text{pix}(p, i)$ is a single pixel. We denote by $v(p)$ the *views* of p , *i.e.* the set of images in which p is visible.

Point-Pixel Mapping Construction. We operate in a general *in the wild* multi-view setting in which the optical axes of the cameras and the 3D sensor are

not necessarily aligned. Consequently, computing the point-image mapping requires a *visibility model* to detect occlusions. This can be done by computing a full mesh reconstruction from the point clouds or by using a depth map obtained by a camera-aligned depth sensor or other means. In contrast, we propose an efficient implementation of the straightforward Z-buffering method [291] to compute the mapping directly from images and point clouds. For each image $i \in I$, we replace all 3D points in the frustum of i under a pre-determined distance by a square plane section facing towards i and whose size depends on their distance to the sensor and the resolution of the point cloud. We can compute the projection mask—or *splat*—of each square onto i using the camera parameters of i . We iteratively accumulate all splats in a depth map called Z-buffer by keeping track of the closest point-camera distance for each pixel. Simultaneously, we store corresponding point indices in an index map, along with other relevant point attributes. Once all splats have been accumulated, *visible* points are the ones whose indices appear in the index map. For each visible point p , we set $\text{pix}(p, i)$ as the pixel of i in which p itself is projected. Our GPU-accelerated implementation can process the entire S3DIS dataset [13] subsampled at 5 cm (12 million points and 1413 high-resolution equirectangular images) within 65 seconds. See Figure 5.2 for an illustration of our mapping construction, and Section 5.4.3 for an analysis of alternate visibility models.

Projection Information. To each compatible point-image pair (p, i) , we associate a D -dimensional vector $o_{(p,i)}$ describing the conditions under which the point p is seen in i . In practice, we define this vector as a set of $D = 8$ handcrafted features qualifying the observation conditions of (p, i) .

- **Normalized depth (1).** An image seeing a point at a distance may contain relevant contextual cues but poor textural information. We

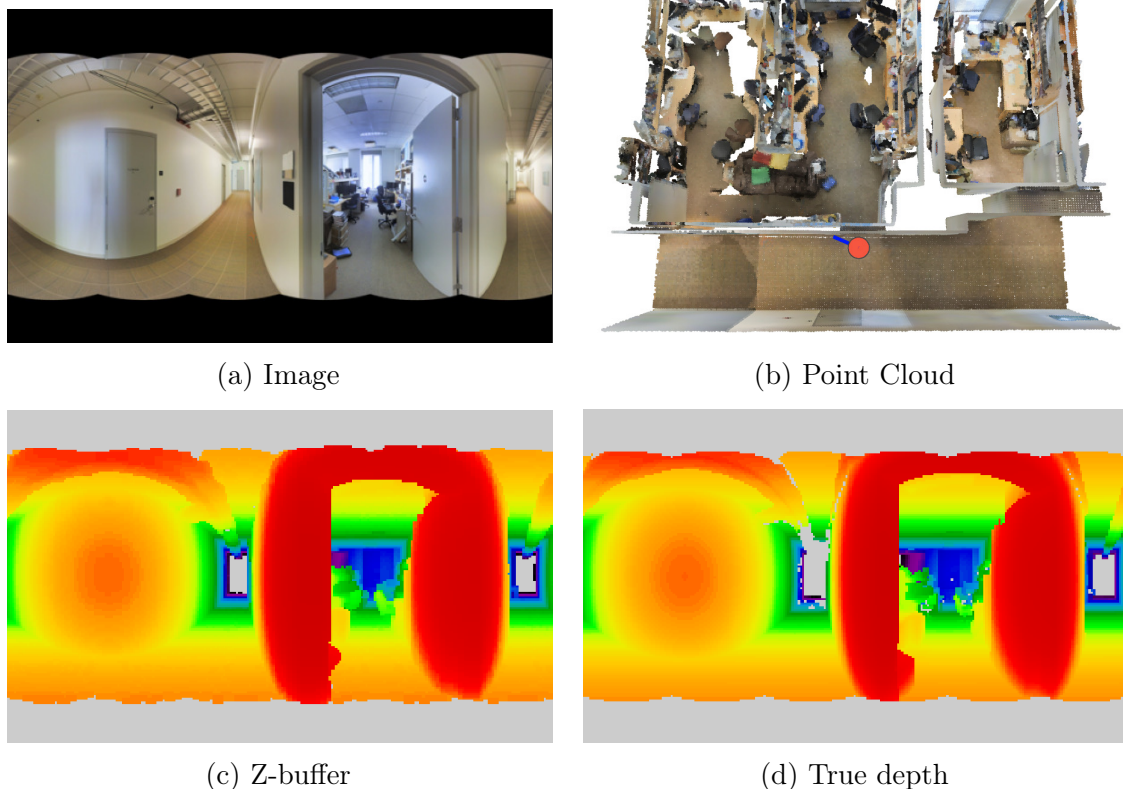


Figure 5.2 – **Mapping Computation.** We estimate pixel depth for all (a) images using the co-registered (b) point cloud. We compute (c) Z-buffers with an efficient GPU-accelerated implementation, resulting in depth maps comparable to the (d) true distance given by camera-aligned depth sensors. We use our estimated depth maps to compute point-image mappings. Better seen on a monitor.

compute the distance $\text{dist}(p, i)$ between point p and image i and divide by the maximum viewing distance $R = 8$ m for indoor scenes and $R = 20$ m for outdoor scenes.

- **Local geometric descriptors (3).** The geometry of a point cloud can impact the quality of its views in images. Indeed, while planar surfaces may be better captured by a camera, a highly irregular surface may present many occlusions or grazing rays. We compute geometric descriptors (linearity, planarity, scattering) based on the eigenvalues of the covariance matrix between a point and its 50 neighbors [70].

- **Viewing angle (1)**. An image seeing a surface from a right angle may better capture its surroundings than if the view angle is slanted with respect to the surface. We compute the absolute value of the cosine between the viewing angle and the normal estimated from the covariance matrix calculated at the previous step.
- **Pixel row (1)**. To account for potential camera distortion near the top and bottom of the image (*e.g.* for equirectangular images), we report the row of pixels and divide by the image height (number of rows). Note that we could derive a similar feature for cameras with radial distortion, such as fisheye cameras.
- **Local density (1)**. Density can impact occlusion and be an indicator of the local precision of the 3D sensor. We compute the area of the smallest disk containing the 50th neighbor and normalize it by the square of the voxel grid resolution.
- **Occlusion rate (1)**. Occlusion may significantly impact the quality of the projected image features. We compute the ratio of the 50 nearest neighbors of p also seen in i .

See Section 5.4.3 for an analysis of the impact of these values.

Efficient Implementation. In the Z-buffering step, we only consider points at a maximum distance $R = 8$ m for indoor scenes and $R = 20$ m for outdoor settings. We replace the points in image i by cubes oriented towards i and with a size given by the following formula involving $\text{dist}(p, i)$ the distance between point p and image i , $k = 1$ a swell factor ruling how much closer cubes are expanded and c the resolution of the voxel grid, or a typical inter-point distance (2-8cm in our experiments):

$$\text{size_of_cubes}(\text{dist}(p, i)) = c(1 + ke^{-\text{dist}(p, i)/R}) . \quad (5.1)$$

This heuristic increases the size of cubes that are close to the image to ensure that they do *hide* the cubes behind them. Note that this heuristic operates on the size of the 3D cubes before camera projection and not on their projected pixel masks, which are computed based on camera intrinsic parameters. See Algorithm 2 for the pseudo-code of the mapping computation.

Storing point-pixel mappings for large-scale scenes with many images can be challenging. To minimize the memory impact of such a procedure, we use the Compressed Sparse Row (CSR) format. This allows us to represent the mappings compactly and treat large scenes at once.

Algorithm 2 Z-buffering-Based Point-Pixel Mapping

Input: I image set, P point cloud

```

for  $i \in I$  do
  zBuffer  $\leftarrow$  maxFloat array of size  $i$ 
  indexMap  $\leftarrow$  NaN array of size  $i$ 
   $P' \leftarrow$  points of  $P$  in frustum of  $i$  and closer than  $R$ 
  for  $p \in P'$  do
     $s \leftarrow$  size_of_cubes(dist( $p, i$ ))
     $mask \leftarrow$  pixel mask covered by the projection
      of a cube of size  $s$  at  $p$  onto the image  $i$ 
    for  $(u, v) \in mask$  do
      if dist( $p, i$ ) < zBuffer[ $u, v$ ] then
        zBuffer[ $u, v$ ]  $\leftarrow$  dist( $p, i$ )
        indexMap[ $u, v$ ]  $\leftarrow p$ 
      end if
    end for
  end for
for  $p \in P'$  do
  if  $p$  appears in indexMap then
    pix( $p, i$ )  $\leftarrow$  pixel at the projection of  $p$  on  $i$ 
  else
    pix( $p, i$ ) not defined,  $p$  not seen in  $i$ 
  end if
end for
end for

```

5.3.2 Learning Multi-View Aggregation

We denote by $\{f_i^{2D}\}_{i \in I}$ a set of 2D feature maps of width C associated to the images I , typically obtained with a convolutional neural network (CNN). Our goal is to transfer these features to the 3D points by exploiting the correspondence between points and images. However, not all viewing images contain equally relevant information for a given 3D point. We propose an attention-based approach to weigh and aggregate features from the viewing images for each point p .

View Features. The mapping $\text{pix}(p, i)$ described in Section 5.3.1 allows us to associate image features to each compatible point-image pair (p, i) :

$$\tilde{f}_{(p,i)}^{2D} = \text{MLP} (f_i^{2D} [\text{pix}(p, i)]) , \quad (5.2)$$

with $\text{MLP} : \mathbb{R}^C \mapsto \mathbb{R}^C$ a Multi-Layer Perceptron (MLP). Learned image features can contain information of different natures: contextual, textural, class-specific, and so on. To reflect this consideration, we split the channels of $\tilde{f}_{(p,i)}^{2D}$ into K contiguous blocks of $\lfloor C/K \rfloor$ channels:

$$\tilde{f}_{(p,i)}^{2D} = \left[\tilde{f}_{(p,i),1}^{2D}, \dots, \tilde{f}_{(p,i),K}^{2D} \right] . \quad (5.3)$$

with $[\cdot]$ the channel-wise concatenation operator. Each block of channels represents a subset of the image information contained in \tilde{f}^{2D} .

View Quality. The conditions under which a point is seen in an image can be more or less conducive to certain types of information, see Figure 5.3. For example, an image viewing a point from a distance may give important contextual cues, while an image taken close and at a straight angle with respect to the local 3D surface may give detailed textural information. In contrast, an image in which a point’s local surface is seen from a slanted angle

or under high distortion may not contain relevant information and may need to be discarded. To model these complex dependencies, we propose to predict for each compatible point-image pair (p, i) a set of K *quality scores* $x_{(p,i)}^k \in \mathbb{R}$ from its viewing conditions $o_{(p,i)}$ defined in Section 5.3.1. The quality $x_{(p,i)}^k$ represents the relevance for point p of the information contained in the feature block k of image i .

For each point p , we consider the set $v(p)$ of images in which it is visible. We propose to learn to predict the view quality $x_{(p,i)}^k$ for each feature block k by considering all images $i \in v(p)$ *simultaneously*. Indeed, the relevance of an image can depend on the context of the other views. For example, while a given image may provide less-than-perfect viewing conditions of a given 3D point, it may be the only available image with global information of the point’s context. We use a deep set architecture [350] to map the set of viewing conditions $\{o_{(p,i)}\}_{i \in v(p)}$ to a vector of size K :

$$z_{(p,i)} = \phi_1(o_{(p,i)}) \quad (5.4)$$

$$x_{(p,i)} = \phi_3 \left(\left[z_{(p,i)}, \phi_2 \left(\max\{z_{(p,i)}\}_{i \in v(p)} \right) \right] \right), \quad (5.5)$$

with $\phi_1 : \mathbb{R}^D \mapsto \mathbb{R}^M$, $\phi_2 : \mathbb{R}^M \mapsto \mathbb{R}^M$, and $\phi_3 : \mathbb{R}^{2M} \mapsto \mathbb{R}^K$ three MLPs, M the size of the set embedding, and \max the channelwise maximum operator for a set of vectors.

View Attention Scores. We can now compute K attention scores $a_{(p,i)}^k$ in $[0, 1]$ corresponding to the relative relevance for point p of the k th feature block of image i . The attentions are obtained by applying a softmax function to the quality scores $x_{(p,i)}^k$ across the images in $v(p)$. To account for the possibly varying number of views per point, we scale the softmax according

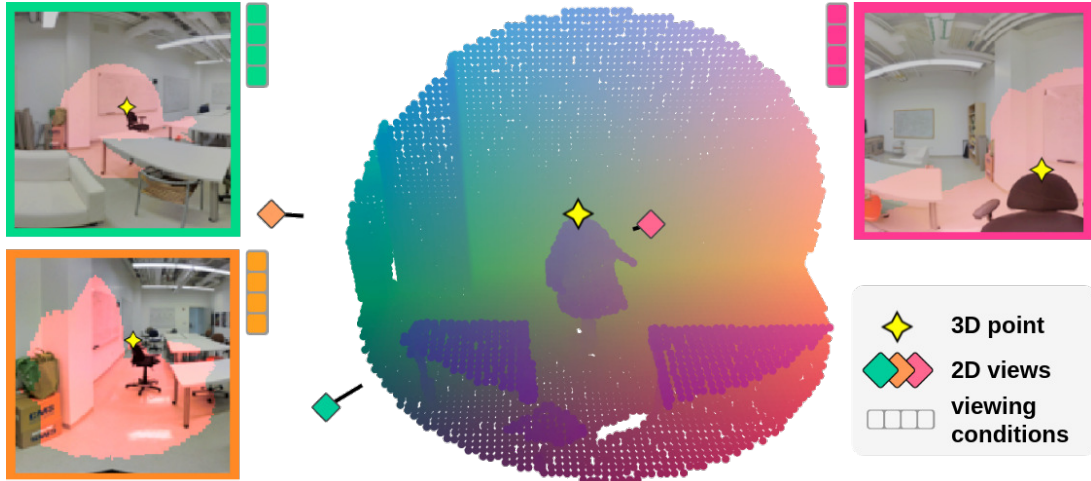


Figure 5.3 – **Multi-View Information.** A 3D point is seen in several images with different insights. Here, the **green** image contains contextual information, while the **pink** image captures the local texture. The **orange** image sees the point at a slanted angle and may contain no additional relevant information.

to the number of images seeing the point p :

$$a_{(p,i)}^k = \text{softmax} \left(\frac{1}{\sqrt{|v(p)|}} \left\{ x_{(p,i)}^k \right\}_{i \in v(p)} \right). \quad (5.6)$$

View Gating. A limitation of using a softmax in this context is that the attention scores $\tilde{a}_{(p,i)}^k$ always sum to 1 over $v(p)$ regardless of the overall quality of the image set. Because of occlusion or limited viewpoints, some 3D points may not be seen by any relevant image for a given feature block k (e.g. no close or far images). In this case, it may be beneficial to discard an information block from all images altogether and purely rely on geometry. This allows the 2D network to learn image features without accounting for potentially spreading corrupted information to points with dubious viewing conditions. To this end, we introduce a gating parameter g_p^k whose role is to block the transfer of the features block k if the overall quality of the image set $v(p)$ is too low:

$$g_p^k = \text{ReLU} \left(\tanh \left(\alpha_k \max_{i \in v(p)} \left(x_{(p,i)}^k \right) + \beta_k \right) \right), \quad (5.7)$$

with $\alpha, \beta \in \mathbb{R}^K$ trainable parameters and ReLU the rectified linear activation [228]. If all quality scores $x_{(p,i)}^k$ are negative for a given point p and block k , the gating parameter g_p^k will be exactly zero and block possibly detrimental information due to sub-par viewing conditions.

Attentive Image Feature Pooling. For each point p seen in one or more images, we merge the feature maps $\tilde{f}_{(p,i)}^{2D}$ from each view (p, i) . For each block k , we compute the sum of the view features $\tilde{f}_{(p,i),k}^{2D}$ weighted by their respective attention scores $a_{(p,i)}^k$ and multiplied by the gating parameter g_p^k . The combined image feature $\mathcal{P}(f^{2D}, p)$ associated to point p is then defined as the channelwise concatenation of the resulting tensors for all blocks:

$$\mathcal{P}(f^{2D}, p) = \left[g_p^k \sum_{i \in v(p)} a_{(p,i)}^k \tilde{f}_{(p,i),k}^{2D} \right]_{k=1}^K. \quad (5.8)$$

5.3.3 Bimodal Point-Image Network.

We can use the multi-view feature aggregation method described above to perform semantic segmentation of a point cloud and co-registered images by combining a network operating on 3D point clouds and 2D CNN.

Fusion Strategies. We use a 2D fully convolutional network to compute pixel-wise image feature maps f^{2D} . We also consider a 3D deep network following the classic *U-Net* architecture [270] and composed of three parts: (i) an encoder \mathcal{E}^{3D} mapping the point cloud into a set of 3D feature maps at different resolution (*innermost* map and skip connections); (ii) a decoder \mathcal{D}^{3D} converting these maps into a 3D feature map at the highest resolution (iii) a classifier \mathcal{C}^{3D} associating to each point a vector for class scores of size N , the number of target classes.

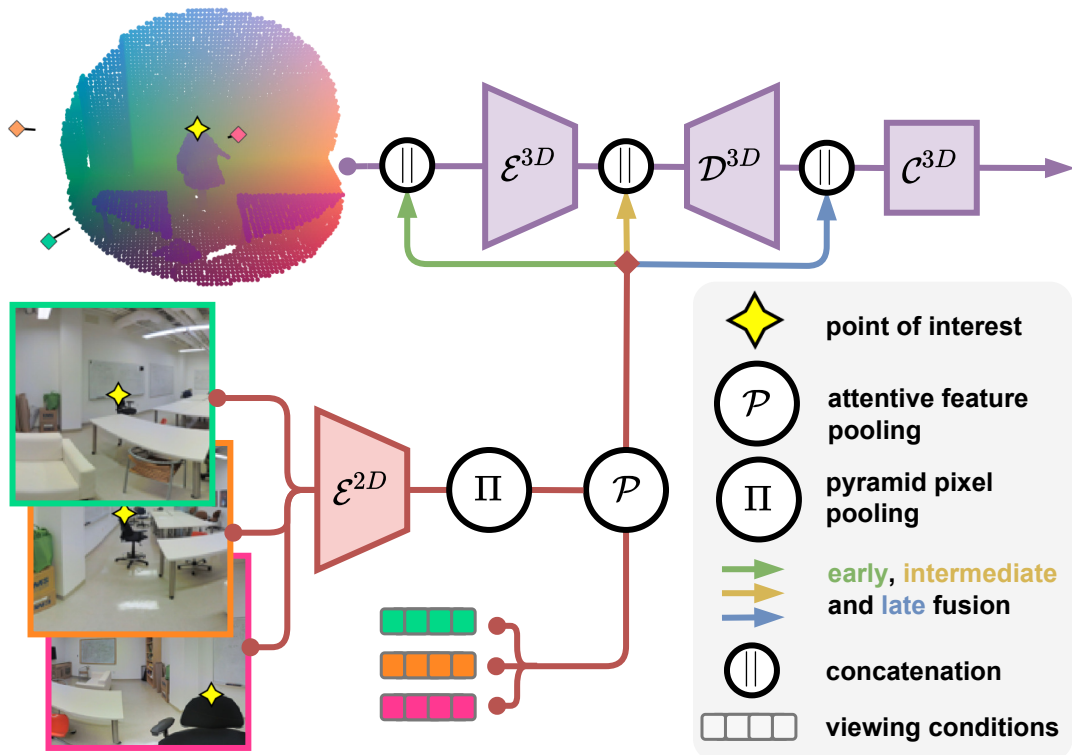


Figure 5.4 – **Bimodal 2D/3D Architecture.** Using our multi-view aggregation module, we combine a 2D convolutional encoder \mathcal{E}^{2D} and a 3D network composed of an encoder \mathcal{E}^{3D} , a decoder \mathcal{D}^{3D} , and a classifier \mathcal{C}^{3D} . We associate relevant 2D features to each 3D point according to their viewing conditions in each compatible image. We propose three different 2D/3D fusion strategies: early (our choice in the experiments), intermediate, and late fusion.

As shown in Figure 5.4, we investigate three classic fusion schemes [117, 149, 168], connecting the image features at different points of the 3D network: (i) directly with the raw 3D features before \mathcal{E}^{3D} (*early fusion*), (ii) in the skip connections (*intermediate fusion*) (iii) between the decoder \mathcal{D}^{3D} and the classifier \mathcal{C}^{3D} (*late fusion*). See the Section D-2 for the details and equations for these fusion schemes.

Dynamic-Size Image-Batching. The number of images $v(p)$ in which a point p is visible can vary significantly. Furthermore, when dealing with large-scale scenes, only a subset P_{sample} of the 3D scene is typically processed at once (*e.g.* spherical sampling). For this reason, the part of an image i for which points of

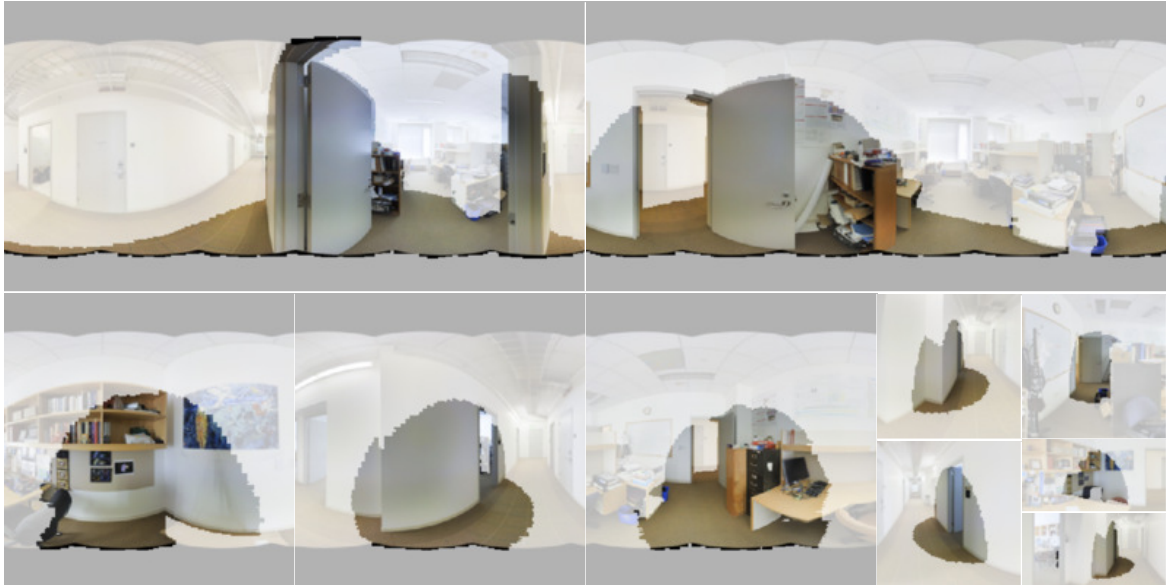


Figure 5.5 – **Dynamic Batching.** We can improve the quantity of information contained in each training batch by cropping images around the sampled point clouds. We represent a set of 10 images with different crop size fitting in a budget of pixels corresponding to 4 full-size images.

P_{sample} are visible can sometimes be only a small fraction of the entire image. This will typically occur with equirectangular images or when P_{sample} is far away from i . We use the adaptive batching scheme depicted in Figure 5.5 to stabilize memory usage across batches and avoid needless computations on excessively large images. The first step is to crop each image using the smallest window across a fixed set of sizes (*e.g.* 64×64 , 128×64 , etc.) such that the crop contains the bounding box of all seen points of P with a given margin. Observing that the memory consumption of a fully convolutional encoder is linear w.r.t. the number of input pixels, we allocate to each point cloud in the batch a *budget* of pixels. Images are then chosen randomly by iteratively selecting images with a probability proportional to their number of pixels and to the number of newly seen points in the cloud, until the pixel budget is spent. Finally, the images are organized into different batches according to their sizes, allowing for their simultaneous processing. Note that at inference time, we can take batches as large as the GPU memory allows.

More details on our dynamic-size batching implementation are provided in Section D-3.

5.3.4 Implementation Details

We use sparse encoding for mappings in order to only store compatible point-image pairs. This proves necessary for the large scale, in-the-wild setting with varying number of images seeing each point. The exact network and training configurations are given in Section D-4. Our code is available at <https://github.com/drprojects/DeepViewAgg>.

5.4 Experiments

We propose several experiments on public large-scale semantic segmentation benchmarks to demonstrate the benefits of our deep multi-view aggregation module (DeepViewAgg). Our approach yields significantly better results than our 3D backbone directly operating on colorized point clouds. We set a new state-of-the-art for the highly contested S3DIS benchmark using only standard 2D and 3D architectures combined with our proposed module.

5.4.1 Datasets

S3DIS [13]. This indoor dataset of office buildings contains over 278 million semantically annotated 3D points across 6 building areas—or *folds*. A companion dataset can be downloaded at <https://github.com/alexsax/2D-3D-Semantics>, and contains 1413 equirectangular images. To represent our large-scale, in-the-wild setting, we merge each fold into a large point cloud and discard all room-related information. We apply minor registration adjustments detailed in Section D-5.

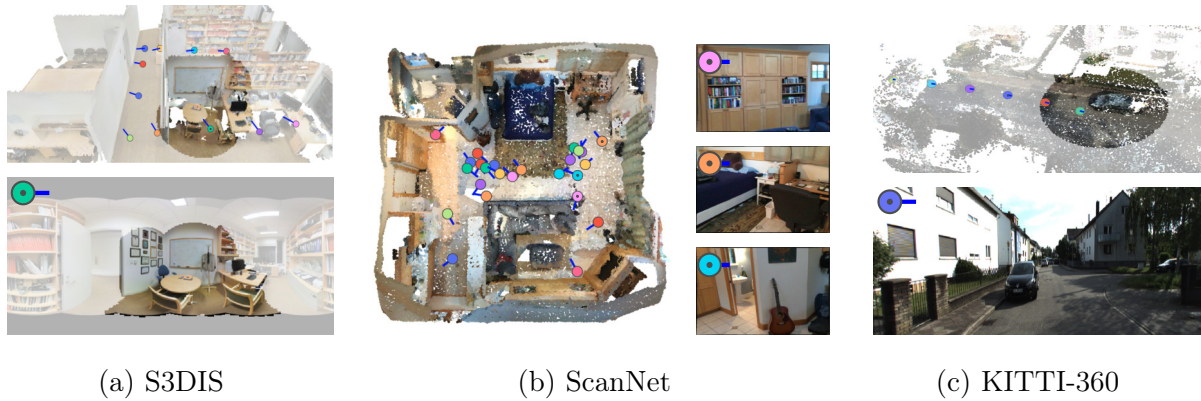


Figure 5.6 – **Datasets.** Illustration of the sampling procedure for all considered datasets with point clouds alongside some of the available images. The 3D components of batches are constituted of spheres for (a) S3DIS, rooms for (b) ScanNet, and cylinders for (c) KITTI-360.

ScanNet [66]. This indoor dataset contains over 1501 scenes obtained from 2.5 million RGB-D images with pose information. To account for the high redundancy between images, we select one in every 50 image. This dataset deviates slightly from our intended setting as 2D and 3D are derived from the same sensors.

KITTI-360 [197]. This large outdoor dataset contains over 100k laser scans and 320k images captured with a multi-sensor mobile platform. We use one image every five from the left perspective camera. We report the class-wise performance on the official withheld test set.

General Setting. All datasets provide colored point clouds obtained with dataset-specific preprocessings. To handle the large size of scans, we define batches using a sampling strategy for S3DIS (2 m-radius spheres) and KITTI-360 (6 m-radius vertical cylinders), while we process ScanNet room-by-room, see Figure 5.6. We down-sample the point clouds for processing (S3DIS: 2cm, ScanNet: 3cm, KITTI-360: 5cm) and interpolate our prediction to full resolution for evaluation. To mitigate the memory impact of the 2D encoder, we also down-sample S3DIS images to 1024×512 but keep the full resolution

for ScanNet (320×240) and KITTI-360 (1408×376).

5.4.2 Quantitative Evaluation

Table 5.1 – **Quantitative Evaluation.** Mean Intersection-over-Union of different state-of-the-art methods on S3DIS’s Fold 5 and 6-fold, ScanNet Val, and KITTI-360 Test. All methods except the last line are trained on colored point clouds. **State-of-the-art, second highest.** ¹ with 3D supervision only.

Model	S3DIS		ScanNet	KITTI
	Fold 5	6-Fold	Val	360 Test
<i>Methods operating on colored point clouds</i>				
PointNet++ [253]	-	56.7 [46]	67.6 [47]	35.7 [197]
SPG+SSP [180, 177]	61.7	68.4	-	-
MinkowskiNet [58]	65.4	65.9[46]	<u>72.4</u> [233]	-
KPConv [301]	67.1	70.6	69.3 [233]	-
RandLANet [133]	-	70.0	-	-
PointTrans.[79]	70.4	<u>73.5</u>	-	-
Our 3D Backbone	64.7	69.5	69.0	<u>53.9</u>
<i>Methods operating on point clouds and images</i>				
MVPNet [149]	62.4	-	68.3	-
VMVF [171]	65.4	-	76.4	-
BPNNet [135]	-	-	69.7 ¹	-
3D Backbone+	<u>67.2</u>	74.7	71.0	58.3
DeepViewAgg (ours)				

In Table 5.1, we compare the performance of our approach and other learning methods on S3DIS, ScanNet Validation, and KITTI-360 Test using the class-wise mean Intersection-over-Union (mIoU) as metric. Our method (DeepViewAgg) uses images in the 2D encoder and *raw uncolored point clouds* in the 3D encoder. All other approaches, including our backbone (3D Backbone), use the colored point clouds provided by the datasets.

DeepViewAgg sets a new state-of-the-art for S3DIS for all 6 folds and the second-highest performance for the 5th fold. In particular, we outperform the VMVF network [171], showing that our multi-view aggregation model can

overtake methods relying on costly virtual view generation using only available images. Furthermore, VMVF uses true depth maps, colorized point clouds, normals, and room-wise normalized information. In contrast, our method only uses raw XYZ data in the 3D encoder and estimates the mappings. Our approach also overtakes the recent PointTransformer [79] (PointTrans.) by 1.2 mIoU points, even though this method outperforms our 3D backbone by 4 points on colorized points. Our model also improves the performance of our 3D backbone on the KITTI-360 test set by 4.4 points, illustrating the importance of images for both indoor and outdoor datasets alike.

While giving reasonable results, our method does not perform as well on the validation set of ScanNet comparatively. We outperform the 2D/3D fusion method of BPNet [135] when restricted to 3D annotations, illustrating the importance of view selection. We argue that the limited variety in the camera points of view of ScanNet RGB-D scans, as well as their small field-of-view and blurriness reduce the quality of the information provided by images. This is reinforced by the impressive performance of VMVF, which synthesizes its own images with controlled points of view and resolution. See Figure 5.7 for qualitative illustrations.

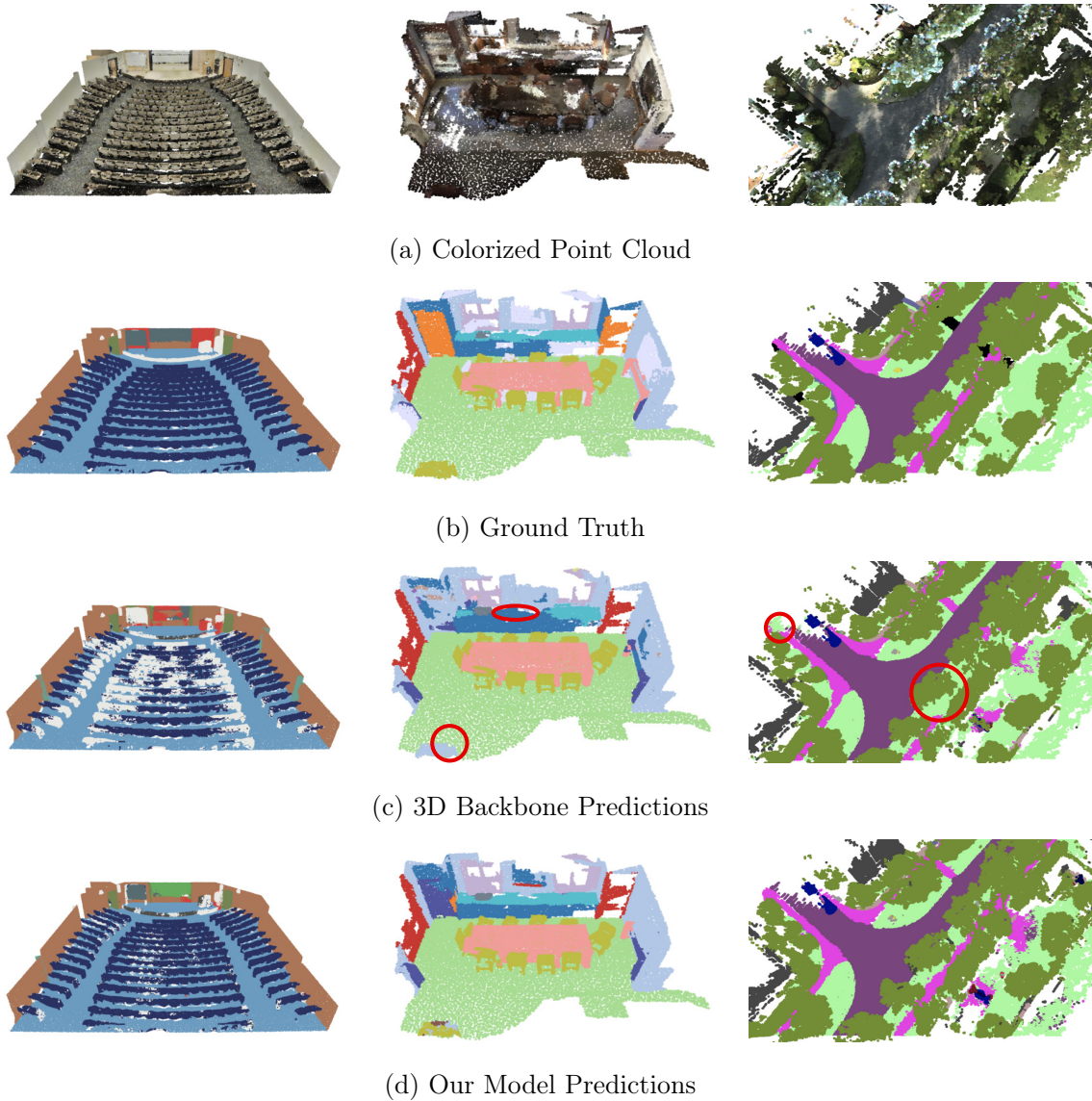


Figure 5.7 – **Qualitative Illustration.** Scenes from our considered datasets (top: S3DIS, middle: ScanNet, bottom: KITTI-360) with (a) colorized point clouds, (b) ground truth point annotations, (c) prediction of the backbone network operating on the colorized point cloud, and (d) our method operating on raw uncolored point clouds and images. Our approach is able to use images to resolve cases in which the geometry is ambiguous or unusual, such as a large amphitheater with tiered rows of seats (top row). Color legend given in Section A-1.

5.4.3 Analysis

We conduct further analyses on Fold 5 and Fold 2 of S3DIS (subsampling at 5cm for processing) and the validation set of KITTI-360 in Table 5.2. We added Fold 2 along the commonly used Fold 5, as it benefited most from our method, and hence is more conducive to evaluating the impact of our design choices.

Modality Combinations. As observed in Table 5.2, combining a 3D deep network operating on raw 3D features and a 2D network with our method (Best Configuration) improves the performance by over 6 to 15 points compared to the same 3D backbone operating on colorized point clouds alone (XYZRGB). To illustrate that point colorization is not a trivial task, we train our 3D backbone with point clouds colorized by averaging for each point the color of all pixels in which it is visible (XYZ Average-RGB). Compared to the “official” colored point clouds, we observe a drop of 1 point for Fold 5 and 1.3 point for KITTI-360, but a gain of almost 5 points for Fold 2. This shows how different point cloud colorization schemes can yield vastly different results. Not using any radiometric information and purely relying on 3D points without color (XYZ) decreases the score of XYZRGB by a further 3 to 4 points on S3DIS. For KITTI-360, XYZ outperforms XYZ Average-RGB, suggesting that poor colorization can even be detrimental.

We also evaluate a scheme in which the 3D network is entirely removed, and 3D points are classified solely based on features coming from a 2D encoder-decoder and our view aggregation module, without any 3D convolution (Pure RGB). This method outperforms even (XYZRGB) for S3DIS, illustrating the relevance of images for point cloud segmentation. On KITTI-360, as many 3D points are not seen by the cameras used, this approach perform worse.

Training our best 2D+3D model with images downsampled by a factor of 2

(Lower Image Resolution) brings a large performance drop. In contrast, using 2 cm of 3D resolution instead of 5 cm (Higher 3D Resolution) has little impact for S3DIS. We conclude that when the images already contain fine-grained information, the impact of the resolution of the 3D voxel grid decreases.

Table 5.2 – **Ablation Study.** Mean IoU comparison of different modalities and design choices on Fold 2 and Fold 5 of S3DIS down-sampled at 5cm for processing and KITTI-360 Val.

Model	S3DIS		KITTI
	Fold 2	Fold 5	360 Val
Best Configuration	63.2	67.5	57.8
<i>Modality Combinations</i>			
XYZRGB	-15.9	-6.0	-3.6
XYZ Average-RGB	-10.8	-7.0	-4.9
XYZ	-19.5	-9.5	-4.1
Pure RGB	-5.3	-5.4	-14.5
Lower Image Resolution	-5.9	-0.8	-0.7
Higher 3D Resolution	-1.0	-0.3	-
<i>Design Choices</i>			
Late Fusion	-9.1	-1.0	-1.3
Only One Group	-4.8	-0.8	-0.4
No Gating	-3.0	-0.4	-1.1
No Dynamic Batch	-6.9	-1.9	-4.5
No Pretraining	-7.2	-6.7	-3.7
MaxPool	-0.8	-1.5	-2.9
Smaller 3D backbone	-0.5	-0.7	+0.7

Design Choices. Using late fusion (Late Fusion) instead of early fusion gives comparable results on Fold 5 and KITTI-360, but significantly worse for Fold 2 for which the gain of using images is more pronounced. Using only one feature group (Only One Group, $K = 1$ in Equation 5.3) results in a drop of 4.8 points for Fold 2, highlighting that our method can learn to treat different types of radiometric information specifically. Removing the gating mechanism (No Gating, see Equation 5.7) decreases the IoU by 3 points for Fold 2, and

1.1 on KITTI-360. Not using dynamic batches forces us to limit ourselves to 4 full-size images per 3D sphere/cylinder, which results in performance drops of 2 to 7 points. Pretraining the 2D network on related open-access datasets (No 2D Pretraining) accounts for up to 7 mIoU points. Not only do images contain rich radiometric information, but they also allow us to leverage the ubiquitous availability of annotated 2D datasets.

Using featurewise max-pooling to merge the views results in a drop of 1 to 1.5 points for S3DIS and 3 points for KITTI-360. This illustrates that as long as we employ proper mapping, batching, and pretraining strategies, even simple pooling operations can perform very well. However, the addition of our model appears necessary to improve the precision even further and reach state-of-the-art results.

Switching our 3D backbone to a lighter version of MinkowskiNet with decreased widths, we observe no significant impact on the prediction quality. This suggests that we could use our approach successfully with smaller models.

Influence of the Viewing Conditions. We propose to highlight the role viewing conditions descriptor. In Table 5.3, we estimate the usage by our model of each feature as the drop in mIoU on S3DIS Fold 5 & KITTI-360 when they are replaced by their dataset average (*e.g.* all points appear at the same distance). We also measure the feature sensitivity by averaging the squared partial derivative [95, p. 3.3.1] of the view compatibility score x defined in (4) w.r.t. each view descriptor. We observe that our model makes use of all observation features, and that the compatibility scores are most sensitive to small differences in scattering for S3DIS Fold 5, and depth for KITTI-360 Val.

To visualize the influence of viewing conditions, we represent in Figure 5.8 quality score heatmaps when varying pairs of features for a given view point

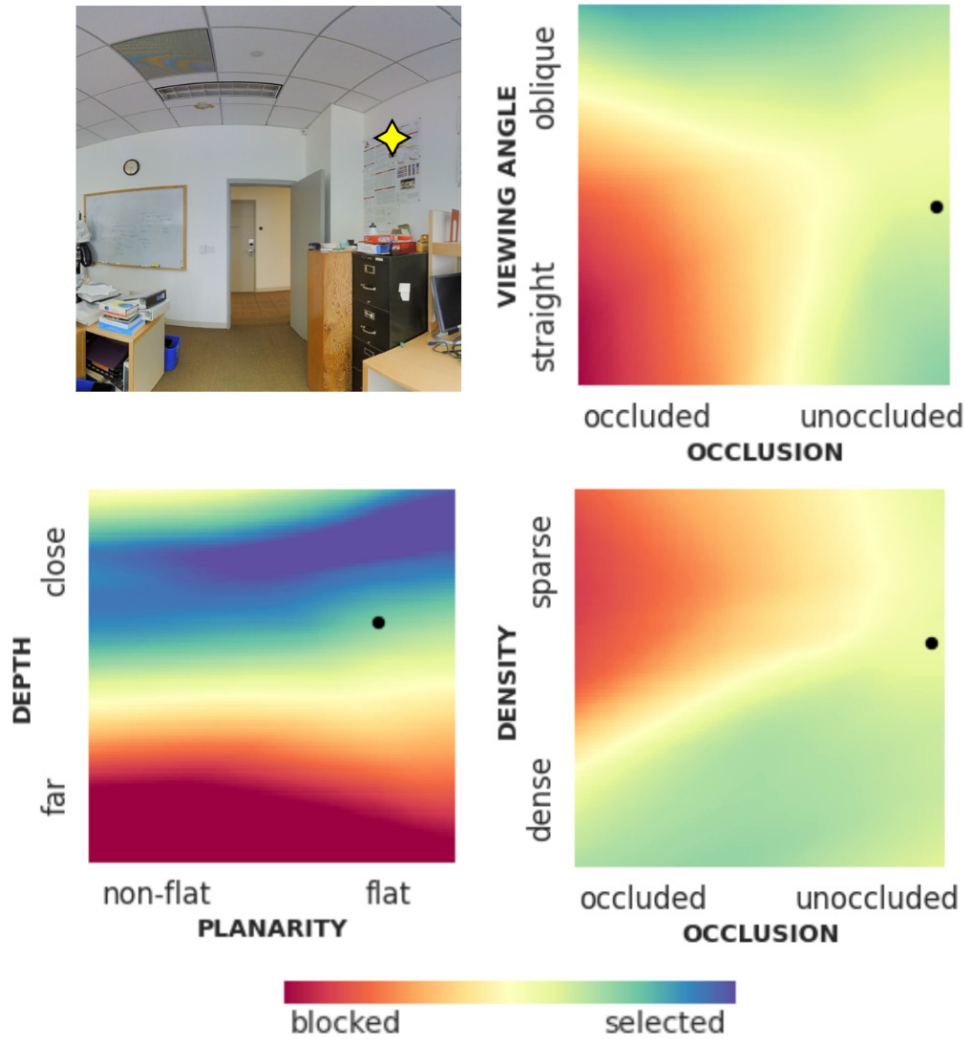


Figure 5.8 – **Influence of Viewing Descriptors.** Given a point-image pair (top left), we compute the quality scores when varying two of its viewing conditions from their initial values \bullet . For simplicity, we omit the influence of other images. We observe feature blocks specializing in retrieving information from views at a given depth range and containing planar objects (bottom left) or blocking straight yet occluded (top right) or sparse and occluded (bottom right) views.

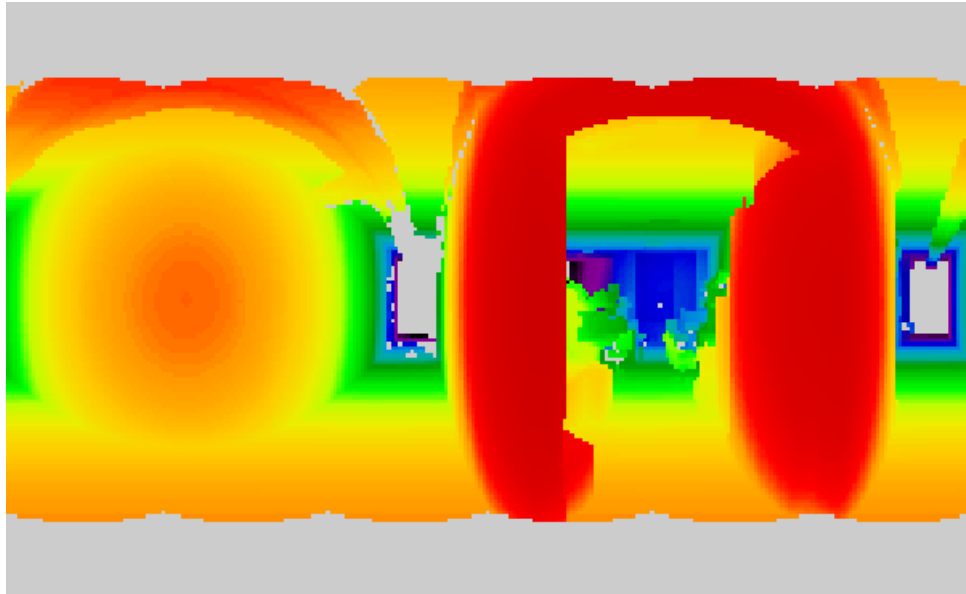
from S3DIS.

Influence of the Visibility Model. We propose further ablations whose results in Table 5.4. To assess the quality of our visibility model, we propose to compute the point-pixel mappings using the depth maps provided with S3DIS instead of Z-buffering. When running our method with such mappings (Mapping from Depth), we observe lower performances. This can be explained

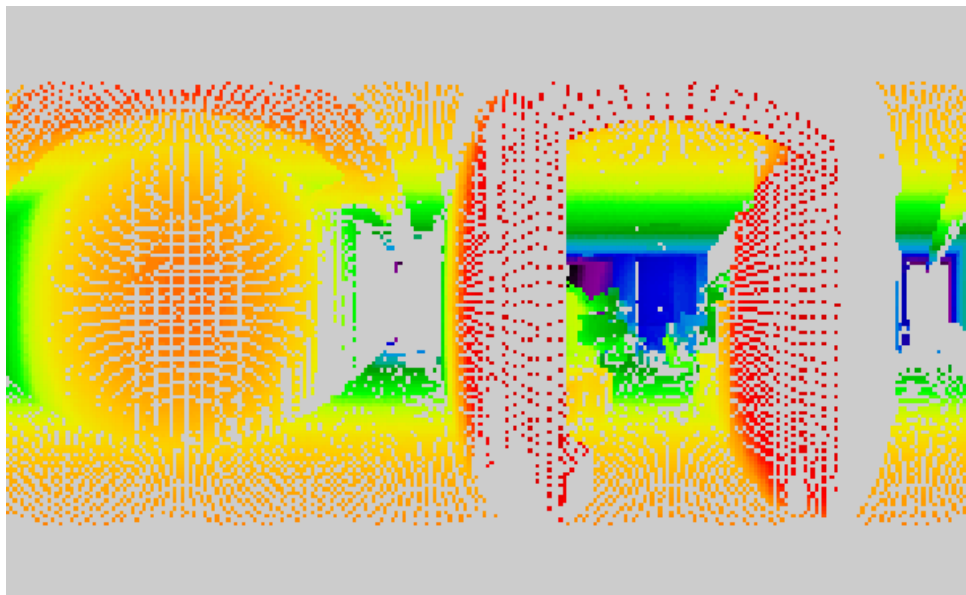
by the fact that depth-based mapping computation is sensitive to minor discrepancies between the depth map and the real point positions. Such a phenomenon can be observed on S3DIS depth images where the surfaces are viewed from a slanted angle, resulting in fewer point-image mappings being recovered. See Figure 5.9 for an illustration of this phenomenon. In conclusion, not only can our approach bypass the need for specialized sensors or costly mesh reconstruction altogether, but our direct point-pixel mapping may yield better results than the provided mappings obtained with more involved methods.

To compare our fusion schemes, we evaluate a model with the intermediate fusion scheme described in D-3 (Intermediate). We observe that, for our module, intermediate fusion does not perform as well as early and late fusion. This could indicate that fusing modalities at their highest respective resolutions yields better results and that matching the encoder levels of 2D and 3D networks may not be straightforward. To ensure that our proposed module captures all radiometric information contained in colorized point clouds, we trained our chosen architecture to run on colorized point clouds and images (XYZRGB + DeepViewAgg). The resulting performance confirms that colorizing 3D points does not bring additional information not already captured by images.

Influence of the Maximum Depth. The maximum point-image depth is chosen as the distance beyond which adjacent 3D points appear in the same image pixel: 8 m for S3DIS sampled at 5 cm with images of width 1024 and 20 m for KITTI-360. As illustrated in Table 5.5, reducing this parameter too much leads to a drop in performance both on S3DIS Fold 5 and KITTI-360, while slight modifications do not significantly affect the results. Since the number of point-image mappings grows quadratically with this parameter, one may



(a) True depth



(b) Depth-based visibility

Figure 5.9 – **Depth-Based Mapping Computation.** Based on an input depth map (a), we compute the point-image mappings (b) by searching points within a small margin of the target depth. We note that slight depth discrepancies near slanted surfaces prevents mapping from being recovered. Better seen on a monitor.

consider smaller values to decrease the memory usage or computing time.

Influence of the Number of Images. In contrast to existing methods (*e.g.* MVPNet [149], VMVF [171], BPNet [135]), our set-based, sparse implementa-

tion of point-image mappings allows us to have a varying number of views per 3D point. In Table 5.6 we investigate the performance drop w.r.t. our best model when limiting the number of images per point cloud to a fixed number of images and not using dynamic batching. Under these conditions, our performance decreases by 6.8 pts on S3DIS Fold 5 and 3.5 pts on KITTI-360 when using only 3 images, *i.e.* the configuration of BPNet. For comparison, 3D points of S3DIS are seen in 5.0 images on average (STD 3.3), and 2.5 for KITTI-360 (STD 2.1).

5.4.4 Limitations

While our method does not require sensors with aligned optical axes, true depth maps, or a meshing step, we still need camera poses. In some “in the wild” settings, they may not be available or require a pose estimation and registration step which may be costly and error-prone. Our mapping computation also relies on the assumption that the 2D and 3D modalities are acquired simultaneously.

Our multi-view aggregation method operates purely on viewing conditions and does not take the geometric and radiometric features into account in the computation of attention scores. We implemented a self-attention-based approach using such features, which resulted in a significant increase in memory usage without tangible benefits: the viewing conditions appear to be the most critical factor when selecting and aggregating images features.

5.5 Conclusion

We proposed a deep learning-based multi-view aggregation model for the semantic segmentation of large 3D scenes. Our approach uses the viewing condition of 3D points in images to select and merge the most relevant 2D features with 3D information. Combined with standard image and point

cloud encoders, our method improves the state-of-the-art for two different datasets. Our full pipeline can run on a point cloud and a set of co-registered images at arbitrary positions without requiring colorization, meshing, or true depth maps. These promising results illustrate the relevancy of using dedicated architectures for extracting information from images even for 3D scene analysis.

Acknowledgements This work was funded by ENGIE Lab CRIGEN and carried on in the LASTIG research unit of Université Paris-Est. The authors wish to thank AI4GEO for sharing their computing resources. AI4GEO is a project funded by the French future investment program led by the Secretariat General for Investment and operated by public investment bank Bpifrance. We thank Philippe Calvez, Dmitriy Slutskiy, Marcos Gomes-Borges, Gisela Lechuga, Romain Loiseau, Vivien Sainte Fare Garnot and Ewelina Rupnik for inspiring discussions and valuable feedback.

Table 5.3 – **Usage and Sensitivity of Viewing Conditions.** Feature usage is reported as a drop in mIoU, and the sensitivity is given as the proportion of squared partial derivative of the compatibility across all features.

view feature	usage (mIoU drop)		sensitivity (in %)	
	S3DIS	KITTI-360	S3DIS	KITTI-360
depth	1.1	1.7	12.6	46.0
linearity	1.0	0.8	11.9	0.7
planarity	1.0	1.4	15.8	1.9
scattering	0.7	1.0	52.7	0.7
viewing angle	1.3	1.2	2.8	7.4
pixel row	1.1	0.8	1.6	33.2
local density	1.2	1.3	0.6	1.8
occlusion	0.7	0.9	2.0	8.2

Table 5.4 – **Supplementary Ablation Study.** Mean IoU comparison of different design choices on Fold 2 and Fold 5 of S3DIS down-sampled at 5cm for processing.

Model	Fold 2	Fold 5
Best Configuration	63.1	67.5
<i>Design Choices</i>		
Mapping from Depth	-5.4	-1.9
Intermediate	-7.6	-3.2
XYZRGB + DeepViewAgg	-0.5	-0.9

Table 5.5 – **Effect of Maximum Depth.** We report the drop in mIoU when removing mappings beyond a threshold distance.

S3DIS FOLD 5							
Max depth	8	7	6	5	4	3	2
mIoU drop	0.0	0.0	0.0	0.4	0.6	1.9	8.6

KITTI-360 Val			
Max depth	20	15	10
mIoU drop	0.0	0.5	2.6

Table 5.6 – Impact of the Number of Images. We report the drop in mIoU when limiting the number of images per point cloud.

# images	8	7	6	5	4	3
S3DIS Fold 5	0.5	1.2	1.7	2.1	3.5	6.8
KITTI-360	0.5	0.1	0.5	1.3	1.9	3.5

Chapter 6

Conclusion

This concluding chapter recapitulates our contributions in Section 6.1, before sketching out in Section 6.2 future research directions that we identify as promising.

6.1 Contributions

This thesis advances the field of 3D computer vision with novel methods for the efficient and scalable analysis of 3D point clouds. Our three contributions can be summarized as follows.

Efficient and Scalable 3D Semantic Segmentation. In Chapter 3, we propose Superpoint Transformer, a novel approach for the efficient semantic segmentation of large 3D point clouds. Our method brings together the best of two worlds: the efficiency of superpoint-based methods and the expressivity of transformer-based methods. We achieve *state-of-the-art* performance on three 3D semantic segmentation benchmarks while being up to $200\times$ more parameter-efficient and $70\times$ faster to train than competing approaches. This work illustrates how designing an adequate data structure with strong inductive priors can render multimillion-parameter models futile.

Efficient and Scalable 3D Panoptic Segmentation. In Chapter 4, we formulate 3D panoptic segmentation as a scalable graph partitioning problem, which our small SuperCluster model is trained to address. Contrary to existing 3D instance and panoptic segmentation, our supervision relies on local objectives only, which allows us to circumvent several limitations of competing methods and scale to very large scenes. We reach *state-of-the-art* performance on four 3D panoptic segmentation benchmarks and show that our method can process scenes of unprecedented size at once on a single consumer-grade GPU.

Learning From Point Clouds and Images in the Wild. In Chapter 5, we introduce DeepViewAgg, an end-to-end multi-view aggregation method for 3D semantic segmentation from images and point clouds. Contrary to existing 2D-3D methods, our approach does not require point cloud colorization, meshing, or depth sensors: only point clouds, images, and their poses. Despite these minimalistic requirements, we achieve *state-of-the-art* performance on two 3D semantic segmentation benchmarks. In a way, this work can be seen as learning to colorize point clouds with features from arbitrarily posed images.

6.2 Perspectives

The works presented in this thesis open the door to several exciting research directions.

6.2.1 Superpoint-Based Learning

We believe that superpoint-based methods introduced in Chapter 3 and Chapter 4 pave the way for promising efficient 3D point cloud processing approaches.

Model Expressivity. Currently, SPT extracts point features using a small PointNet-like model [252] for each superpoint taken in isolation. This approach may lack expressivity and ignores the local relationship between points at the border between superpoints. Inspired by other transformer-based architectures for 2D [206] and 3D [175] analysis that rely on convolution-based modules to extract features from the raw input signal, we hypothesize that our model may benefit from a more expressive point encoder inspired by KPConv [301] or MinkowskiNet [58].

Point Cloud Tokenization. Contrary to other 2D and 3D transformer-based models [76, 206, 175] which reason on arbitrary grid-based representations of the input signal, we show in Chapter 3 that training a transformer to reason on superpoint partitions, allows for tremendous compute and memory efficiency gains. We believe partitioning point clouds into geometrically homogeneous primitives could become a standard preprocessing step similar to tokenization [222] for natural language processing. Such point cloud tokenizer could then be used by 3D deep learning methods as a blackbox preprocessing converting million-point clouds into much smaller collections of basic shapes. Unlike our current algorithm, which relies on handcrafted features to characterize the local geometry, a point cloud tokenizer should ideally be an unsupervised procedure based on dataset statistics, with minimal parameterization.

Learnable Partition. Following SGP [180], our framework in Chapter 3 relies on the cut-pursuit algorithm [179, 261] for computing a fixed partition, at preprocessing time, and based on handcrafted point features. Obviously enough, learning to partition end-to-end would be an attractive alternative. However, our cut-pursuit step is currently a non-differentiable operation. To this end, Landrieu *et al.* [177] propose to pretrain with a dedicated network to learn point features presenting a high contrast at object boundaries, to improve the partition with cut-pursuit. Unfortunately, this approach involves a two-step training and requires instance-level annotations. An end-to-end trainable solution leveraging semantic segmentation labels or no labels at all would be a more convenient solution.

Several directions could be explored. First, the partition could be updated every few training epochs, based on point features learned for the task at hand. Yet, with this approach, the gradient would not be used to adjust

the partition itself. An alternative would be to treat the partition step as a blackbox combinatorial optimization step [250]. Finally and more ambitiously, the partition step could be made entirely differentiable. Although this may require substantial theoretical work for cut-pursuit, differentiable alternatives exist for superpixel [148] or superpoint [140] partitions. However, these approaches are essentially variants of k-means that operate on a fixed number of sampled centroids, which limits their ability to scale to point clouds of arbitrary size or to adapt to the geometric complexity.

Super-X Reasoning. While the methods presented in Chapter 3 and Chapter 4 operate on point clouds, adapting our framework to other modalities, such as images, depth maps, or videos would be a natural next step. Essentially, operating on a new modality would be easy and would only require defining an adjacency graph and local features for the partition step, as well as a trainable modality-specific feature extractor, which is likely to be found in the relevant literature.

6.2.2 2D-3D Learning

Our work in Chapter 5 may be further extended for multimodal learning on point clouds and images.

Multi-View Multi-Sensor Aggregation. Some acquisition systems, such as autonomous vehicles, can be equipped with several image sensors with different optical properties. For example, the KITTI-360 [197] dataset comprises point clouds and images from perspective and fisheye cameras located at different positions on the vehicle. By extending the observation condition features with information about the sensor of each image, we may be able to train DeepViewAgg to selectively attend to different sensors depending on the viewing conditions. For example, our method may learn that the distortion

in fisheye images makes pixel features less reliable as we move away from the center. Meanwhile, the model may also learn that some cameras may be more adequate than others to detect pedestrians, based on their orientation on the acquisition platform.

Multimodal Aggregation. DeepViewAgg proposes to exploit the synergy between point clouds and localized images. However, other modalities may carry complementary information about the 3D scene. Radar sensors, for example, are more robust to visibility and weather conditions than RGB cameras, making them ideal for autonomous driving, as exemplified by the nuScenes [37] dataset. Other potentially informative modalities include HD maps, street camera images, aerial images, and satellite images. Provided a feature extractor and a mapping to 3D points for each modality, we believe the attentive, multi-view formulation of our framework could allow for more than two modalities.

Robust 2D-3D Mapping. Similar to other works on 2D-3D learning, the mapping construction in DeepViewAgg makes two important assumptions: static scenes and known camera parameters.

If an object has moved between the acquisition times of the 3D and 2D sensors, our purely geometric mapping construction will be incorrect. Yet, such dynamic scenes are ubiquitous in robotic and autonomous driving scenarios, where surrounding objects such as vehicles or pedestrians may be dynamic. Taking temporality into account for multimodal mapping and multi-view aggregation would make for an interesting extension of DeepViewAgg. This problem is related to moving object segmentation [245, 161, 53, 166], which is an active research field.

Our mapping computation will also suffer from incorrect camera poses or intrinsics. This typically affects small pixel structures, which correspond to

either small or far-away 3D objects. Existing works on 2D-3D registration [191, 73] could help alleviate this issue, making our method more robust to camera parameter errors.

Super-X Multimodal Learning. Combining the superpoint and multimodal paradigms developed in this thesis offers multiple possibilities. Using superpoints may increase the robustness of DeepViewAgg. Indeed, aggregating pixel features at the superpoint level rather than for individual points may lead to more robust features. In addition, more robust observation condition features may also be computed at the superpoint level.

Aligning superpixel and superpoint partitions provides a promising framework for large-scale, cross-modal self-supervised learning. For instance, SLiDR [275] learns to distill image features into a 3D model, by partitioning images with SLIC and projecting the resulting superpixels onto 2.5D LiDAR range images. However, this approach does not handle scenes with multiple arbitrarily-posed 2D views and occlusions. Besides, it relies on images to partition the point cloud, ignoring the available 3D geometric information. We expect several building blocks introduced in this thesis to be helpful in exploring these directions.

Finally, as mentioned above, SPT could naturally be adapted to other modalities such as images and depth maps. Combined with DeepViewAgg, this would open the way to efficient models capable of processing multimodal, multi-view, large-scale data.

Bibliography

- [1] Karim Abou Zeid et al. « Point2vec for self-supervised representation learning on point clouds ». In: *arXiv preprint arXiv:2303.16570* (2023) (cit. on p. 9).
- [2] Radhakrishna Achanta and Sabine Susstrunk. « Superpixels and polygons using simple non-iterative clustering ». In: *CVPR* (2017) (cit. on p. 40).
- [3] Radhakrishna Achanta et al. « SLIC superpixels compared to state-of-the-art superpixel methods ». In: *TPAMI* (2012) (cit. on pp. 39–41).
- [4] Eva Agapaki and Ioannis Brilakis. « CLOI-NET: Class segmentation of industrial facilities’ point cloud datasets ». In: *Advanced Engineering Informatics* (2020) (cit. on p. 11).
- [5] Abien Fred Agarap. « Deep learning using rectified linear units (relu) ». In: *arXiv preprint arXiv:1803.08375* (2018) (cit. on p. 4).
- [6] Apoorv Agnihotri and Nipun Batra. « Exploring bayesian optimization ». In: *Distill* (2020) (cit. on p. 36).
- [7] Simegnew Yihunie Alaba and John E Ball. « A survey on deep-learning-based LiDAR 3D object detection for autonomous driving ». In: *Sensors* (2022) (cit. on pp. 2, 80).
- [8] Jean-Baptiste Alayrac et al. « Flamingo: A visual language model for few-shot learning ». In: *NeurIPS* (2022) (cit. on p. 8).
- [9] Cédric Allène, Jean-Philippe Pons, and Renaud Keriven. « Seamless image-based texture atlases using multi-band blending ». In: *ICPR* (2008) (cit. on pp. 108, 111).
- [10] Md Zahangir Alom et al. « A state-of-the-art survey on deep learning theory and architectures ». In: *Electronics* (2019) (cit. on p. 17).
- [11] Geethanjali Anjanappa. « Deep learning on 3D point clouds for safety-related asset management in buildings ». In: *University of Twente* (2022) (cit. on p. 11).
- [12] Pablo Arbelaez. « Boundary extraction in natural images using ultrametric contour maps ». In: *CVPR Workshop* (2006) (cit. on p. 41).
- [13] Iro Armeni et al. « 3D semantic parsing of large-scale indoor spaces ». In: *CVPR* (2016) (cit. on pp. 14, 20, 43, 49, 61, 62, 77, 109, 113, 123, 202).
- [14] Iro Armeni et al. « 3D Scene Graph: A structure for unified semantics, 3D space, and camera ». In: *ICCV* (2019) (cit. on pp. 78, 79, 88, 111, 207, 216).
- [15] Mehmet Aygun et al. « 4D panoptic LiDAR segmentation ». In: *CVPR* (2021) (cit. on pp. 77, 80).

- [16] Randall Balestriero et al. « A cookbook of self-supervised learning ». In: *arXiv preprint arXiv:2304.12210* (2023) (cit. on p. 13).
- [17] Hangbo Bao et al. « BEiT: Bert pre-training of image transformers ». In: *arXiv preprint arXiv:2106.08254* (2021) (cit. on p. 8).
- [18] James Bergstra and Yoshua Bengio. « Random search for hyper-parameter optimization ». In: *Journal of Machine Learning Research* (2012) (cit. on p. 36).
- [19] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. « Is space-time attention all you need for video understanding? ». In: *ICML* (2021) (cit. on p. 6).
- [20] Stan Birchfield and Carlo Tomasi. « Multiway cut for stereo and motion with slanted surfaces ». In: *ICCV* (1999) (cit. on p. 111).
- [21] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995 (cit. on p. 5).
- [22] Rishi Bommasani et al. « On the opportunities and risks of foundation models ». In: *arXiv preprint arXiv:2108.07258* (2021) (cit. on p. 8).
- [23] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. « A training algorithm for optimal margin classifiers ». In: *Proceedings of the fifth annual workshop on Computational learning theory* (1992) (cit. on p. 4).
- [24] Alexandre Boulch. « ConvPoint: Continuous convolutions for point cloud processing ». In: *Computers & Graphics* (2020) (cit. on pp. 33, 63, 94, 186, 188).
- [25] Alexandre Boulch et al. « SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks ». In: *Computers & Graphics* (2018) (cit. on pp. 9, 30, 31, 108, 110).
- [26] Yuri Boykov and Vladimir Kolmogorov. « An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision ». In: *TPAMI* (2004) (cit. on p. 40).
- [27] Yuri Boykov, Olga Veksler, and Ramin Zabih. « Fast approximate energy minimization via graph cuts ». In: *TPAMI* (2001) (cit. on p. 83).
- [28] Aljaz Bozic et al. « TransformerFusion: Monocular RGB scene reconstruction using transformers ». In: *NeurIPS* (2021) (cit. on p. 112).
- [29] Stevo Bozinovski. « Reminder of the first paper on transfer learning in neural networks, 1976 ». In: *Informatika* (2020) (cit. on p. 17).
- [30] Stevo Bozinovski and Ante Fulgosi. « The influence of pattern similarity and transfer learning upon training of a base perceptron b2 ». In: *Proceedings of Symposium Informatika* (1976) (cit. on p. 17).
- [31] Michael M Bronstein et al. « Geometric deep learning: Grids, groups, graphs, geodesics, and gauges ». In: *arXiv preprint arXiv:2104.13478* (2021) (cit. on p. 33).
- [32] Tom Brown et al. « Language models are few-shot learners ». In: *NeurIPS* (2020) (cit. on pp. 8, 13).
- [33] Chris Buehler et al. « Unstructured lumigraph rendering ». In: *SIGGRAPH* (2001) (cit. on p. 111).

- [34] Yaroslav Bulatov. *Fitting larger networks into memory*. 2018. URL: <https://medium.com/tensorflow/fitting-larger-networks-into-memory-583e3c758ff9> (cit. on p. 37).
- [35] Peter J Burt. « Fast filter transform for image processing ». In: *Computer graphics and image processing* (1981) (cit. on p. 19).
- [36] Peter J Burt and Edward H Adelson. « The laplacian pyramid as a compact image code ». In: *Readings in computer vision* (1987) (cit. on p. 19).
- [37] Holger Caesar et al. « nuScenes: A multimodal dataset for autonomous driving ». In: *CVPR* (2020) (cit. on p. 144).
- [38] Ozan Caglayan, Loïc Barrault, and Fethi Bougares. « Multimodal attention for neural machine translation ». In: *arXiv preprint arXiv:1609.03976* (2016) (cit. on p. 110).
- [39] Tianle Cai et al. « GraphNorm: A principled approach to accelerating graph neural network training ». In: *ICML* (2021) (cit. on pp. 54, 183).
- [40] Ziyun Cai et al. « RGB-D datasets using Microsoft Kinect or similar sensors: A survey ». In: *Multimedia Tools and Applications* (2017) (cit. on p. 108).
- [41] Yulong Cao and Morley Mao. *Autonomous vehicles can be fooled to “see” nonexistent obstacles*. 2020. URL: <https://theconversation.com/autonomous-vehicles-can-be-fooled-to-see-nonexistent-obstacles-129427> (cit. on p. 12).
- [42] Marc-André Carbonneau et al. « Multiple instance learning: A survey of problem characteristics and applications ». In: *Pattern Recognition* (2018) (cit. on p. 36).
- [43] Susan Carey and Elsa Bartlett. « Acquiring a single new word ». In: *ERIC* (1978) (cit. on p. 6).
- [44] Nicolas Carion et al. « End-to-end object detection with transformers ». In: *ECCV* (2020) (cit. on p. 80).
- [45] Mathilde Caron et al. « Emerging properties in self-supervised vision transformers ». In: *ICCV* (2021) (cit. on p. 8).
- [46] Thomas Chaton et al. « Torch-Points3D: A modular multi-task framework for reproducible deep learning on 3D point clouds ». In: *3DV* (2020) (cit. on pp. 63, 89, 125, 180, 200).
- [47] Dave Z. Chen. *Pointnet2.scannet*. 2021. URL: <https://github.com/daveredrum/Pointnet2.ScanNet> (cit. on p. 125).
- [48] Liang-Chieh Chen et al. « DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs ». In: *TPAMI* (2017) (cit. on p. 37).
- [49] Shaoyu Chen et al. « Hierarchical aggregation for 3D instance segmentation ». In: *ICCV* (2021) (cit. on pp. 41, 95).
- [50] Tianqi Chen et al. « Training deep nets with sublinear memory cost ». In: *arXiv preprint arXiv:1604.06174* (2016) (cit. on p. 37).
- [51] Ting Chen et al. « A simple framework for contrastive learning of visual representations ». In: *ICML* (2020) (cit. on pp. 5, 8, 36).

- [52] Xiaozhi Chen et al. « Multi-view 3D object detection network for autonomous driving ». In: *CVPR* (2017) (cit. on p. 30).
- [53] Xieyuanli Chen et al. « Moving object segmentation in 3D LiDAR data: A learning-based approach exploiting sequential data ». In: *Robotics and Automation Letters* (2021) (cit. on p. 144).
- [54] Bowen Cheng et al. « Masked-attention mask transformer for universal image segmentation ». In: *CVPR* (2022) (cit. on p. 80).
- [55] Hung-Yueh Chiang et al. « A unified point-based framework for 3D segmentation ». In: *3DV* (2019) (cit. on pp. 108, 110).
- [56] François Chollet. « Xception: Deep learning with depthwise separable convolutions ». In: *CVPR* (2017) (cit. on pp. 37, 38).
- [57] Yoni Choukroun et al. « Low-bit quantization of neural networks for efficient inference ». In: *ICCV Workshop* (2019) (cit. on p. 36).
- [58] Christopher Choy, JunYoung Gwak, and Silvio Savarese. « 4D spatio-temporal convnets: Minkowski convolutional neural networks ». In: *CVPR* (2019) (cit. on pp. 9, 20, 22, 32, 38, 43, 63, 65, 92, 94, 109, 125, 141, 186, 187, 200).
- [59] Olivier Collignon et al. « Cross-modal plasticity for the spatial processing of sounds in visually deprived subjects ». In: *Experimental Brain Research* () (cit. on p. 4).
- [60] Ronan Collobert et al. « Natural language processing (almost) from scratch ». In: *Journal of Machine Learning Research* (2011) (cit. on p. 4).
- [61] Cl Connolly. « The determination of next best views ». In: *ICRA* (1985) (cit. on p. 111).
- [62] Marius Cordts et al. « The cityscapes dataset for semantic urban scene understanding ». In: *CVPR* (2016) (cit. on p. 201).
- [63] CSAILVision. *Semantic-segmentation-pytorch*. 2020. URL: <https://github.com/CSAILVision/semantic-segmentation-pytorch> (cit. on p. 201).
- [64] George E Dahl et al. « Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition ». In: *Transactions on Audio, Speech, and Language Processing* (2011) (cit. on p. 4).
- [65] Angela Dai and Matthias Nießner. « 3DMV: Joint 3D-multi-view prediction for 3D semantic scene segmentation ». In: *ECCV* (2018) (cit. on pp. 21, 108, 110).
- [66] Angela Dai et al. « ScanNet: Richly-annotated 3D reconstructions of indoor scenes ». In: *CVPR* (2017) (cit. on pp. 14, 43, 77, 79, 89, 109, 124).
- [67] Mostafa Dehghani et al. « Scaling vision transformers to 22 billion parameters ». In: *ICLR* (2023) (cit. on p. 12).
- [68] Matt Deitke et al. « Objaverse-XL: A universe of 10M+ 3D objects ». In: *arXiv preprint arXiv:2307.05663* (2023) (cit. on p. 2).
- [69] Matt Deitke et al. « Objaverse: A universe of annotated 3D objects ». In: *CVPR* (2023) (cit. on p. 2).
- [70] Jérôme Demantké et al. « Dimensionality based scale selection in 3D LiDAR point clouds ». In: *Laserscanning* (2011) (cit. on pp. 58, 114).

- [71] Jia Deng et al. « ImageNet: A large-scale hierarchical image database ». In: *CVPR* (2009) (cit. on pp. 8, 14).
- [72] Jacob Devlin et al. « BERT: Pre-training of deep bidirectional transformers for language understanding ». In: *arXiv preprint arXiv:1810.04805* (2018) (cit. on pp. 5, 7, 8).
- [73] Rahima Djahel, Pascal Monasse, and Bruno Vallet. « A 3D segments based algorithm for heterogeneous data registration ». In: *ISPRS Archives* (2022) (cit. on p. 145).
- [74] Carl Doersch, Abhinav Gupta, and Alexei A Efros. « Unsupervised visual representation learning by context prediction ». In: *ICCV* (2015) (cit. on p. 7).
- [75] Linhao Dong, Shuang Xu, and Bo Xu. « Speech-Transformer: A no-recurrence sequence-to-sequence model for speech recognition ». In: *ICASSP* (2018) (cit. on p. 6).
- [76] Alexey Dosovitskiy et al. « An image is worth 16x16 words: Transformers for image recognition at scale ». In: *ICLR* (2020) (cit. on pp. 6, 13, 33, 142).
- [77] David Eagleman. *Can we create new senses for humans?* 2021. URL: https://www.ted.com/talks/david_eagleman_can_we_create_new_senses_for_humans (cit. on p. 5).
- [78] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. « Neural architecture search: A survey ». In: *Journal of Machine Learning Research* (2019) (cit. on p. 36).
- [79] Nico Engel, Vasileios Belagiannis, and Klaus Dietmayer. « Point transformer ». In: *ICCV* (2021) (cit. on pp. 125, 126).
- [80] Francis Engelmann et al. « 3D-MPA: Multi-proposal aggregation for 3D semantic instance segmentation ». In: *CVPR* (2020) (cit. on pp. 41, 80).
- [81] Dumitru Erhan et al. « Why does unsupervised pre-training help deep learning? » In: *Journal of Machine Learning Research* (2010) (cit. on p. 5).
- [82] Yuxue Fan, Yan Huang, and Jingliang Peng. « Point cloud compression based on hierarchical point clustering ». In: *APSIPA ASC* (2013) (cit. on p. 41).
- [83] *Faro Freestyle 2 handheld scanner*. Accessed: 2023-11-15. URL: <https://www.faro.com/en/Products/Hardware/Freestyle-2-Handheld-Scanner> (cit. on pp. 20, 107).
- [84] Olivier Faugeras, Quang-Tuan Luong, and Theo Papadopoulos. *The geometry of multiple images: The laws that govern the formation of multiple images of a scene and some of their applications*. MIT Press, 2001 (cit. on p. 9).
- [85] Pedro F Felzenszwalb and Daniel P Huttenlocher. « Efficient graph-based image segmentation ». In: *IJCV* (2004) (cit. on p. 40).
- [86] Yifan Feng et al. « GVCNN: Group-view convolutional neural networks for 3D shape recognition ». In: *CVPR* (2018) (cit. on pp. 108, 111).
- [87] Marc Finzi et al. « Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data ». In: *ICML* (2020) (cit. on p. 5).

- [88] Martin A Fischler and Robert C Bolles. « Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography ». In: *Communications of the ACM* (1981) (cit. on p. 58).
- [89] Whye Kit Fong et al. « Panoptic nuScenes: A large-scale benchmark for LiDAR panoptic segmentation and tracking ». In: *Robotics and Automation Letters* (2022) (cit. on pp. 77, 80).
- [90] Barak Freedman. « Depth mapping using projected patterns ». In: *US Application Publication, US 2010/0118123 A1* (2010) (cit. on p. 9).
- [91] Raghudeep Gadde et al. « Superpixel convolutional networks using bilateral inceptions ». In: *ECCV* (2016) (cit. on p. 39).
- [92] Ran Gal et al. « Seamless montage for texturing models ». In: *Computer Graphics Forum* (2010) (cit. on p. 111).
- [93] Hongyang Gao and Shuiwang Ji. « Graph U-Nets ». In: *ICML* (2019) (cit. on p. 52).
- [94] geospatialworld. *What is LiDAR technology and how does it work?* 2022. URL: <https://www.geospatialworld.net/prime/technology-and-innovation/what-is-lidar-technology-and-how-does-it-work> (cit. on p. 10).
- [95] Muriel Gevrey, Ioannis Dimopoulos, and Sovan Lek. « Review and comparison of methods to study the contribution of variables in artificial neural network models ». In: *Ecological Modelling* (2003) (cit. on p. 130).
- [96] Charlie Giattino et al. « Artificial intelligence ». In: *Our World in Data* (2022). <https://ourworldindata.org/artificial-intelligence> (cit. on p. 47).
- [97] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. « Unsupervised representation learning by predicting image rotations ». In: *arXiv preprint arXiv:1803.07728* (2018) (cit. on pp. 7, 8, 36).
- [98] GISGeography. *Top 6 free LiDAR data sources*. 2023. URL: <https://gisgeography.com/top-6-free-lidar-data-sources> (cit. on p. 12).
- [99] Yuan Gong, Yu-An Chung, and James Glass. « AST: Audio spectrogram transformer ». In: *arXiv preprint arXiv:2104.01778* (2021) (cit. on p. 6).
- [100] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016. URL: <http://www.deeplearningbook.org> (cit. on p. 3).
- [101] Ian Goodfellow et al. « Generative adversarial nets ». In: *NeurIPS* (2014) (cit. on p. 4).
- [102] Anirudh Goyal and Yoshua Bengio. « Inductive biases for deep learning of higher-level cognition ». In: *Proceedings of the Royal Society A* (2022) (cit. on p. 5).
- [103] Priya Goyal et al. « Self-supervised pretraining of visual features in the wild ». In: *arXiv preprint arXiv:2103.01988* (2021) (cit. on p. 8).
- [104] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. « 3D semantic segmentation with submanifold sparse convolutional networks ». In: *CVPR* (2018) (cit. on p. 38).
- [105] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. « Speech recognition with deep recurrent neural networks ». In: *ICASSP* (2013) (cit. on p. 4).

- [106] Jean-Bastien Grill et al. « Bootstrap your own latent-a new approach to self-supervised learning ». In: *NeurIPS* (2020) (cit. on p. 8).
- [107] Yue Gu et al. « Multimodal affective analysis using hierarchical attention strategy with word-level alignment ». In: *Association for Computational Linguistics* (2018) (cit. on p. 110).
- [108] Stéphane Guinard and Loic Landrieu. « Weakly supervised segmentation-aided classification of urban scenes from 3D LiDAR point clouds ». In: *ISPRS Workshop* (2017) (cit. on pp. 51, 58).
- [109] Meng-Hao Guo et al. « PCT: Point Cloud Transformer ». In: *CVM* (2021) (cit. on p. 47).
- [110] Yulan Guo et al. « Deep learning for 3D point clouds: A survey ». In: *TPAMI* (2020) (cit. on pp. 30, 80, 107, 110).
- [111] Timo Hackel et al. « Semantic3D.Net: A new large-scale point cloud classification benchmark ». In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2017) (cit. on pp. 18, 43, 109).
- [112] Abdullah Hamdi, Silvio Giancola, and Bernard Ghanem. « MVTN: Multi-view transformation network for 3D shape recognition ». In: *ICCV* (2021) (cit. on p. 111).
- [113] Lei Han et al. « OccuSeg: Occupancy-aware 3D instance segmentation ». In: *CVPR* (2020) (cit. on pp. 41, 80, 97).
- [114] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003 (cit. on p. 9).
- [115] Babak Hassibi, David G Stork, and Gregory J Wolff. « Optimal brain surgeon and general network pruning ». In: *International Conference on Neural Networks* (1993) (cit. on p. 35).
- [116] Marius Hauglin et al. « Large scale mapping of forest attributes using heterogeneous sets of airborne laser scanning and national forest inventory data ». In: *Forest Ecosystems* (2021) (cit. on pp. 2, 77).
- [117] Caner Hazirbas et al. « FuseNet: Incorporating depth into semantic segmentation via fusion-based CNN architecture ». In: *ACCV* (2016) (cit. on pp. 110, 121).
- [118] Kaiming He et al. « Deep residual learning for image recognition ». In: *CVPR* (2016) (cit. on pp. 4, 5, 54, 201).
- [119] Kaiming He et al. « Identity mappings in deep residual networks ». In: *ECCV* (2016) (cit. on p. 200).
- [120] Kaiming He et al. « Momentum contrast for unsupervised visual representation learning ». In: *CVPR* (2020) (cit. on pp. 8, 36).
- [121] Kaiming He et al. « Masked autoencoders are scalable vision learners ». In: *CVPR* (2022) (cit. on pp. 8, 36).
- [122] Tong He, Chunhua Shen, and Anton van den Hengel. « DyCo3D: Robust instance segmentation of 3D point clouds through dynamic convolution ». In: *CVPR* (2021) (cit. on p. 80).

- [123] Yong He et al. « Deep learning based 3D segmentation: A survey ». In: *arXiv preprint arXiv:2103.05423* (2021) (cit. on p. 97).
- [124] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. « Distilling the knowledge in a neural network ». In: *arXiv preprint arXiv:1503.02531* (2015) (cit. on p. 36).
- [125] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. « A fast learning algorithm for deep belief nets ». In: *Neural computation* (2006) (cit. on p. 5).
- [126] Tin Kam Ho. « Random decision forests ». In: *Proceedings of the International Conference on Document Analysis and Recognition* (1995) (cit. on p. 4).
- [127] Sepp Hochreiter and Jürgen Schmidhuber. « Long short-term memory ». In: *Neural computation* (1997) (cit. on p. 5).
- [128] David Hodgetts. « Laser scanning and digital outcrop geology in the petroleum industry: A review ». In: *Marine and Petroleum Geology* (2013) (cit. on p. 108).
- [129] Chiori Hori et al. « Attention-based multimodal fusion for video description ». In: *ICCV* (2017) (cit. on p. 110).
- [130] Ji Hou et al. « Mask3D: Pre-training 2D vision transformers by learning masked 3D priors ». In: *CVPR* (2023) (cit. on p. 9).
- [131] Pai-Hui Hsu and Zong-Yi Zhuang. « Incorporating handcrafted features into deep learning for point cloud classification ». In: *Remote Sensing* (2020) (cit. on pp. 65, 70).
- [132] Edward J Hu et al. « LoRA: Low-rank adaptation of large language models ». In: *arXiv preprint arXiv:2106.09685* (2021) (cit. on p. 36).
- [133] Qingyong Hu et al. « RandLA-Net: Efficient semantic segmentation of large-scale point clouds ». In: *CVPR* (2020) (cit. on pp. 38, 63, 125).
- [134] Qingyong Hu et al. « Towards semantic segmentation of urban-scale 3D point clouds: A dataset, benchmarks and challenges ». In: *CVPR* (2021) (cit. on p. 12).
- [135] Wenbo Hu et al. « Bidirectional projection network for cross dimension scene understanding ». In: *CVPR* (2021) (cit. on pp. 21, 108, 110, 125, 126, 133).
- [136] Xiao Shi Huang et al. « Improving transformer optimization through better initialization ». In: *ICML* (2020) (cit. on p. 13).
- [137] Yanping Huang et al. « GPipe: Efficient training of giant neural networks using pipeline parallelism ». In: *NeurIPS* (2019) (cit. on p. 37).
- [138] Po-Yao Huang et al. « Attention-based multimodal neural machine translation ». In: *Conference on Machine Translation* (2016) (cit. on p. 110).
- [139] Itay Hubara et al. « Quantized neural networks: Training neural networks with low precision weights and activations ». In: *Journal of Machine Learning Research* (2017) (cit. on p. 36).
- [140] Le Hui et al. « Superpoint network for point cloud oversegmentation ». In: *ICCV* (2021) (cit. on pp. 41, 48, 51, 63, 69, 80, 143).
- [141] Yerlan Idelbayev and Miguel A Carreira-Perpinán. « Low-rank compression of neural nets: Learning the rank of each layer ». In: *CVPR* (2020) (cit. on p. 36).

- [142] IGN. *LiDAR HD*. Accessed: 2023-11-15. URL: <https://geoservices.ign.fr/lidarhd> (cit. on p. 12).
- [143] Sergey Ioffe and Christian Szegedy. « Batch normalization: Accelerating deep network training by reducing internal covariate shift ». In: *ICML* (2015) (cit. on pp. 4, 201).
- [144] Stefan Isler et al. « An information gain formulation for active volumetric 3D reconstruction ». In: *ICRA* (2016) (cit. on p. 111).
- [145] Michel Jaboyedoff et al. « Use of LiDAR in landslide investigations: A review ». In: *Natural hazards* (2012) (cit. on p. 11).
- [146] Benoit Jacob et al. « Quantization and training of neural networks for efficient integer-arithmetical-only inference ». In: *CVPR* (2018) (cit. on p. 36).
- [147] Andrew Jaegle et al. « Perceiver: General perception with iterative attention ». In: *ICML* (2021) (cit. on pp. 5, 6).
- [148] Varun Jampani et al. « Superpixel sampling networks ». In: *ECCV* (2018) (cit. on pp. 39, 40, 143).
- [149] Maximilian Jaritz, Jiayuan Gu, and Hao Su. « Multi-view pointnet for 3D scene understanding ». In: *CVPR Workshops* (2019) (cit. on pp. 21, 108, 110, 121, 125, 133).
- [150] Albert V Jelalian. *Laser radar systems*. Artech House Publishers, 1992 (cit. on pp. 9, 19, 43, 109).
- [151] Ding Jia et al. « DETRs with hybrid matching ». In: *CVPR* (2023) (cit. on p. 80).
- [152] Li Jiang et al. « PointGroup: Dual-set point grouping for 3D instance segmentation ». In: *CVPR* (2020) (cit. on pp. 80, 95, 97).
- [153] Yuchen Jiang et al. « Industrial applications of digital twins ». In: *Philosophical Transactions of the Royal Society A* (2021) (cit. on pp. 2, 77).
- [154] Arttu Julin et al. « Evaluating the quality of TLS point cloud colorization ». In: *MDPI Remote Sensing* (2020) (cit. on pp. 43, 107, 109).
- [155] John Jumper et al. « Highly accurate protein structure prediction with alphafold ». In: *Nature* (2021) (cit. on p. 4).
- [156] Ekaterina Kalinicheva et al. « Multi-layer modeling of dense vegetation from aerial LiDAR scans ». In: *CVPR Workshop* (2022) (cit. on p. 11).
- [157] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. « RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints ». In: *CVPR* (2018) (cit. on p. 111).
- [158] Xin Kang, Chaoqun Wang, and Xuejin Chen. « Region-enhanced feature learning for scene semantic segmentation ». In: *arXiv preprint arXiv:2304.07486* (2023) (cit. on p. 41).
- [159] Jared Kaplan et al. « Scaling laws for neural language models ». In: *arXiv preprint arXiv:2001.08361* (2020) (cit. on pp. 12, 14, 15, 17).
- [160] Andrej Karpathy et al. « Large-scale video classification with convolutional neural networks ». In: *CVPR* (2014) (cit. on p. 4).

- [161] Giseop Kim and Ayoung Kim. « Remove, then revert: Static point cloud map construction using multiresolution range images ». In: *IROS* (2020) (cit. on p. 144).
- [162] Alexander Kirillov et al. « Panoptic segmentation ». In: *CVPR* (2019) (cit. on pp. 42, 77, 82).
- [163] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. « Reformer: The efficient transformer ». In: *arXiv preprint arXiv:2001.04451* (2020) (cit. on p. 38).
- [164] Laurent Kneip. « Real-time scalable structure from motion: From fundamental geometric vision to collaborative mapping ». In: *ETH Zurich* (2012) (cit. on p. 10).
- [165] Vladimir Kolmogorov and Ramin Zabini. « What energy functions can be minimized via graph cuts? ». In: *TPAMI* (2004) (cit. on p. 81).
- [166] Thomas Kreutz, Max Mühlhäuser, and Alejandro Sanchez Guinea. « Unsupervised 4D LiDAR moving object segmentation in stationary settings with multivariate occupancy time series ». In: *WACV* (2023) (cit. on p. 144).
- [167] Raghuraman Krishnamoorthi. « Quantizing deep convolutional networks for efficient inference: A whitepaper ». In: *arXiv preprint arXiv:1806.08342* (2018) (cit. on p. 36).
- [168] Georg Krispel et al. « FuseSeg: LiDAR point cloud segmentation fusing multi-modal data ». In: *WACV* (2020) (cit. on pp. 110, 121).
- [169] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. « Imagenet classification with deep convolutional neural networks ». In: *NeurIPS* (2012) (cit. on pp. 4, 5, 36).
- [170] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. « Regularization for deep learning: A taxonomy ». In: *arXiv preprint arXiv:1710.10686* (2017) (cit. on p. 5).
- [171] Abhijit Kundu et al. « Virtual multi-view fusion for 3D semantic segmentation ». In: *ECCV* (2020) (cit. on pp. 108, 110, 125, 133).
- [172] Patrick Labatut, J-P Pons, and Renaud Keriven. « Robust and efficient surface reconstruction from range data ». In: *Computer Graphics Forum* (2009) (cit. on p. 83).
- [173] Florent Lafarge and Clément Mallet. « Creating large-scale city models from 3D-point clouds: A robust approach with hybrid representation ». In: *IJCV* (2012) (cit. on pp. 11, 77).
- [174] Jean Lahoud et al. « 3D instance segmentation via multi-task metric learning ». In: *ICCV* (2019) (cit. on p. 97).
- [175] Xin Lai et al. « Stratified transformer for 3D point cloud segmentation ». In: *CVPR* (2022) (cit. on pp. 9, 20, 34, 47, 49, 63, 65, 92, 93, 95, 141, 142, 186).
- [176] Barbara Landau, Linda B Smith, and Susan S Jones. « The importance of shape in early lexical learning ». In: *Cognitive development* (1988) (cit. on p. 6).
- [177] Loic Landrieu and Mohamed Boussaha. « Point cloud oversegmentation with graph-structured deep metric learning ». In: *CVPR* (2019) (cit. on pp. 41, 48, 51, 63, 69, 80, 86, 125, 142, 186).

- [178] Loic Landrieu and Guillaume Obozinski. « Cut Pursuit: Fast algorithms to learn piecewise constant functions ». In: *AISTATS* (2016) (cit. on pp. 41, 60, 181).
- [179] Loic Landrieu and Guillaume Obozinski. « Cut Pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs ». In: *SIAM Journal on Imaging Sciences* (2017) (cit. on pp. 52, 81, 85, 142).
- [180] Loic Landrieu and Martin Simonovsky. « Large-scale point cloud semantic segmentation with superpoint graphs ». In: *CVPR* (2018) (cit. on pp. 9, 34, 41–43, 48, 51, 57, 59, 63, 65, 80, 125, 142, 186, 188).
- [181] Loic Landrieu et al. « A structured regularization framework for spatially smoothing semantic labelings of 3D point clouds ». In: *ISPRS Journal of Photogrammetry and Remote Sensing* (2017) (cit. on pp. 83, 85).
- [182] Yvan G Leclerc. « Constructing simple stable descriptions for image partitioning ». In: *IJCV* (1989) (cit. on pp. 83, 85).
- [183] Yann LeCun. *Self-supervised learning*. 2019. URL: <https://www.facebook.com/722677142/posts/10155934004262143> (cit. on p. 7).
- [184] Yann LeCun, Yoshua Bengio, et al. « Convolutional networks for images, speech, and time series ». In: *The handbook of brain theory and neural networks* (1995) (cit. on p. 5).
- [185] Yann LeCun, John Denker, and Sara Solla. « Optimal brain damage ». In: *NeurIPS* (1989) (cit. on p. 35).
- [186] Yann LeCun et al. « Gradient-based learning applied to document recognition ». In: *Proceedings of the IEEE* (1998) (cit. on pp. 4, 37).
- [187] Xiangtai Lee. *Sfsegnets*. 2021. URL: <https://github.com/lxtGH/SFSegNets> (cit. on p. 201).
- [188] *Leica Rtc360 3D laser scanner*. Accessed: 2023-11-15. URL: <https://leica-geosystems.com/products/laser-scanners/scanners/leica-rtc360> (cit. on pp. 20, 107).
- [189] Victor Lempitsky and Denis Ivanov. « Seamless mosaicing of image-based texture maps ». In: *CVPR* (2007) (cit. on pp. 108, 111).
- [190] Hao Li et al. « Training quantized nets: A deeper understanding ». In: *NeurIPS* (2017) (cit. on p. 36).
- [191] Jiaxin Li and Gim Hee Lee. « DeepI2P: Image-to-point cloud registration via deep classification ». In: *CVPR* (2021) (cit. on p. 145).
- [192] Siqi Li et al. « Attention-based multi-modal fusion network for semantic scene completion ». In: *AAAI* (2020) (cit. on p. 110).
- [193] Yangyan Li et al. « PointCNN: Convolution on χ -transformed points ». In: *NeurIPS* (2018) (cit. on pp. 33, 188).
- [194] Ying Li et al. « Deep learning for LiDAR point clouds in autonomous driving: A review ». In: *Transactions on Neural Networks and Learning Systems* (2020) (cit. on pp. 2, 11).

- [195] Zhengqin Li and Jiansheng Chen. « Superpixel segmentation using linear spectral clustering ». In: *CVPR* (2015) (cit. on p. 40).
- [196] Zhihao Liang et al. « Instance segmentation in 3D scenes using semantic superpoint tree networks ». In: *CVPR* (2021) (cit. on pp. 41, 81).
- [197] Yiyi Liao, Jun Xie, and Andreas Geiger. « KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2D and 3D ». In: *TPAMI* (2022) (cit. on pp. 20, 43, 49, 61, 62, 77, 79, 89, 109, 124, 125, 143).
- [198] Anne-Laure Ligozat et al. « Unraveling the hidden environmental impacts of AI solutions for environment ». In: *arXiv preprint arXiv:2110.11822* (2021) (cit. on p. 13).
- [199] Liqiang Lin et al. « Capturing, reconstructing, and simulating: The Urbanscene3D dataset ». In: *ECCV* (2022) (cit. on p. 12).
- [200] Yangbin Lin et al. « Toward better boundary preserved supervoxel segmentation for 3D point clouds ». In: *ISPRS Journal of Photogrammetry and Remote Sensing* (2018) (cit. on pp. 41, 80).
- [201] Seppo Linnainmaa. « The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors ». In: *Master's Thesis (in Finnish), Univ. Helsinki* (1970) (cit. on p. 16).
- [202] Liyuan Liu et al. « Understanding the difficulty of training transformers ». In: *arXiv preprint arXiv:2004.08249* (2020) (cit. on p. 13).
- [203] Ming-Yu Liu et al. « Entropy rate superpixel segmentation ». In: *CVPR* (2011) (cit. on p. 40).
- [204] Shuo Liu et al. « Audio self-supervised learning: A survey ». In: *Patterns* (2022) (cit. on p. 8).
- [205] Yong-Jin Liu et al. « Manifold SLIC: A fast method to compute content-sensitive superpixels ». In: *CVPR* (2016) (cit. on p. 40).
- [206] Ze Liu et al. « Swin Transformer: Hierarchical vision transformer using shifted windows ». In: *CVPR* (2021) (cit. on pp. 33, 141, 142).
- [207] Zhijian Liu et al. « Point-voxel CNN for efficient 3D deep learning ». In: *NeurIPS* (2019) (cit. on pp. 32, 39, 92).
- [208] Zhuang Liu et al. « A convnet for the 2020s ». In: *CVPR* (2022) (cit. on p. 38).
- [209] Romain Loiseau, Mathieu Aubry, and Loïc Landrieu. « Online segmentation of LiDAR sequences: Dataset and algorithm ». In: *ECCV* (2022) (cit. on pp. 39, 47, 80).
- [210] Jonathan Long, Evan Shelhamer, and Trevor Darrell. « Fully convolutional networks for semantic segmentation ». In: *CVPR* (2015) (cit. on p. 5).
- [211] Xiang Long et al. « Multimodal keyless attention fusion for video classification ». In: *AAAI* (2018) (cit. on p. 110).
- [212] Ilya Loshchilov and Frank Hutter. « Decoupled weight decay regularization ». In: *arXiv preprint arXiv:1711.05101* (2017) (cit. on p. 63).

- [213] Ilya Loshchilov and Frank Hutter. « SGDR: Stochastic gradient descent with warm restarts ». In: *ICLR* (2017) (cit. on p. 63).
- [214] Jiasen Lu et al. « Hierarchical question-image co-attention for visual question answering ». In: *NeurIPS* (2016) (cit. on p. 110).
- [215] Stéphane Mallat. « Group invariant scattering ». In: *Communications on Pure and Applied Mathematics* (2012) (cit. on p. 19).
- [216] Ellen M Markman. *Categorization and naming in children: Problems of induction*. MIT Press, 1989 (cit. on p. 6).
- [217] Daniel Maturana and Sebastian Scherer. « VoxNet: A 3D convolutional neural network for real-time object recognition ». In: *IROS* (2015) (cit. on p. 32).
- [218] Warren S McCulloch and Walter Pitts. « A logical calculus of the ideas immanent in nervous activity ». In: *The bulletin of mathematical biophysics* (1943) (cit. on p. 4).
- [219] Miguel Mendoza et al. « Supervised learning of the next-best-view for 3D object reconstruction ». In: *Pattern Recognition Letters* (2020) (cit. on p. 111).
- [220] Gaurav Menghani. « Efficient deep learning: A survey on making deep learning models smaller, faster, and better ». In: *ACM Computing Surveys* (2023) (cit. on p. 35).
- [221] Paulius Micikevicius et al. « Mixed precision training ». In: *arXiv preprint arXiv:1710.03740* (2017) (cit. on pp. 16, 37).
- [222] Sabrina J Mielke et al. « Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp ». In: *arXiv preprint arXiv:2112.10508* (2021) (cit. on p. 142).
- [223] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. « Shuffle and learn: Unsupervised learning using temporal order verification ». In: *ECCV* (2016) (cit. on p. 7).
- [224] Jonas Močkus. « On bayesian methods for seeking the extremum ». In: *Optimization Techniques IFIP Technical Conference* (1975) (cit. on p. 36).
- [225] Rohit Mohan and Abhinav Valada. « EfficientPS: Efficient panoptic segmentation ». In: *IJCV* (2021) (cit. on p. 80).
- [226] Farzin Mokhtarian and Sadegh Abbasi. « Automatic selection of optimal views in multi-view object recognition ». In: *BMVC* (2000) (cit. on p. 111).
- [227] David Bryant Mumford and Jayant Shah. « Optimal approximations by piecewise smooth functions and associated variational problems ». In: *Communications on Pure and Applied Mathematics* (1989) (cit. on pp. 83, 85).
- [228] Vinod Nair and Geoffrey E Hinton. « Rectified linear units improve restricted boltzmann machines ». In: *ICML* (2010) (cit. on p. 120).
- [229] J Narasimhamurthy et al. « Hierarchical-based semantic segmentation of 3D point cloud using deep learning ». In: *Smart Computer Vision* (2023) (cit. on p. 41).

- [230] Deepak Narayanan et al. « Efficient large-scale language model training on gpu clusters using megatron-lm ». In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2021) (cit. on p. 8).
- [231] Gaku Narita et al. « PanopticFusion: Online volumetric semantic mapping at the level of stuff and things ». In: *IROS* (2019) (cit. on pp. 80, 93, 95).
- [232] NavVis. *How to work more efficiently with large point cloud datasets*. 2019. URL: <https://www.navvis.com/blog/how-to-work-more-efficiently-with-large-point-cloud-datasets> (cit. on p. 12).
- [233] Alexey Nekrasov et al. « Mix3D: Out-of-context data augmentation for 3D scenes ». In: *3DV* (2021) (cit. on p. 125).
- [234] Tuan Duc Ngo, Binh-Son Hua, and Khoi Nguyen. « ISNBET: A 3D point cloud instance segmentation network with instance-aware sampling and box-aware dynamic convolution ». In: *CVPR* (2023) (cit. on pp. 80, 95).
- [235] Toan Q Nguyen and Julian Salazar. « Transformers without tears: Improving the normalization of self-attention ». In: *arXiv preprint arXiv:1910.05895* (2019) (cit. on p. 13).
- [236] Alex Nichol et al. « GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models ». In: *arXiv preprint arXiv:2112.10741* (2021) (cit. on p. 8).
- [237] Alex Nichol et al. « Point-E: A system for generating 3D point clouds from complex prompts ». In: *arXiv preprint arXiv:2212.08751* (2022) (cit. on p. 9).
- [238] Steven A Niederer et al. « Scaling digital twins from the artisanal to the industrial ». In: *Nature Computational Science* (2021) (cit. on p. 77).
- [239] Timea Nochtá et al. « A socio-technical perspective on urban analytics: The case of city-scale digital twins ». In: *Journal of Urban Technology* (2021) (cit. on p. 77).
- [240] Thierry Oggier et al. « An all-solid-state optical range camera for 3D real-time imaging with sub-centimeter depth resolution (swissranger) ». In: *Optical Design and Engineering* (2004) (cit. on pp. 9, 43, 109).
- [241] Aaron van den Oord et al. « WaveNet: A generative model for raw audio ». In: *arXiv preprint arXiv:1609.03499* (2016) (cit. on p. 4).
- [242] OpenAI. *Introducing chatgpt*. 2023. URL: <https://openai.com/blog/chatgpt> (cit. on p. 14).
- [243] Jeremie Papon et al. « Voxel cloud connectivity segmentation-supervoxels for point clouds ». In: *CVPR* (2013) (cit. on pp. 41, 80).
- [244] Chungyun Park et al. « Fast point transformer ». In: *CVPR* (2022) (cit. on p. 47).
- [245] Prashant W Patil et al. « An end-to-end edge aggregation network for moving object segmentation ». In: *CVPR* (2020) (cit. on p. 144).
- [246] David Patterson et al. « Carbon emissions and large neural network training ». In: *arXiv preprint arXiv:2104.10350* (2021) (cit. on p. 13).

- [247] Songyou Peng et al. « OpenScene: 3D scene understanding with open vocabularies ». In: *CVPR* (2023) (cit. on p. 9).
- [248] Massimiliano Pepe et al. « 3D point cloud model color adjustment by combining terrestrial laser scanner and close range photogrammetry datasets ». In: *International Conference on Digital Heritage* (2016) (cit. on p. 108).
- [249] Hieu Pham et al. « Efficient neural architecture search via parameters sharing ». In: *ICML* (2018) (cit. on p. 5).
- [250] Marin Vlastelica Pogančić et al. « Differentiation of blackbox combinatorial solvers ». In: *ICLR* (2019) (cit. on p. 143).
- [251] Renfrey Burnard Potts. « Some generalized order-disorder transformations ». In: *Mathematical Proceedings of the Cambridge Philosophical Society* (1952) (cit. on p. 85).
- [252] Charles R Qi et al. « PointNet: Deep learning on point sets for 3D classification and segmentation ». In: *CVPR* (2017) (cit. on pp. 9, 32, 38, 43, 109, 141, 186).
- [253] Charles R Qi et al. « PointNet++: Deep hierarchical feature learning on point sets in a metric space ». In: *NeurIPS* (2017) (cit. on pp. 9, 20, 32, 38, 63, 65, 92, 94, 125, 188).
- [254] Guocheng Qian et al. « Pix4Point: Image pretrained transformers for 3D point cloud understanding ». In: *arXiv preprint arXiv:2208.12259* (2022) (cit. on p. 9).
- [255] Guocheng Qian et al. « PointNeXt: Revisiting PointNet++ with improved training and scaling strategies ». In: *NeurIPS* (2022) (cit. on pp. 20, 33, 38, 43, 49, 63, 92, 93, 109).
- [256] Xingwen Quana et al. « Hierarchical semantic segmentation of urban scene point clouds via group proposal and graph attention network ». In: *International Journal of Applied Earth Observations and Geoinformation* (2016) (cit. on p. 48).
- [257] Camillo Quattrocchi et al. « Panoptic segmentation in industrial environments using synthetic and real data ». In: *ICIAP* (2022) (cit. on p. 77).
- [258] Alec Radford et al. « Improving language understanding by generative pre-training ». In: *OpenAI blog* (2018) (cit. on pp. 7, 8).
- [259] Alec Radford et al. « Language models are unsupervised multitask learners ». In: *OpenAI blog* (2019) (cit. on p. 8).
- [260] Alec Radford et al. « Learning transferable visual models from natural language supervision ». In: *ICML* (2021) (cit. on p. 8).
- [261] Hugo Raguey and Loic Landrieu. « Parallel cut pursuit for minimization of the graph total variation ». In: *ICML Workshop on Graph Reasoning* (2019) (cit. on pp. 52, 85, 142).
- [262] Aditya Ramesh et al. « Zero-shot text-to-image generation ». In: *ICML* (2021) (cit. on pp. 8, 14).
- [263] Haoxi Ran, Jun Liu, and Chengjie Wang. « Surface representation for point clouds ». In: *CVPR* (2022) (cit. on pp. 63, 65, 70).

- [264] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Póczos. « Equivariance through parameter-sharing ». In: *ICML* (2017) (cit. on p. 5).
- [265] Xiaofeng Ren and Jitendra Malik. « Learning a classification model for segmentation ». In: *ICCV* (2003) (cit. on pp. 39, 40).
- [266] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. « OctNet: Learning deep 3D representations at high resolutions ». In: *CVPR* (2017) (cit. on pp. 32, 41).
- [267] Damien Robert, Hugo Raguét, and Loïc Landrieu. « Efficient 3D semantic segmentation with superpoint transformer ». In: *ICCV* (2023) (cit. on pp. 23, 80, 87, 90, 92–94, 190, 191).
- [268] Damien Robert, Hugo Raguét, and Loïc Landrieu. « Scalable 3D panoptic segmentation as superpoint graph clustering ». In: *3DV* (2024) (cit. on p. 23).
- [269] Damien Robert, Bruno Vallet, and Loïc Landrieu. « Learning multi-view aggregation in the wild for large-scale 3D semantic segmentation ». In: *CVPR* (2022) (cit. on pp. 63, 93, 94, 186, 187).
- [270] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. « U-Net: Convolutional networks for biomedical image segmentation ». In: *MICCAI* (2015) (cit. on pp. 52, 54, 120).
- [271] Frank Rosenblatt. « The Perceptron: A probabilistic model for information storage and organization in the brain ». In: *Psychological review* (1958) (cit. on p. 4).
- [272] Corby Rosset. *Turing-NLG: A 17-billion-parameter language model by microsoft*. 2020. URL: <https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft> (cit. on p. 15).
- [273] Mark Sandler et al. « MobileNetV2: Inverted residuals and linear bottlenecks ». In: *CVPR* (2018) (cit. on p. 38).
- [274] K Sathian and Randall Stilla. « Cross-modal plasticity of tactile perception in blindness ». In: *Restorative Neurology and Neuroscience* (2010) (cit. on p. 4).
- [275] Corentin Sautier et al. « Image-to-LiDAR self-supervised distillation for autonomous driving data ». In: *CVPR* (2022) (cit. on p. 145).
- [276] Stefano Savazzi et al. « An energy and carbon footprint analysis of distributed and federated learning ». In: *Transactions on Green Communications and Networking* (2022) (cit. on p. 13).
- [277] Madeline C Schiappa, Yogesh S Rawat, and Mubarak Shah. « Self-supervised learning for videos: A survey ». In: *ACM Computing Surveys* (2023) (cit. on p. 8).
- [278] Johannes L Schönberger et al. « Pixelwise view selection for unstructured multi-view stereo ». In: *ECCV* (2016) (cit. on pp. 31, 111).
- [279] Christoph Schuhmann et al. « LAION-5B: An open large-scale dataset for training next generation image-text models ». In: *NeurIPS* (2022) (cit. on p. 2).
- [280] Jonas Schult et al. « Mask3D: Mask transformer for 3D semantic instance segmentation ». In: *ICRA* (2023) (cit. on pp. 78, 80, 95).

- [281] William R Scott, Gerhard Roth, and Jean-François Rivest. « View planning for automated three-dimensional object reconstruction and inspection ». In: *Computing Surveys* (2003) (cit. on p. 111).
- [282] Claude E Shannon. « Communication in the presence of noise ». In: *Proceedings of the IRE* (1949) (cit. on p. 18).
- [283] Abhishek Sharma, Oncel Tuzel, and Ming-Yu Liu. « Recursive context propagation network for semantic scene labeling ». In: *NeurIPS* (2014) (cit. on p. 39).
- [284] Guang Shu, Afshin Dehghan, and Mubarak Shah. « Improving an object detector and extracting regions using superpixels ». In: *CVPR* (2013) (cit. on p. 39).
- [285] Nathan Silberman et al. « Indoor segmentation and support inference from RGBD images ». In: *ECCV* (2012) (cit. on p. 10).
- [286] David Silver et al. « Mastering the game of go with deep neural networks and tree search ». In: *Nature* (2016) (cit. on p. 4).
- [287] Martin Simonovsky and Nikos Komodakis. « Dynamic edge-conditioned filters in convolutional neural networks on graphs ». In: *CVPR* (2017) (cit. on pp. 33, 41, 48).
- [288] Nina M Singer and Vijayan K Asari. « DALES Objects: A large scale benchmark dataset for instance segmentation in aerial LiDAR ». In: *IEEE Access* (2021) (cit. on pp. 12, 49, 61, 62, 77, 79, 89).
- [289] Julien Smeckaert et al. « Large-scale classification of water areas using airborne topographic LiDAR data ». In: *Remote sensing of environment* (2013) (cit. on pp. 2, 77).
- [290] Nitish Srivastava et al. « Dropout: A simple way to prevent neural networks from overfitting ». In: *Journal of Machine Learning Research* (2014) (cit. on p. 5).
- [291] Wolfgang Strasser. « Fast curve and surface generation for interactive shape design ». In: *Computers in Industry* (1982) (cit. on p. 113).
- [292] Hang Su et al. « Multi-view convolutional neural networks for 3D shape recognition ». In: *CVPR* (2015) (cit. on pp. 30, 108, 110, 111).
- [293] Chen Sun et al. « Revisiting unreasonable effectiveness of data in deep learning era ». In: *ICCV* (2017) (cit. on p. 36).
- [294] Jiahao Sun et al. « Superpoint transformer for 3D scene instance segmentation ». In: *AAAI* (2023) (cit. on pp. 80, 81).
- [295] Rich Sutton. *The bitter lesson*. 2019. URL: <http://www.incompleteideas.net/IncIdeas/BitterLesson.html> (cit. on p. 6).
- [296] Mingxing Tan and Quoc Le. « EfficientNet: Rethinking model scaling for convolutional neural networks ». In: *ICML* (2019) (cit. on p. 36).
- [297] Haotian Tang et al. « Searching efficient 3D architectures with sparse point-voxel convolution ». In: *ECCV* (2020) (cit. on p. 32).
- [298] Yi Tay et al. « Efficient transformers: A survey ». In: *ACM Computing Surveys (CSUR)* (2020) (cit. on p. 38).

- [299] Lyne Tchapmi et al. « SEGCloud: Semantic segmentation of 3D point clouds ». In: *3DV* (2017) (cit. on p. 32).
- [300] Gusi Te et al. « RGCNN: Regularized graph CNN for point cloud segmentation ». In: *ACM International Conference on Multimedia* (2018) (cit. on p. 33).
- [301] Hugues Thomas et al. « KPConv: Flexible and deformable convolution for point clouds ». In: *ICCV* (2019) (cit. on pp. 20, 33, 43, 63, 65, 89, 92–94, 109, 125, 141, 180, 186, 188).
- [302] Anirud Thyagarajan et al. « Segment-Fusion: Hierarchical context fusion for robust 3D semantic segmentation ». In: *CVPR* (2022) (cit. on p. 41).
- [303] Carlo Tomasi and Takeo Kanade. « Shape and motion from image streams under orthography: A factorization method ». In: *IJCV* (1992) (cit. on pp. 9, 19, 43, 109).
- [304] Hugo Touvron et al. « LLaMA: Open and efficient foundation language models ». In: *arXiv preprint arXiv:2302.13971* (2023) (cit. on pp. 8, 14).
- [305] Ministère de la Transition Ecologique et de la Cohésion des Territoires. *L’empreinte carbone de la france de 1995 à 2021*. 2022. URL: <https://www.statistiques.developpement-durable.gouv.fr/lempreinte-carbone-de-la-france-de-1995-2021> (cit. on p. 13).
- [306] *Trimble Tx8 3D laser scanner*. Accessed: 2023-11-15. URL: <https://geospatial.trimble.com/products-and-solutions/trimble-tx8> (cit. on pp. 20, 107).
- [307] Sik-Ho Tsang. *Review: Xception — with depthwise separable convolution, better than Inception-V3 (image classification)*. 2018. URL: <https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568> (cit. on p. 37).
- [308] Wei-Chih Tu et al. « Learning superpixels with segmentation-aware affinity loss ». In: *CVPR* (2018) (cit. on p. 39).
- [309] Robert J Valkenburg and Alan M McIvor. « Accurate 3D measurement using a structured light system ». In: *Image and Vision Computing* (1998) (cit. on p. 108).
- [310] Michael Van den Bergh, Daniel Carton, and Luc Van Gool. « Depth seeds: Recovering incomplete depth data using superpixels ». In: *WACV* (2013) (cit. on p. 39).
- [311] Nina Varney, Vijayan K Asari, and Quinn Graehling. « DALES: A large-scale aerial LiDAR data set for semantic segmentation ». In: *CVPR Workshops* (2020) (cit. on pp. 49, 61, 62, 77, 79, 89).
- [312] J Irving Vasquez-Gomez, L Enrique Sucar, and Rafael Murrieta-Cid. « View/state planning for three-dimensional object reconstruction under uncertainty ». In: *Autonomous Robots* (2017) (cit. on p. 111).
- [313] Ashish Vaswani et al. « Attention is all you need ». In: *NeurIPS* (2017) (cit. on pp. 6, 33).
- [314] Petar Veličković et al. « Graph attention networks ». In: *ICLR* (2018) (cit. on p. 54).

- [315] Pierre-Francois Verhulst. « Mathematical researches into the law of population growth increase ». In: *Nouveaux Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Bruxelles* (1845) (cit. on p. 4).
- [316] Juho-Pekka Virtanen et al. « Interactive dense point clouds in a game engine ». In: *ISPRS Journal of Photogrammetry and Remote Sensing* (2020) (cit. on p. 108).
- [317] Thang Vu et al. « SoftGroup for 3D instance segmentation on point clouds ». In: *CVPR* (2022) (cit. on pp. 80, 95).
- [318] Michael Waechter, Nils Moehrle, and Michael Goesele. « Let there be color! large-scale texturing of 3D reconstructions ». In: *ECCV* (2014) (cit. on pp. 31, 108, 111).
- [319] Johanna Wald et al. « Learning 3D semantic scene graphs from 3D indoor reconstructions ». In: *CVPR* (2020) (cit. on p. 93).
- [320] Cheng Wang et al. « Urban 3D modeling with mobile laser scanning: A review ». In: *Virtual Reality & Intelligent Hardware* (2020) (cit. on p. 77).
- [321] Chu Wang, Marcello Pelillo, and Kaleem Siddiqi. « Dominant set clustering and pooling for multi-view 3D object recognition ». In: *BMVC* (2019) (cit. on p. 111).
- [322] Peng-Shuai Wang. « OctFormer: Octree-based transformers for 3D point clouds ». In: *SIGGRAPH* (2023) (cit. on p. 93).
- [323] Qi Wang et al. « Window normalization: Enhancing point cloud understanding by unifying inconsistent point densities ». In: (2022). DOI: [10.48550/arXiv.2212.02287](https://doi.org/10.48550/arXiv.2212.02287) (cit. on pp. 63, 92, 93).
- [324] Ruisheng Wang, Jiju Peethambaran, and Dong Chen. « LiDAR point clouds to 3D urban models : A review ». In: *Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2018) (cit. on pp. 2, 11).
- [325] Sinong Wang et al. « Linformer: Self-attention with linear complexity ». In: *arXiv preprint arXiv:2006.04768* (2020) (cit. on p. 38).
- [326] Wenhui Wang et al. « Image as a foreign language: Beit pretraining for vision and vision-language tasks ». In: *CVPR* (2023) (cit. on p. 8).
- [327] Yue Wang et al. « Dynamic graph CNN for learning on point clouds ». In: *ACM Transactions on Graphics* (2019) (cit. on p. 33).
- [328] Aloysius Wehr and Uwe Lohr. « Airborne laser scanning—an introduction and overview ». In: *ISPRS Journal of Photogrammetry and Remote Sensing* (1999) (cit. on p. 9).
- [329] Xin Wei, Ruixuan Yu, and Jian Sun. « View-GCN: View-based graph convolutional network for 3D shape analysis ». In: *CVPR* (2020) (cit. on pp. 108, 111).
- [330] Kristoffer Wickstrøm et al. « Mixing up contrastive learning: Self-supervised representation learning for time series ». In: *Pattern Recognition Letters* (2022) (cit. on p. 8).
- [331] Iain H Woodhouse et al. « A multispectral canopy LiDAR demonstrator project ». In: *Geoscience and Remote Sensing Letters* (2011) (cit. on p. 107).

- [332] Carole-Jean Wu et al. « Sustainable AI: Environmental implications, challenges and opportunities ». In: *Proceedings of Machine Learning and Systems* (2022) (cit. on p. 13).
- [333] Shun-Cheng Wu et al. « SceneGraphFusion: Incremental 3D scene graph prediction from RGB-D sequences ». In: *CVPR* (2021) (cit. on pp. 80, 93).
- [334] Xiaoyang Wu et al. « Point Transformer V2: Grouped vector attention and partition-based pooling ». In: *NeurIPS* (2022) (cit. on p. 93).
- [335] Zhirong Wu et al. « 3D ShapeNets: A deep representation for volumetric shapes ». In: *CVPR* (2015) (cit. on p. 111).
- [336] Michael A Wulder et al. « LiDAR sampling for large-area forest characterization: A review ». In: *Remote sensing of environment* (2012) (cit. on p. 11).
- [337] Binbin Xiang et al. « A review of panoptic segmentation for mobile mapping point clouds ». In: *arXiv preprint arXiv:2304.13980* (2023) (cit. on pp. 77, 80, 89, 92, 97).
- [338] Saining Xie et al. « PointContrast: Unsupervised pre-training for 3D point cloud understanding ». In: *ECCV* (2020) (cit. on pp. 9, 13).
- [339] Dandan Xu et al. « LiDAR applications to estimate forest biomass at individual tree scale: Opportunities, challenges and future perspectives ». In: *Forests* (2021) (cit. on pp. 2, 77).
- [340] Yongchao Xu, Thierry Géraud, and Laurent Najman. « Hierarchical image simplification and segmentation based on mumford–shah-salient level line selection ». In: *Pattern Recognition Letters* (2016) (cit. on p. 41).
- [341] Fan Xue et al. « From LiDAR point cloud towards digital twin city: Clustering city objects based on gestalt principles ». In: *ISPRS Journal of Photogrammetry and Remote Sensing* (2020) (cit. on p. 77).
- [342] Junjie Yan et al. « Object detection by labeling superpixels ». In: *CVPR* (2015) (cit. on p. 39).
- [343] Wai Yeung Yan, Ahmed Shaker, and Nagwa El-Ashmawy. « Urban land cover classification using airborne LiDAR data: A review ». In: *Remote Sensing of Environment* (2015) (cit. on pp. 2, 77).
- [344] Bo Yang et al. « Learning object bounding boxes for 3D instance segmentation on point clouds ». In: *NeurIPS* (2019) (cit. on pp. 80, 111).
- [345] Xiangli Yang et al. « A survey on deep semi-supervised learning ». In: *Transactions on Knowledge and Data Engineering* (2022) (cit. on p. 36).
- [346] Haoxuan You et al. « PVRNet: Point-view relation neural network for 3D shape recognition ». In: *AAAI* (2019) (cit. on p. 111).
- [347] Fisher Yu and Vladlen Koltun. « Multi-scale context aggregation by dilated convolutions ». In: *arXiv preprint arXiv:1511.07122* (2015) (cit. on p. 5).
- [348] Manzhu Yu, Chaowei Yang, and Yun Li. « Big data in natural disaster management: A review ». In: *Geosciences* (2018) (cit. on p. 11).

- [349] Tan Yu, Jingjing Meng, and Junsong Yuan. « Multi-view harmonized bilinear network for 3D object recognition ». In: *CVPR* (2018) (cit. on p. 111).
- [350] Manzil Zaheer et al. « Deep sets ». In: *NeurIPS* (2017) (cit. on pp. 32, 118).
- [351] Georgios Zamanakos et al. « A comprehensive survey of LiDAR-Based 3D Object detection methods with deep learning for autonomous driving ». In: *Computers & Graphics* (2021) (cit. on p. 80).
- [352] Cheng Zhang et al. « PVT: Point-voxel transformer for point cloud learning ». In: *International Journal of Intelligent Systems* (2022) (cit. on p. 32).
- [353] Hongyi Zhang et al. « mixup: Beyond empirical risk minimization ». In: *arXiv preprint arXiv:1710.09412* (2017) (cit. on p. 5).
- [354] Li Zhang, Brian Curless, and Steven M Seitz. « Rapid shape acquisition using color structured light and multi-pass dynamic programming ». In: *3D Data Processing Visualization and Transmission* (2002) (cit. on p. 9).
- [355] Yang Zhang et al. « PolarNet: An improved grid representation for online LiDAR point clouds semantic segmentation ». In: *CVPR* (2020) (cit. on p. 80).
- [356] Zaiwei Zhang et al. « Self-supervised pretraining of 3D features on any point-cloud ». In: *ICCV* (2021) (cit. on pp. 9, 13).
- [357] Zizhao Zhang et al. « Nested hierarchical transformer: Towards accurate, data-efficient and interpretable visual understanding ». In: *AAAI* (2022) (cit. on p. 41).
- [358] Hengshuang Zhao et al. « Pyramid scene parsing network ». In: *CVPR* (2017) (cit. on p. 201).
- [359] Hengshuang Zhao et al. « Point Transformer ». In: *ICCV* (2021) (cit. on pp. 9, 34, 38, 47, 63, 92–95, 186).
- [360] Lin Zhao et al. « When brain-inspired AI meets AGI ». In: *Meta-Radiology* (2023) (cit. on p. 4).
- [361] Yiming Zhao, Xiao Zhang, and Xinming Huang. « A technical survey and evaluation of traditional point cloud clustering methods for LiDAR panoptic segmentation ». In: *ICCV Workshop* (2021) (cit. on p. 80).
- [362] Bolei Zhou et al. « Scene parsing through ade20k dataset ». In: *CVPR* (2017) (cit. on p. 201).
- [363] Dingfu Zhou et al. « Joint 3D instance segmentation and object detection for autonomous driving ». In: *CVPR* (2020) (cit. on p. 80).
- [364] Zhi-Hua Zhou. « A brief introduction to weakly supervised learning ». In: *National Science Review* (2018) (cit. on p. 36).
- [365] Zixiang Zhou, Yang Zhang, and Hassan Foroosh. « Panoptic-PolarNet: Proposal-free LiDAR point cloud panoptic segmentation ». In: *CVPR* (2021) (cit. on pp. 77, 80).
- [366] Xinge Zhu et al. « Cylindrical and asymmetrical 3D convolution networks for LiDAR segmentation ». In: *CVPR* (2021) (cit. on p. 80).

List of Figures

1.1	Large-Scale 3D Scene Understanding	2
1.2	The Rising Popularity Deep Learning	5
1.3	Learning Image Rotations	8
1.4	3D Point Cloud Acquisition Techniques	10
1.5	Domains of Application for 3D Point Cloud Processing	12
1.6	Size of Language Models Across Time	15
1.7	Challenges of 3D Point Cloud Analysis	18
1.8	Hierarchical Representation	19
1.9	Multimodal Fusion	21
2.1	3D Deep Learning Methods	31
2.2	Pruning	35
2.3	Depth-Separable Convolution	37
2.4	SLIC Superpixel Partition	40
2.5	Superpoint Graph	42
3.1	Model Size vs. Performance	47
3.2	Superpoint Transformer	50
3.3	Superpoint Transformer	53
3.4	Point Geometric Features	57
3.5	Qualitative Results	72
3.6	Training Speed	73
3.7	Partition Purity	74
4.1	Large-Scale Panoptic Segmentation	78
4.2	SuperCluster	81
4.3	Superpoint Object Agreement	88
4.4	Qualitative Results	96
4.5	Large-Scale Inference on S3DIS	99
4.6	Large-Scale Inference on ScanNet	99

4.7	Large-Scale Inference on DALES	100
4.8	Large-Scale Inference on KITTI-360	101
5.1	Combining 2D and 3D Information	107
5.2	Mapping Computation	114
5.3	Multi-View Information	119
5.4	Bimodal 2D/3D Architecture	121
5.5	Dynamic Batching	122
5.6	Datasets	124
5.7	Qualitative Illustration	127
5.8	Influence of Viewing Descriptors	131
5.9	Depth-Based Mapping Computation	133
A-1	Colormaps	176
B-1	Interactive Visualization	177
C-1	Interactive Visualization	190
D-1	Interactive Visualization	203
E-1	Analyse de Scènes 3D à Grande Échelle	207
E-2	Méthodes d'apprentissage profond 3D	212
E-3	Taille du modèle vs Performance	215
E-4	Segmentation Panoptique à Grande Échelle	216
E-5	Fusion Multimodale et Multi-vues	218

List of Tables

3.1	Partition Configuration	62
3.2	Performance Evaluation	63
3.3	Efficiency Analysis	65
3.4	Ablation Study	66
3.5	Impact of Model Scaling	68
3.6	Impact of Query-Key Dimension	68
3.7	Impact of Heads Count	68
3.8	Ablation on Supervision	69
4.1	Graph Clustering Parameters	91
4.2	S3DIS Area 5	92
4.3	S3DIS 6-Fold	93
4.4	ScanNetv2 Val	93
4.5	KITTI-360	94
4.6	DALES	94
4.7	Runtime	95
4.8	Ablation Study	98
5.1	Quantitative Evaluation	125
5.2	Ablation Study	129
5.3	Usage and Sensitivity of Viewing Conditions	136
5.4	Supplementary Ablation Study	136
5.5	Effect of Maximum Depth	136
5.6	Impact of the Number of Images	137
B-1	Ablation on Handcrafted Features	179
B-2	Model Configuration	184
B-3	S3DIS Class-wise Performance	186
B-4	KITTI-360 Class-wise Performance	187
B-5	Class-wise Performance	188

C-1	S3DIS Class-wise Performance	192
C-2	S3DIS Class-wise Performance Without “Stuff”	193
C-3	ScanNetv2 Val. Class-wise Performance	194
C-4	KITTI-360 Val. Class-wise Performance	195
C-5	DALES Class-wise Performance	196
D-1	S3DIS Class-wise Performance	204
D-2	ScanNet Val. Class-wise Performance	205
D-3	KITTI-360 Class-wise Performance	206

Appendices

Appendix A

Additional Notes

In this appendix, we detail the colormaps used for each dataset in Section [A-1](#).

A-1 Dataset Colormaps

We use the same colormaps for each dataset across all chapters. These colormaps are detailed in Table [A-1](#).





























































S3DIS				
 ceiling	 floor	 wall	 beam	 column
 window	 door	 chair	 table	 bookcase
 sofa	 board	 clutter	 unlabeled	
ScanNet				
 wall	 floor	 cabinet	 bed	 chair
 sofa	 table	 door	 window	 bookshelf
 picture	 counter	 desk	 curtain	 refrigerator
 shower	 toilet	 sink	 bathtub	 otherfurniture
 ignored				
KITTI-360				
 road	 sidewalk	 building	 wall	 fence
 pole	 traffic light	 traffic sign	 vegetation	 terrain
 person	 car	 truck	 motorcycle	 bicycle
 ignored				
DALES				
 ground	 vegetation	 car	 truck	 power line
 fence	 pole	 building	 unknown	

Figure A-1 – **Colormaps.** Throughout all visualizations in this work, we use these colormaps to represent the semantic of each point.

Appendix B

Additional Results on Efficient and Scalable 3D Semantic Segmentation

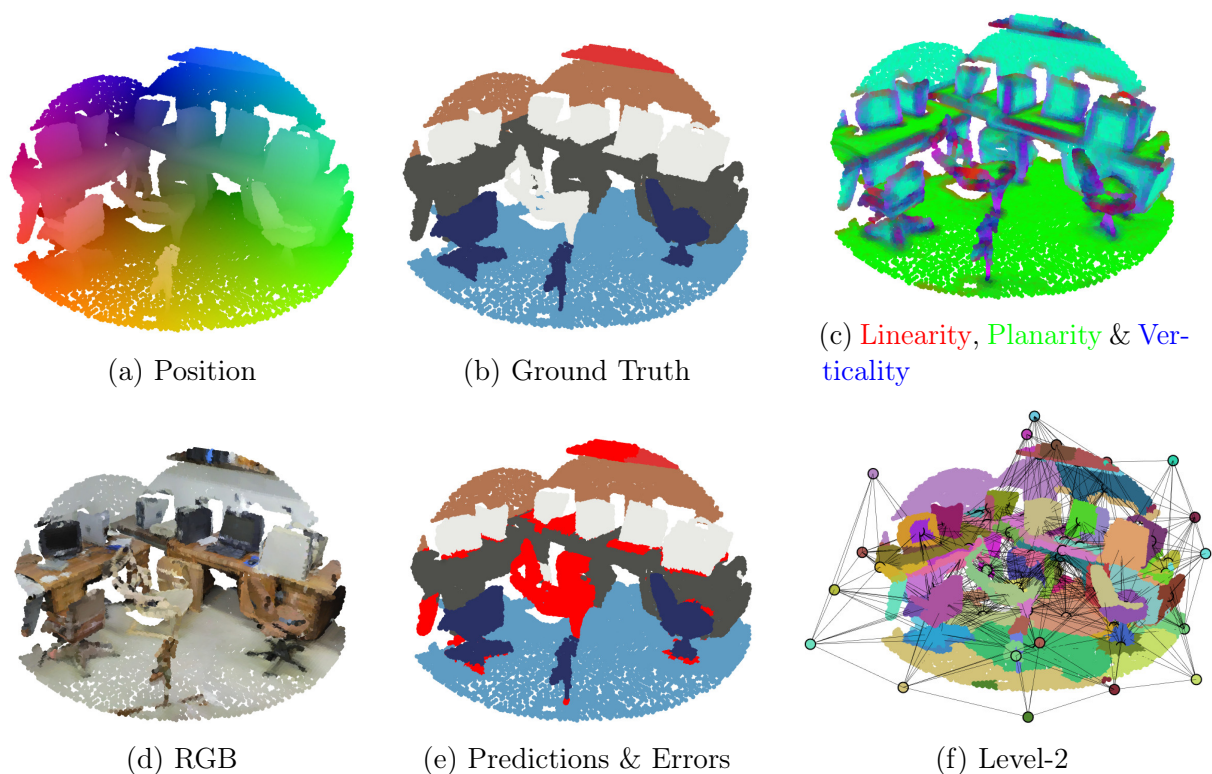


Figure B-1 – **Interactive Visualization.** Our interactive viewing tool allows for the manipulation and visualization of sample point clouds colored according to their position (a), semantic labels (b), selected geometric features (c), radiometry (d), and to visualize our network’s prediction (e) and partitions (f).

In this document, we introduce our interactive visualization tool (Section B-1), share our source code (Section B-2), provide an analysis (Section B-3) of all handcrafted features used by our method, detail the partition process (Section B-4), and provide guidelines on how to choose the partition’s hyperparameters (Section B-5). Finally, we clarify our architecture parameters (Section B-6) and detail the class-wise performance of our approach on each dataset (Section B-7).

B-1 Interactive Visualization

We release for this project an interactive plotly visualization tool that produces HTML files compatible with any browser. As shown in Figure B-1, we can visualize samples from S3DIS, KITTI-360, and DALES with different point attributes and from any angle. These visualizations were instrumental in designing and validating our model, and we hope that they will facilitate the reader’s understanding as well.

B-2 Source Code

We make our source code publicly available at https://github.com/drprojects/superpoint_transformer.

The code provides all necessary instructions for installing and navigating the project, simple commands to reproduce our main results on all datasets, ready-to-use pretrained models, and ready-to-use notebooks.

Our method is developed in PyTorch and relies on PyTorch Geometric, PyTorch Lightning, and Hydra.

Table B-1 – **Ablation on Handcrafted Features.** Impact of handcrafted features on the mIoU for all tested datasets.

Experiment	S3DIS 6-Fold	KITTI 360 Val	DALES
Best Model	76.0	63.5	79.6
<i>a) Point Features</i>			
No radiometric feat.	-2.7	-4.0	-1.2
No geometric feat.	-0.7	-4.1	-1.4
<i>b) Adjacency Features</i>			
No interface feat.	-0.2	-0.6	-0.7
No ratio feat.	-1.1	-2.2	-0.4
No pose feat.	-5.5	-1.2	-0.8
<i>c) Room Features</i>			
Room-level samples	-3.8	-	-
Normalized Room pos.	-0.7	-	-

B-3 Influence of Handcrafted Features

In Table B-1, we quantify the impact of the handcrafted features detailed in Section 3.2.4 on performance. To this end, we retrain SPT without each feature group and evaluate the prediction on S3DIS Area 5.

a) Point Features. Our experiments show that removing radiometric features has a strong impact on performance, with a drop of 2.7 to 4.0 mIoU. In contrast, removing geometric features results in a performance drop of 0.7 on S3DIS, but 4.1 on KITTI-360.

We observe that both outdoor datasets strongly benefit from local geometric features, which we hypothesize is due to their lower resolution and noise level. These results indicate that radiometric features play an important role for all datasets and that geometric features may facilitate learning on noisy or

subsampled datasets.

b) Adjacency Features. The analysis of the impact of adjacency features on our model’s performance indicates that they play a crucial role in leveraging contextual information from superpoints: removing all adjacency features leads to a significant drop of 3.0 to 6.3 mIoU points on the datasets, as shown in Section 3.3.3. Among the different types of adjacency features, pose features appear particularly useful in characterizing the adjacency relationships between superpoints of S3DIS, while interface features have a smaller impact. These results suggest that the relative pose of objects in the scene may have more influence on the 3D semantic analysis performed by our model than the precise characterization of their interface. On the other hand, interface and ratio features seem to have more impact on outdoor datasets, while the pose information seems to be less informative in the semantic understanding of the scene.

c) S3DIS Room Partition. The S3DIS dataset is divided into individual rooms aligned along the x and y axes. This setup simplifies the identification of classes such as walls, doors, or windows as they are consistently located at the edge of the room samples. Some methods also add normalized room coordinates to each points. However, we argue that this partition may not generalize well to other environments, such as open offices, industrial facilities, or mobile mapping acquisitions, which cannot naturally be split into rooms.

To address this limitation, we use the absolute room positions to reconstruct the entire floor of each S3DIS area [301, 46]. This enables our model to consider large multi-room samples, resulting in a performance increase of 3.8 points. This highlights the advantage of capturing long-range contextual information. Additionally, we remark that SPT performs better without using room-normalized coordinates, which may lead to overfitting and poor

performance on layouts that deviate from the room-based structure of the S3DIS dataset such as large amphitheaters.

B-4 Details on Hierarchical Partitions

We present here a more detailed explanation of the hierarchical partition process. We define for each point c of \mathcal{C} a feature f_c of dimension D , and $G := (\mathcal{C}, \mathcal{E}, w)$ is the k-nn adjacency between the points, with $w \in \mathbb{R}_+^{\mathcal{E}}$ a nonnegative proximity value. Our goal is to compute a hierarchical multilevel partition of the point cloud into superpoints homogeneous with respect to f at increasing coarseness.

Piecewise Constant Approximation on a Graph. We first explain how to compute a single-level partition of the point cloud. We consider the pointwise features f_c as a D -dimensional signal $f \in \mathbb{R}^{D \times |\mathcal{C}|}$ defined on the nodes of the weighted graph $G := (\mathcal{C}, \mathcal{E}, w)$. We first define an energy $\mathcal{J}(e; f, \mathcal{G}, \lambda)$ measuring the fidelity between a vertex-valued signal $e \in \mathbb{R}^{D \times |\mathcal{C}|}$ and the length of its contours, defined as the weight of the cut between its constant components [178]:

$$\mathcal{J}(e; f, \mathcal{G}, \lambda) := \|e - f\|^2 + \lambda \sum_{(u,v) \in \mathcal{E}} w_{u,v} [e_u \neq e_v] , \quad (\text{B-1})$$

with $\lambda \in \mathbb{R}_+$ a regularization strength and $[a \neq b]$ the function equals to 0 if $a = b$ and 1 otherwise. Minimizers of \mathcal{J} are approximations of f that are piecewise constant with respect to a partition with simple contours in \mathcal{G} .

We can characterize such signal $e \in \mathbb{R}^{D \times |\mathcal{C}|}$ by the coarsest partition \mathcal{P}^e of \mathcal{P} and its associated variable $f^e \in \mathbb{R}^{D \times |\mathcal{P}^e|}$ such that e is constant within each segment p of \mathcal{P}^e with value f_p^e . The partition \mathcal{P}^e also induces a graph $\hat{\mathcal{G}}^e := (\mathcal{P}^e, \mathcal{E}^e, w^e)$ with \mathcal{E}^e linking the component of \mathcal{P}^e adjacent in \mathcal{G} and w^e

the weight of the cut between adjacent elements of \mathcal{P}^e :

$$\mathcal{E}^e := \{(U, V) \mid U, V \in \mathcal{P}^e, (U \times V) \cap \mathcal{E} \neq \emptyset\} \quad (\text{B-2})$$

$$\text{For } (U, V) \in \mathcal{E}^e, w_{U,V}^e := \sum_{(u,v) \in U \times V \cap \mathcal{E}} w_{u,v} \quad (\text{B-3})$$

We denote by $\text{partition}(e)$ the function mapping e to these uniquely defined variables:

$$f^e, \mathcal{P}^e, \hat{\mathcal{G}}^e := \text{partition}(e) . \quad (\text{B-4})$$

Point Cloud Hierarchical Partition. A set of partitions $\mathcal{P} := [\mathcal{P}_0, \dots, \mathcal{P}_I]$ defines a hierarchical partition of \mathcal{C} with I levels if $\mathcal{P}_0 = \mathcal{C}$ and \mathcal{P}_{i+1} is a partition of \mathcal{P}_i for $i \in [0, I - 1]$. We propose to use the formulations above to define a hierarchical partition of the point cloud \mathcal{C} characterized by a list $\lambda_1, \dots, \lambda_I$ of nonnegative regularization strengths defining the coarseness of the successive partitions. In particular, We chose λ_1 such that $|\mathcal{P}_1|/|\mathcal{P}_0| \sim 30$ in our experiments.

We first define $\hat{\mathcal{G}}_0$ as the point-level adjacency graph $\hat{\mathcal{G}}$ and f_0 as f . We can now define the levels of a hierarchical partition \mathcal{P}_i for $i \in [1, I]$:

$$f_i, \mathcal{P}_i, \hat{\mathcal{G}}_i := \text{partition}\left(\arg \min_{e \in \mathbb{R}^{D \times |\mathcal{P}_{i-1}|}} \mathcal{J}\left(e; f_{i-1}, \hat{\mathcal{G}}_{i-1}, \lambda_{i-1}\right)\right). \quad (\text{B-5})$$

Given that the optimization problems defined in Equation B-5 for $i > 1$ operate on the component graphs $\hat{\mathcal{G}}_i$, which are smaller than $\hat{\mathcal{G}}_0$, the first partition is the most demanding in terms of computation.

Note that we used the hat notation $\hat{\mathcal{G}}_i$, because these graphs are only used for computing the hierarchical partitions \mathcal{P}_i , and should be distinguished from the superpoint graphs \mathcal{G}_i on which is based our self-attention mechanism, constructed from \mathcal{P}_i as explained in Section 3.2.5.

B-5 Parameterizing the Partition

We define \mathcal{G} as the $k = 10$ -nearest neighbor adjacency graph and set all edge weights w to 1. The point features f_p whose piecewise constant approximation yields the partition are of three types: geometric, radiometric, and spatial.

Geometric features ensure that the superpoints are geometrically homogeneous and with simple shapes. We use the normalized dimensionality-based method described in Section 3.2.4. Radiometric features encourage the border of superpoints to follow the color contrast of the scene and are either RGB or intensity values; they must be normalized to fall in the $[0,1]$ range. Lastly, we can add to each point their spatial coordinates with a normalization factor μ in m^{-1} to limit the size of the superpoints. We recommend setting μ as the inverse of the maximum radius expected for a superpoint: the largest sought object (facade, wall, roof) or an application-dependent constraint.

The coarseness of the partitions depends on the regularization strength λ as defined in Section 3.2.1. Finer partitions should generally lead to better results but to an increase in training time and memory requirement. We chose a ratio $|\mathcal{P}_0| / |\mathcal{P}_1| \sim 30$ across all datasets as it proved to be a good compromise between efficiency and precision. Depending on the desired trade-off, different ratios can be chosen by trying other values of λ .

B-6 Implementation Details

We provide the exact parameterization of the SPT architecture used for our experiments. All MLPs in the architecture use LeakyReLU activations and GraphNorm [39] normalization. For simplicity, we represent an MLP by the list of its layer widths: `[in_channels, hidden_channels, out_channels]`.

Table B-2 – **Model Configuration.** We provide the detailed architecture of the SPT-X architecture. In this work, we use $X = 64$ and $X = 128$.

Parameter	Value
<i>Handcrafted features</i>	
$D_{\text{point}}^{\text{hf}}$	$D_{\text{point}}^{\text{radio}} + D_{\text{point}}^{\text{geof}}$
$D_{\text{adj}}^{\text{hf}}$	18
<i>Embeddings sizes</i>	
D_{point}	128
D_{adj}	32
<i>Transformer blocks</i>	
D_{val}	\mathbf{X}
D_{key}	4
# blocks encoder	3
# blocks decoder	1
# heads	16
<i>MLPs</i>	
ϕ_{adj}^i	$[D_{\text{adj}}^{\text{hf}}, D_{\text{adj}}, D_{\text{adj}}, 3D_{\text{adj}}]$
ϕ_{enc}^0	$[D_{\text{point}}^{\text{hf}} + D_{\text{point}}^{\text{pos}}, 32, 64, D_{\text{point}}]$
ϕ_{enc}^1	$[D_{\text{point}} + D_{\text{point}}^{\text{pos}}, D_{\text{val}}, D_{\text{val}}]$
ϕ_{enc}^2	$[D_{\text{val}} + D_{\text{point}}^{\text{pos}}, D_{\text{val}}, D_{\text{val}}]$
ϕ_{dec}^1	$[D_{\text{val}} + D_{\text{val}} + D_{\text{point}}^{\text{pos}}, D_{\text{val}}, D_{\text{val}}]$

Point Input Features. We refer here to the dimension of point positions, radiometry, and geometric features as $D_{\text{point}}^{\text{pos}} = 3$, $D_{\text{point}}^{\text{radio}}$, and $D_{\text{point}}^{\text{geof}} = 4$ respectively. As seen in Section 3.2.4, S3DIS and KITTI-360 use $D_{\text{point}}^{\text{radio}} = 3$, while DALES uses $D_{\text{point}}^{\text{radio}} = 1$.

Model Architecture. The exact architecture SPT-64 used for S3DIS and DALES is detailed in Table B-2. The other models evaluated are SPT-16, SPT-32, SPT-128 (used for KITTI-360), and SPT-256, which use the same parameters except for D_{val} .

SPT-nano. For SPT-nano, we use and $D_{\text{val}} = 16$, $D_{\text{adj}} = 16$, and $D_{\text{key}} = 2$. As SPT-nano does not compute point embedding, it does not use ϕ^0 , and we set up ϕ_{enc}^1 as $[D_{\text{point}}^{\text{hf}} + D_{\text{point}}^{\text{pos}}, D_{\text{val}}, D_{\text{val}}]$.

B-7 Detailed Results

We report in Table B-3, Table B-4, and Table B-5 the class-wise performance across all datasets for SPT and other methods for which this information was available. As previously stated, SPT performs close to state-of-the-art methods on all datasets, while being significantly smaller and faster to train. By design, superpoint-based methods can capture long-range interactions and their predictions are more spatially regular than point-based approaches. This may explain the performance of SPT on S3DIS, which encompasses large, geometrically homogeneous objects or whose identification requires long-range context understanding, such as ceiling, floor, columns, and windows. For all datasets, results show that some progress could be made in analyzing smaller objects with intricate geometries. This suggests that a more powerful point-level encoding may be beneficial.

Table B-3 – S3DIS Class-wise Performance. Class-wise mIoU on S3DIS for our Superpoint Transformer.

Method	S3DIS Area 5													
	mIoU	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter
PointNet [252]	41.1	88.8	97.3	69.8	0.1	3.9	46.3	10.8	52.6	58.9	40.3	5.9	26.4	33.2
SPG [180]	58.4	89.4	96.9	78.1	0.0	42.8	48.9	61.6	84.7	75.4	69.8	52.6	2.1	52.2
MinkowskiNet [58]	65.4	91.8	98.7	86.2	0.0	34.1	48.9	62.4	81.6	89.8	47.2	74.9	74.4	58.6
SPG + SSP [177]	61.7	91.9	96.7	80.8	0.0	28.8	60.3	57.2	85.5	76.4	70.5	49.1	51.6	53.3
KPConv [301]	67.1	92.8	97.3	82.4	0.0	23.9	58.0	69.0	91.0	81.5	75.3	75.4	66.7	58.9
PointTrans [359]	70.4	94.0	98.5	86.3	0.0	38.0	63.4	74.3	89.1	82.4	74.3	80.2	76.0	59.3
DeepViewAgg [269]	67.2	87.2	97.3	84.3	0.0	23.4	67.6	72.6	87.8	81.0	76.4	54.9	82.4	58.7
Stratified T. [175]	72.0	96.2	98.7	85.6	0.0	46.1	60.0	76.8	92.6	84.5	77.8	75.2	78.1	64.0
SPT	68.9	92.6	97.7	83.5	0.2	42.0	60.6	67.1	88.8	81.0	73.2	86.0	63.1	60.0
SPT-nano	64.9	92.4	97.1	81.6	0.0	38.2	56.4	58.6	86.3	77.3	69.6	82.5	50.5	53.4
S3DIS 6-FOLD														
PointNet [252]	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	42.0	54.1	38.2	9.6	29.4	35.2
SPG [180]	62.1	89.9	95.1	76.4	62.8	47.1	55.3	68.4	73.5	69.2	63.2	45.9	8.7	52.9
ConvPoint [24]	68.2	95.0	97.3	81.7	47.1	34.6	63.2	73.2	75.3	71.8	64.9	59.2	57.6	65.0
MinkowskiNet [58, 269]	69.5	91.2	90.6	83.0	59.8	52.3	63.2	75.7	63.2	64.0	69.0	72.1	60.1	59.2
SPG + SSP [177]	68.4	91.7	95.5	80.8	62.2	54.9	58.8	68.4	78.4	69.2	64.3	52.0	54.2	59.2
KPConv [301]	70.6	93.6	92.4	83.1	63.9	54.3	66.1	76.6	57.8	64.0	69.3	74.9	61.3	60.3
DeepViewAgg [269]	74.7	90.0	96.1	85.1	66.9	56.3	71.9	78.9	79.7	73.9	69.4	61.1	75.0	65.9
SPT	76.0	93.9	96.3	84.3	71.4	61.3	70.1	78.2	84.6	74.1	67.8	77.1	63.6	65.0
SPT-nano	70.8	93.1	96.0	80.9	68.4	54.0	62.2	71.3	76.3	70.8	63.3	74.3	51.9	57.6

Table B-4 – KITTI-360 Class-wise Performance. Class-wise mIoU on KITTI-360 for our Superpoint Transformer.

Method	mIoU	KITTI-360 Val														
		road	sidewalk	building	wall	fence	pole	traffic lig.	traffic sig.	vegetation	terrain	person	car	truck	motorcycle	bicycle
MinkowskiNet [58, 269]	54.2	90.6	74.4	84.5	45.3	42.9	52.7	0.5	38.6	87.6	70.3	26.9	87.3	66.0	28.2	17.2
DeepViewAgg [269]	57.8	93.5	77.5	89.3	53.5	47.1	55.6	18.0	44.5	91.8	71.8	40.2	87.8	30.8	39.6	26.1
SPT	63.5	93.3	79.3	90.8	56.2	45.7	52.8	20.4	51.4	89.8	73.6	61.6	95.1	79.0	53.1	10.9
SPT-nano	57.2	91.7	74.7	87.8	49.3	38.8	49.0	12.2	39.2	88.0	69.5	39.9	94.2	80.1	33.7	10.4

Table B-5 – DALES Class-wise Performance. Class-wise mIoU on DALES for our Superpoint Transformer.

Method	DALES									
	mIoU	ground	vegetation	car	truck	power line	fence	pole	building	
PointNet++ [253]	68.3	94.1	91.2	75.4	30.3	79.9	46.2	40.0	89.1	
ConvPoint [24]	67.4	96.9	91.9	75.5	21.7	86.7	29.6	40.3	96.3	
SPG [180]	60.6	94.7	87.9	62.9	18.7	65.2	33.6	28.5	93.4	
PointCNN [193]	58.4	97.5	91.7	40.6	40.8	26.7	52.6	57.6	95.7	
KPConv [301]	81.1	97.1	94.1	85.3	41.9	95.5	63.5	75.0	96.6	
SPT	79.6	96.7	93.1	86.1	52.4	94.0	52.7	65.3	96.7	
SPT-nano	75.2	96.5	92.6	78.1	35.8	92.1	50.8	59.9	96.0	

Appendix C

Additional Results on Efficient and Scalable 3D Panoptic Segmentation

In this appendix, we introduce our interactive visualization tool in Section C-1, our source code in Section C-2, and implementation details in Section C-3. Finally, we provide detailed class-wise results for each dataset in Section C-4.

C-1 Interactive Visualization

We propose a Plotly-based interactive visualization tool in the form of HTML pages compatible with any browser and provided in the supplementary materials. As shown in Figure C-1, we can visualize samples from the datasets with different point attributes and from any angle. These visualizations were instrumental in designing and validating our model; we hope they will also facilitate the reader’s understanding. We added one such visualizations for each dataset in the supplementary material.

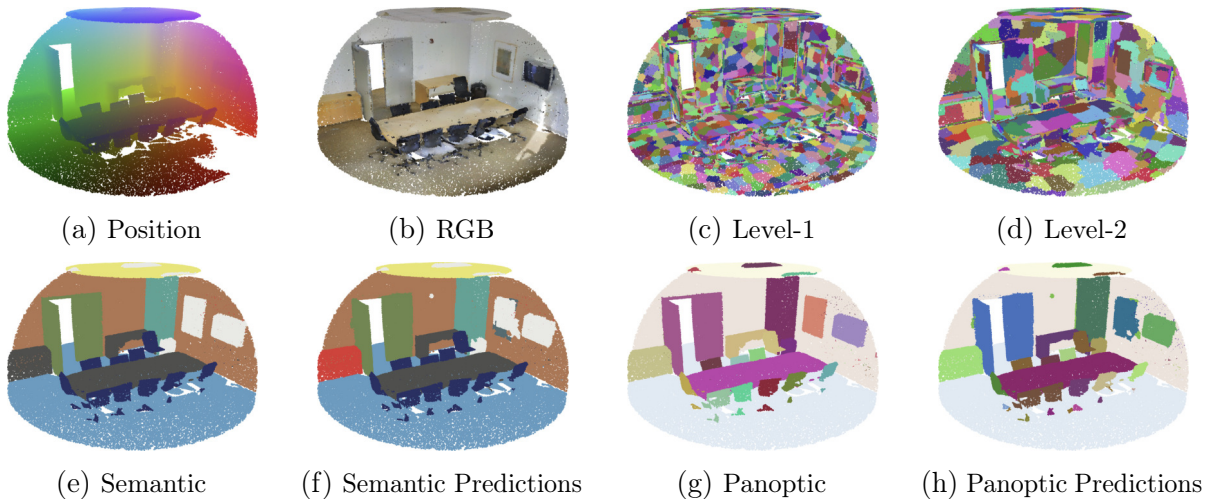


Figure C-1 – **Interactive Visualization.** Our interactive viewing tool allows for the manipulation and visualization of point cloud samples colored according to their position (a), radiometry (b), partition level (c)(d), semantic annotations (e) and predictions (f), and panoptic annotations (g) and predictions (h).

C-2 Source Code

We make our source code available to the reviewers at github.com/drprojects/super_cluster.

The code provides all necessary instructions for installing and navigating the project, simple commands to reproduce our main results on all datasets, and ready-to-use notebooks.

Our method is developed in PyTorch and relies on PyTorch Geometric, PyTorch Lightning, and Hydra.

C-3 Implementation Details

We provide the exact parameterization of the SuperCluster architecture used for our experiments. For simplicity, we represent an MLP by the list of its layer widths: `[in_channels, hidden_channels, out_channels]`.

Backbone. Our backbone model follows the same model configurations as the Superpoint Transformer [267] with minor modifications, described below. We

use SPT-64 for S3DIS and DALES and SPT-128 for KITTI360 and ScanNet.

For all datasets, we reduce the output dimension of the point encoder ϕ_{enc}^0 from 128 to 64 [267]. We find that this does not affect SPT performance while reducing its memory impact.

For ScanNet, we find that using 32 heads instead of 16 and setting $D_{\text{adj}} = 64$ instead of 32 [267] improves performance.

Object Agreement Head. The object agreement prediction head ϕ^{object} is a normalization-free MLP with LeakyReLU activations and layers $[2D, 32, 1]$, where D is the output feature dimension of the backbone (*i.e.* 64 for S3DIS and DALES, and 128 for ScanNet and KITTI-360).

C-4 Detailed Results

We report in Table C-1, Table C-2, Table C-3, Table C-4, and Table C-5 the average and per-class performances of SuperCluster on each dataset.

Table C-1 – **S3DIS Class-wise Performance.** We report the average and per-class panoptic quality (PQ), recognition quality (RQ), segmentation quality (SQ), precision (Prec), and recall (Rec) performance of SuperCluster on S3DIS. We indicate “stuff” classes with †.

		S3DIS Area 5													
Metric	Avg.	ceiling †	floor †	wall †	beam	column	window	door	chair	table	bookcase	sofa	board	clutter	
PQ	58.4	93.8	96.2	84.0	0.0	48.5	64.7	45.3	64.3	40.1	47.7	62.9	70.5	40.6	
RQ	68.4	100.0	100.0	100.0	0.0	61.0	77.2	57.4	76.4	52.6	55.8	78.3	81.1	49.7	
SQ	77.8	93.8	96.2	84.0	0.0	79.5	83.9	78.9	84.1	76.2	85.4	80.4	86.9	81.7	
Prec.	71.4	100.0	100.0	100.0	0.0	64.2	79.6	59.8	75.7	53.3	66.0	75.0	93.8	60.5	
Rec.	66.2	100.0	100.0	100.0	0.0	58.1	75.0	55.1	77.1	52.0	48.4	81.8	71.4	42.2	
S3DIS 6-FOLD															
PQ	62.7	93.8	95.2	84.1	58.9	64.7	70.2	41.1	48.0	45.5	45.8	55.7	64.3	47.2	
RQ	73.2	100.0	100.0	100.0	67.9	78.2	81.8	55.3	57.6	58.6	54.9	65.3	74.7	56.8	
SQ	84.7	93.8	95.2	84.1	86.8	82.7	85.8	74.4	83.4	77.7	83.4	85.3	86.0	83.2	
Prec.	77.8	100.0	100.0	100.0	67.9	80.2	84.7	61.7	69.0	55.9	66.2	74.4	80.0	71.1	
Rec.	69.8	100.0	100.0	100.0	67.9	76.4	79.2	50.1	49.4	61.5	46.9	58.2	70.1	47.2	

Table C-2 – **S3DIS Class-wise Performance Without “Stuff”**. We report the average and per-class panoptic quality (PQ), recognition quality (RQ), segmentation quality (SQ), precision (Prec), and recall (Rec) performance of SuperCluster on S3DIS without “stuff” classes.

		S3DIS Area 5 - no “stuff”												
Metric	Avg.	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter
PQ	50.1	46.9	69.5	39.0	0.0	45.7	68.1	47.9	64.2	41.1	48.6	66.2	74.8	39.1
RQ	60.1	52.0	78.4	49.3	0.0	58.6	80.8	60.2	75.9	53.5	57.6	81.8	85.7	47.8
SQ	76.6	90.3	88.6	79.1	0.0	78.0	84.2	79.6	84.5	76.8	84.3	80.9	87.3	81.8
Prec.	63.6	45.5	70.6	43.7	0.0	62.1	90.5	68.7	76.7	58.5	74.4	81.8	94.3	59.9
Rec.	58.4	60.5	88.2	56.6	0.0	55.4	73.1	53.5	75.2	49.3	47.0	81.8	78.6	39.8

		S3DIS 6-FOLD - no “stuff”												
PQ	55.9	68.6	64.1	40.0	65.6	64.0	70.1	42.7	48.0	48.3	43.7	55.4	69.4	46.7
RQ	66.3	74.8	72.6	50.8	74.3	76.7	81.8	57.1	57.3	62.8	52.5	64.6	80.5	56.0
SQ	83.8	91.8	88.2	78.6	88.3	83.4	85.8	74.7	83.7	76.9	83.2	85.7	86.2	83.4
Prec.	72.8	76.4	69.8	50.2	77.9	78.3	86.7	68.8	70.7	63.9	67.0	72.7	90.8	73.1
Rec.	61.7	73.3	75.7	51.4	71.1	75.2	77.4	48.8	48.2	61.8	43.1	58.2	72.3	45.4

Table C-3 – **ScanNetv2 Val. Class-wise Performance.** We report the average and per-class panoptic quality (PQ), recognition quality (RQ), segmentation quality (SQ), precision (Prec), and recall (Rec) performance of SuperCluster on ScanNet. We indicate ‘stuff’ classes with †.

Metric	Avg	wall †	floor †	cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	refrigerator	shower	toilet	sink	bathub	otherfurniture
PQ	58.7	73.3	91.8	50.5	70.3	61.3	69.0	58.9	42.5	44.5	65.9	27.7	42.9	49.6	40.9	64.1	72.0	88.6	51.0	61.7	46.7
RQ	69.1	88.7	99.3	61.9	77.9	70.7	79.2	68.9	52.9	54.6	72.1	34.7	58.4	62.6	49.6	71.7	84.2	99.2	64.5	75.4	56.0
SQ	84.1	82.6	92.4	81.6	90.2	86.8	87.2	85.5	80.3	81.5	91.4	79.7	73.5	79.1	82.6	89.4	85.5	89.3	79.1	81.8	83.4
Prec.	76.7	93.1	100.0	69.7	81.1	79.0	80.0	68.0	65.9	63.4	75.7	64.9	68.4	60.1	58.0	94.3	82.8	98.3	86.0	76.7	69.5
Rec.	64.3	84.6	98.7	55.7	75.0	63.9	78.4	69.9	44.2	48.0	68.8	23.7	51.0	65.4	43.3	57.9	85.7	100.0	51.6	74.2	46.8

Table C-4 – **KITTI-360 Val. Class-wise Performance.** We report the average and per-class panoptic quality (PQ), recognition quality (RQ), segmentation quality (SQ), precision (Prec), and recall (Rec) performance of SuperCluster on the KITTI-360 Validation set. We indicate “stuff” classes with †.

Metric	Avg.	road †	sidewalk †	building	wall †	fence †	pole †	traffic lig. †	traffic sig. †	vegetation †	terrain †	person †	car	truck †	motorcycle †	bicycle †
PQ	48.3	94.3	75.6	44.8	31.5	20.0	43.4	0.0	30.5	89.5	49.5	19.4	84.2	81.5	55.5	5.4
RQ	58.4	100.0	95.3	53.4	51.4	33.3	65.6	0.0	40.8	100.0	68.5	20.7	89.1	83.3	66.7	7.1
SQ	75.1	94.3	79.3	83.9	61.2	60.0	66.2	0.0	74.6	89.5	72.3	93.6	94.4	97.8	83.3	75.5
Prec.	60.3	100.0	96.2	48.4	51.9	33.3	65.6	0.0	43.5	100.0	76.0	23.1	92.4	90.9	73.3	10.0
Rec.	56.9	100.0	94.4	59.5	50.9	33.3	65.6	0.0	38.5	100.0	62.3	18.8	86.1	76.9	61.1	5.6

Table C-5 – **DALES Class-wise Performance.** We report the average and per-class panoptic quality (PQ), recognition quality (RQ), segmentation quality (SQ), precision (Prec), and recall (Rec) performance of SuperCluster on DALES. We indicate “stuff” classes with †.

Metric	Avg.	ground †	vegetation †	car	truck	power line	fence	pole	building
PQ	61.2	95.6	90.3	70.9	45.0	18.8	23.5	64.3	81.5
RQ	68.6	100.0	99.0	78.4	51.1	23.1	31.3	79.6	86.6
SQ	87.1	95.6	91.2	90.4	88.2	81.3	75.0	80.8	94.1
Prec.	68.5	100.0	99.0	87.3	55.1	16.3	24.2	81.5	84.7
Rec.	71.0	100.0	99.0	71.1	47.5	39.7	44.3	77.8	88.5

Appendix D

Additional Results on Learning From Point Clouds and Images in the Wild

In this appendix, we introduce our interactive visualization tool and code in Section D-1, describe our various multimodal fusion schemes in Section D-2, detail our Dynamic-Size Image-Batching algorithm in Section D-3, and provide a full description of our model implementation in Section D-4. Section D-5 details the camera pose adjustments made to S3DIS for 2D-3D mapping. Finally, we provide detailed class-wise results for each dataset in Section D-6.

D-1 Interactive Visualization and Code

We release our code at

<https://github.com/drprojects/DeepViewAgg>.

The provided code allows for reproduction of our experiments and inference using pretrained models.

Our repository also contains interactive visualizations as HTML files, showing different images and model predictions for spheres sampled in S3DIS, as shown in Figure D-1. This tool makes it easier to see the additional insights

brought by images than the visuals included in the chapter.

D-2 Fusion schemes

We denote by $\{f_i^{2D}\}_{i \in I}$ a set of 2D feature maps of width C associated with the images I , typically obtained with a convolutional neural network. f^{3D} designates the raw feature of the point cloud given by the sensor: position, but also intensity/reflectance if available (not used in this work). We denote by $\mathcal{P}(f^{2D}, P)$ the projection of the learned image features f^{2D} onto the point cloud P by our multi-view aggregation technique. The early and late fusion schemes can be written as follows:

$$y_{\text{early}} = \mathcal{C}^{3D} \circ \mathcal{D}^{3D} \circ \mathcal{E}^{3D} ([f^{3D}, \mathcal{P}(f^{2D}, P)]) \quad (\text{D-1})$$

$$y_{\text{late}} = \mathcal{C}^{3D} ([\mathcal{D}^{3D} \circ \mathcal{E}^{3D} (f^{3D}), \mathcal{P}(f^{2D}, P)]) . \quad (\text{D-2})$$

For the intermediate fusion scheme, the 2D and 3D features are merged directly in the 3D encoder. Our 3D backbone follows a classic U-Net architecture, and its encoder is organized in L levels $\{\mathcal{E}_l^{3D}\}_{l=1}^L$ processing maps of increasingly coarse resolution. Each level $l > 1$ is composed of a downsampling module down_l , typically strided convolutions ($\text{down}_1 = \text{Id}$), and a convolutional module conv_l , typically a sequence of ResNet blocks. The 2D encoder is also composed of L levels $\{\mathcal{E}_l^{2D}\}_{l=1}^L$ corresponding to the different image resolutions. We propose to match the 2D and 3D levels at full resolution (1024×512 and 2 cm for S3DIS), and all subsequent levels after the same number of 2D/3D downsampling. At each level $l = 1 \dots L$, we concatenate the downsampled higher resolution 3D map f_{l-1}^{3D} with the map f_l^{2D} obtained from the images at the matched resolution:

$$f_l^{3D} = \text{conv} ([\text{down} (f_{l-1}^{3D}), \mathcal{P}(f_l^{2D}, P)]) , \quad (\text{D-3})$$

with f_0^{3D} the raw 3D features. The decoder and classifiers follow the same organization than the 3D backbone.

D-3 Dynamic-Size Image-Batching

Algorithm D-3 Dynamic-Size Image-Batching

Input: P_{sample} point cloud, I image set
Parameters: B budget of pixels, m border margin
 $I_{\text{sample}} \leftarrow \{i \mid i \in I \text{ and } \exists p \in P_{\text{sample}} \text{ s.t. } i \in v(p)\}$
scores \leftarrow array of 0 of size I_{sample}
for $i \in I_{\text{sample}}$ **do**
 $i \leftarrow$ tightest crop(i) to contain P_{sample} with margin m
 scores[i] \leftarrow score($i, P_{\text{sample}}, I_{\text{sample}}, \emptyset$)
end for
batch \leftarrow []
while $B > 0$ and length $I_{\text{sample}} > 0$ **do**
 pick $i \in I_{\text{sample}}$ randomly w.r.t. scores
 batch \leftarrow [batch, i]
 $I_{\text{sample}} \leftarrow I_{\text{sample}} \setminus \{i\}$
 $B \leftarrow B - \text{area}(i)$
 for $i \in I_{\text{sample}}$ **do**
 scores[i] \leftarrow score($i, P_{\text{sample}}, I_{\text{sample}}, \text{batch}$)
 end for
end while

We consider P_{sample} a portion of a point cloud to add to the 3D part of a batch. In order to build the image batch we iteratively select images according to the following procedure. We first select the image set I_{sample} seeing at least one point of P_{sample} . For equirectangular images, we rotate the images to place the mappings at the center. We then crop each image i along the tightest bounding box containing all seen point of P_{sample} with a minimum margin of m along a fixed set of image size along: crops = $\{64 \times 64, 128 \times 64, 128 \times 128, 256 \times 128, 256 \times 256, 512 \times 256, 512 \times 512, 1024 \times 512\}$. We

then associate with each cropped image a score defined as follows:

$$\text{score}(i, P_{\text{sample}}, I_{\text{sample}}, \text{batch}) = \frac{\text{area}(i)}{\max(\text{area}(i) \mid i \in I_{\text{sample}})} + \lambda \frac{\text{unseen}(i, P_{\text{sample}}, \text{batch})}{\max(\text{unseen}(i, P_{\text{sample}}, \text{batch}) \mid i \in I_{\text{sample}})}, \quad (\text{D-4})$$

with $\text{area}(i)$ the area of the cropped image i in pixels, batch the current image batch, $\text{unseen}(i, P_{\text{sample}}, \text{batch})$ the number of points of P_{sample} seen in image i but not in any image of the current batch, and $\lambda = 2$ a parameter controlling the trade-off between maximum area and maximum coverage. The current image batch is initialized as an empty set, but the scores must be updated as it is filled. The images are chosen randomly with a probability proportional to their score. We chose in all experiments a margin $m = 8$ pixels and a budget corresponding to 4 full resolution mapping.

D-4 Implementation Details

Our method is developed in Pytorch and is implemented within the open-source framework Torch-Points3d [46].

For our backbone 3D network, we use TorchPoint3D’s [46] Res16UNet34 implementation of MinkowskiNet [58]. This UNet-like architecture comprises 5 encoding layers and 5 decoding layers. Encoding layers are composed of a strided convolution of $\text{kernel_size} = [3, 2, 2, 2, 2]$ and $\text{stride} = [1, 2, 2, 2, 2]$ followed by $N = [0, 2, 3, 4, 6]$ ResNet blocks [119] of channel size $[128, 32, 64, 128, 256]$, respectively. The decoding layers are built in the same manner, with a strided transposed convolution of $\text{kernel_size} = [2, 2, 2, 2, 3]$ and $\text{stride} = [2, 2, 2, 2, 1]$ as their first operation, followed by a concatenation with the corresponding skipped-features from the encoder and $N = 1$ ResNet block of channel size $[128, 128, 96, 96, 96]$, respectively. A fully connected

linear layer followed by a softmax converts the last features into class scores. Unless specified otherwise, ReLU activation and batch normalization [143] are used across the architecture.

For our 2D encoders, we use the encoder part of pretrained ResNet18 networks [118]. For indoor scenes, we use the modified ResNet18 from [63] pretrained on ADE20K [362], which has 5 layers of output channel sizes [128, 64, 128, 256, 512] and resolution scale = [4, 4, 8, 8, 8]. The relatively high resolution of the output feature map allows us to map the 2D features of size $C = 512$ from the last layer directly to the point cloud without upsampling. For outdoor scenes, we use the ResNet18 from [187] pretrained on Cityscapes [62], which has 5 layers of output channel sizes [128, 64, 128, 256, 512] and resolution scale = [4, 4, 8, 16, 32]. We use pyramid feature pooling [358] on layers 1, 2, 3, and 4, which results in a feature vector of size $C = 960$ passed to the mapped points.

For our DeepViewAgg module, the extracted image features are converted into view features of size C with the MLP ϕ_0 of size $C \mapsto C \mapsto C$. For the computation of quality scores, we use the following MLPs: $\phi_1 : 8 \mapsto M \mapsto M$, $\phi_2 : M \mapsto M \mapsto M$, and $\phi_3 : 2M \mapsto M \mapsto K$ with $M = 32$ and $K = 4$.

For indoor scenes, we use the lowest resolution map of a network [63] pretrained on ADE20K [362]. For the outdoor scenes, we use pyramid feature pooling [358] with a network [187] pretrained on Cityscapes [62]. We use early fusion in all experiments unless specified otherwise.

All models are trained with SGD with an initial learning rate of 0.1 for 200 epochs with decay steps of 0.3 at epoch 80, 120, 160 for indoor datasets, and 20 epochs with decay at epoch 10, 16 for the outdoor dataset. The pretrained 2D networks use a learning rate 100 times smaller than the rest of the model. We use random rotation and jittering for point clouds, random horizontal flip and color jittering for images, and featurewise jittering for descriptors of the

viewing conditions.

For more details on the implementation, we refer the reader to our provided code.

D-5 S3DIS Adjustment

We adjusted some room and image positions in S3DIS [13] and <https://github.com/alexsax/2D-3D-Semantics> to recover mappings between points and equirectangular images. More specifically, we rotate `hallway_11` from `Area_2` and `hallway_6` from `Area_5` by 180° around the Z-axis, and we shift and rotate all images in `Area_5b` by the same manually-found corrective offset and angle. These fixes are all available in our repository.

D-6 Detailed Results

We report in Table D-1, Table D-2, and Table D-3 the class-wise performance across all datasets. We see a clear improvement for indoor datasets for classes such as windows, boards, and pictures. These are expected results because these classes are hard to parse in 3D but easily identified in 2D. Besides, we can see that S3DIS’s classes such as beams, columns, chairs, and tables also benefit from the contextual information provided by images. For the KITTI-360 dataset, the multimodal model outperforms the 3D-only baseline for all classes. We can see the benefit of image features on small objects or underrepresented classes in 3D, such as traffic signs, persons, trucks, motorcycles, and bicycles.

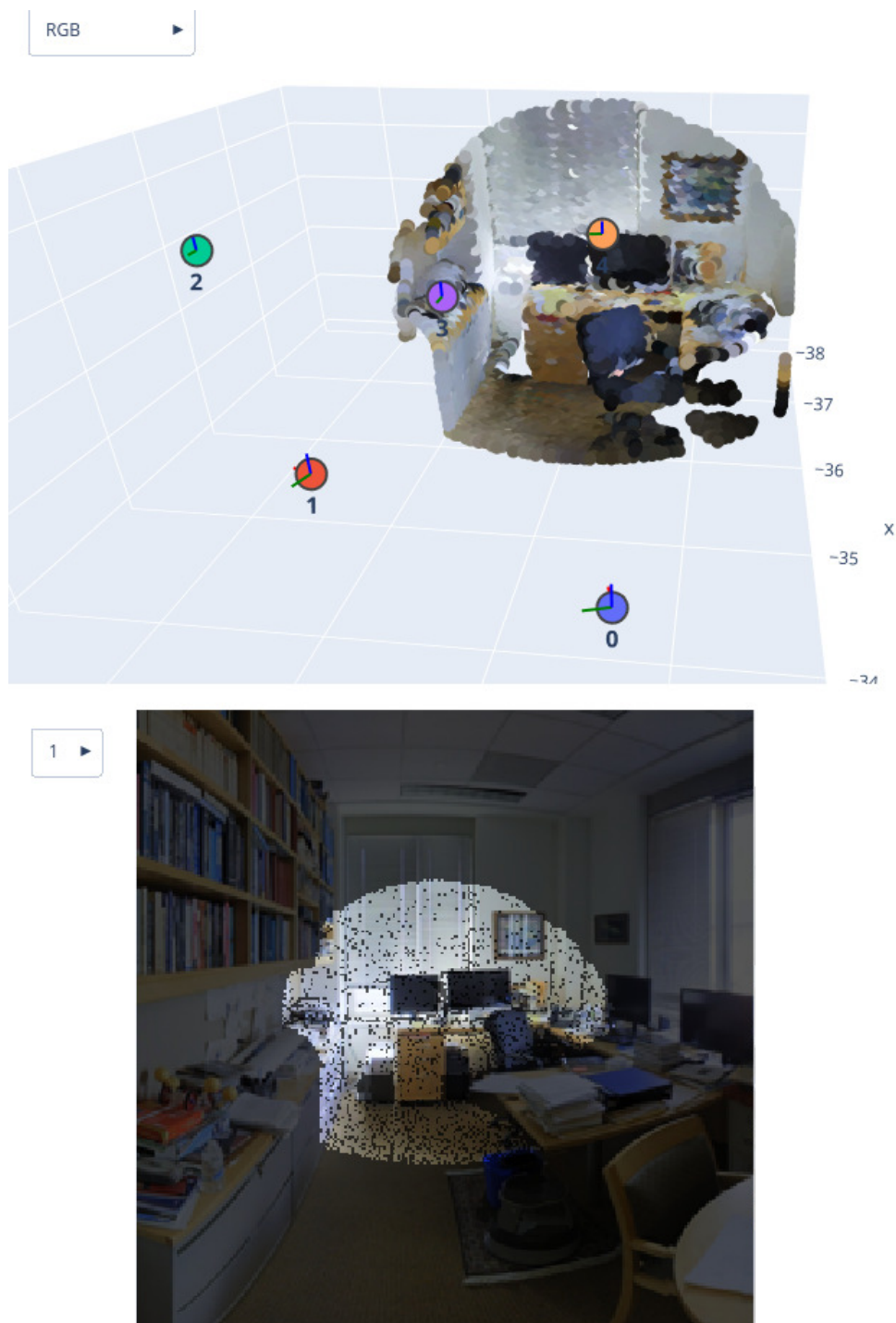


Figure D-1 – **Interactive Visualization.** We propose interactive visualizations of hybrid 2D/3D data along with predictions of our model. We also provide the code necessary to create more such visualizations.

Table D-1 – **S3DIS Class-wise Performance.** Class-wise mIoU on S3DIS for our 3D backbone network with and without learned multi-view aggregation.

Method	Avg	S3DIS FOLD 5												
		ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter
3D Backbone	64.7	88.6	97.5	82.1	0.0	16.6	53.8	66.2	89.1	78.2	73.4	66.0	69.6	59.2
+ DeepViewAgg	67.2	87.2	97.3	84.3	0.0	23.4	67.6	72.6	87.8	81.0	76.4	54.9	82.4	58.7
S3DIS 6-FOLD														
3D Backbone	69.5	91.2	90.6	83.0	59.8	52.3	63.2	75.7	63.2	64.0	69.0	72.1	60.1	59.2
+ DeepViewAgg	74.7	90.0	96.1	85.1	66.9	56.3	71.9	78.9	79.7	73.9	69.4	61.1	75.0	65.9

Table D-2 – ScanNet Val. Class-wise Performance. Class-wise mIoU on ScanNet Val. for our 3D backbone network with and without learned multi-view aggregation.

Method	ScanNet																				
	Avg	wall	floor	cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	refrigerator	shower cur.	toilet	sink	bathrub	other
3D Backbone	69.0	82.7	94.5	62.2	77.9	88.9	80.7	70.3	61.4	60.2	80.0	28.4	60.3	60.8	70.4	40.2	67.4	80.9	62.6	85.3	51.0
+ DeepViewAgg	71.0	84.3	94.4	57.8	78.9	90.1	78.0	71.9	63.4	66.9	77.8	39.9	62.2	55.6	71.9	57.4	71.4	89.5	66.2	82.8	56.3

Table D-3 – **KITTI-360 Class-wise Performance.** Class-wise mIoU on KITTI-360 for our 3D backbone network with and without learned multi-view aggregation.

Method	Avg	KITTI-360 Val															
		road	sidewalk	building	wall	fence	pole	traffic lig.	traffic sig.	vegetation	terrain	person	car	truck	motorcycle	bicycle	
3D Backbone	54.2	90.6	74.4	84.5	45.3	42.9	52.7	0.5	38.6	87.6	70.3	26.9	87.3	66.0	28.2	17.2	
+ DeepViewAgg	57.8	93.5	77.5	89.3	53.5	47.1	55.6	18.0	44.5	91.8	71.8	40.2	87.8	30.8	39.6	26.1	

Annexe E

Résumé Long en Français

Apprentissage Efficace sur Nuages de Points 3D à Grande Échelle

E-1 Introduction



FIGURE E-1 – **Analyse de Scènes 3D à Grande Échelle.** Nous développons des méthodes pour l’analyse sémantique efficace de grands nuages de points 3D. Ici, nous présentons les résultats de notre travail sur la segmentation panoptique, sur S3DIS [14]. Alors que la plupart des méthodes opèrent sur une pièce à la fois, la nôtre peut en considérer des centaines. En particulier, nous atteignons l’état de l’art tout en traitant des bâtiments de dizaines de millions de points en une seule inférence, sur un seul GPU, en seulement 7.4 secondes.

Cette thèse aborde l’analyse automatisée de scènes 3D, pouvant trouver des applications dans des domaines tels que la surveillance de l’environnement et la planification urbaine. Contrairement à la tendance actuelle consistant

à entraîner des modèles 3D massifs avec d'énormes jeux de données, nous préconisons des approches sobres et efficaces, adaptées aux chercheurs et praticiens disposant de ressources limitées. Les méthodes proposées sont conçues pour traiter des scènes 3D étendues sans compromettre les performances, avec une évaluation sur des jeux de données publics et un code open source pour la reproductibilité. En résumé, cette thèse vise des solutions accessibles, efficaces et respectueuses de l'environnement pour l'analyse de nuages de points 3D.

Dans ce chapitre introductif, nous situons d'abord cette thèse dans le contexte de l'histoire récente de l'apprentissage profond en 3D. Ensuite, nous exposons nos motivations pour travailler sur l'analyse efficace de grands nuages de points 3D, ainsi que les défis correspondants à nos objectifs. Nous présentons ensuite les principales contributions de cette thèse. Enfin, nous résumons la structure du présent document.

Au cours de la dernière décennie, la recherche en apprentissage profond a connu des avancées notables, passant par les réseaux neuronaux convolutifs à l'adoption généralisée des architectures de type transformer et à la révolution de l'apprentissage auto-supervisé. Le domaine a été marqué par la montée en puissance de modèles toujours plus grands et par l'essor de l'apprentissage auto-supervisé, mais ces avancées ont été principalement axées sur le traitement du texte et de l'image. En parallèle, la vision par ordinateur 3D a évolué avec l'explosion des données 3D et l'émergence d'architectures adaptées, bien que les jeux de données annotés restent limités. Cette thèse se positionne dans ce contexte, cherchant à développer des méthodes d'apprentissage profond 3D efficaces malgré des données moins abondantes et des ressources plus modestes.

Cette thèse se concentre sur des méthodes efficaces pour analyser de grands nuages de points 3D, notamment dans des domaines tels que la conduite autonome et la gestion urbaine. Nous justifions cette approche en considération

des tendances actuelles de l'apprentissage profond 3D, soulignant la nécessité de l'efficacité face à la course à la taille des modèles. Pour guider notre travail, nous proposons une typologie de cinq dimensions d'efficacité : mémoire, calcul, matériel, données et travail humain. En résumé, notre objectif est de développer des techniques d'apprentissage profond 3D efficaces pour traiter efficacement les vastes scènes 3D du monde réel.

Les nuages de points offrent une représentation imparfaite des scènes 3D en raison de l'absence de connectivité entre les points, entraînant des défis tels que la perte de détails géométriques fins et des caractéristiques distinctes selon les techniques d'acquisition. Pour surmonter ces défis, cette thèse se concentre sur trois aspects clés : le raisonnement multiniveau efficace, la fusion multimodale efficace et une mise en œuvre efficiente. Le raisonnement multiniveau efficace repose sur une représentation de données hiérarchique adaptative à la complexité géométrique de la scène. La fusion multimodale efficace propose une méthode pour extraire et fusionner efficacement des descripteurs à partir de nuages de points et d'images, sans dépendre d'une acquisition spécifique ou de prétraitements coûteux. Enfin, une mise en œuvre efficace est soulignée comme un aspect essentiel, nécessitant des efforts d'ingénierie algorithmique pour surmonter les défis liés à la nature non structurée et variante en taille des nuages de points 3D. Ces contributions visent à relever les défis liés au traitement efficace des vastes scènes 3D du monde réel.

Cette thèse propose deux contributions majeures pour l'analyse efficace de scènes 3D à grande échelle. La première consiste en un modèle léger pour l'analyse sémantique des nuages de points, surpassant les méthodes existantes avec moins de paramètres et des temps plus rapides. La seconde est une architecture capable d'extraire des informations à partir de nuages de points 3D et d'images 2D de manière conjointe, avec une implémentation

parcimonieuse et parallélisée. En résumé, ces contributions exploitent les structures des nuages de points 3D et des données multimodales pour créer des architectures d'apprentissage profond performantes, efficaces et évolutives.

Chapitre 2 : État de l'Art. Introduction aux principales familles de modèles d'apprentissage profond 3D. Revue des stratégies existantes pour l'apprentissage profond efficace sur images et nuages de points. Présentation des méthodes basées sur les superpoints pour le traitement 3D efficace et aperçu des approches d'apprentissage multimodal avec les nuages de points.

Chapitre 3 : Segmentation Sémantique Efficace à Grande Echelle. Présentation de notre architecture de *transformer* basée sur les superpoints pour effectuer efficacement la segmentation sémantique de scènes 3D à grande échelle. Cette méthode repose sur un algorithme rapide de partitionnement des nuages de points en une structure hiérarchique de superpoints, ainsi que sur un mécanisme d'auto-attention pour apprendre les relations entre les superpoints à plusieurs échelles. Nous démontrons des performances de pointe sur trois référentiels de segmentation sémantique 3D, avec jusqu'à $200\times$ moins de paramètres et jusqu'à $70\times$ d'entraînement plus rapide par rapport aux approches concurrentes.

Chapitre 4 : Segmentation Panoptique Efficace à Grande Echelle. Formulation de la tâche de segmentation panoptique 3D comme un problème évolutif de partitionnement de graphe, que peut traiter un petit modèle en se basant uniquement sur des objectifs locaux. Notre cadre contourne plusieurs limitations des méthodes concurrentes et peut être étendu naturellement au paradigme des superpoints présenté dans le Chapitre 3, permettant une mise à l'échelle efficace vers des scènes 3D vastes. Nous atteignons des performances de pointe sur quatre référentiels de segmentation panoptique 3D, démontrant l'efficacité

de notre méthode à l'entraînement et à l'inférence.

Chapitre 5 : Apprentissage sur Nuages de Points et Images Arbitrairement Localisées. Proposition d'une méthode d'agrégation multi-vues de bout en bout pour la segmentation sémantique 3D à partir d'images et de nuages de points. Nous atteignons des performances de pointe sur deux référentiels de segmentation sémantique 3D sans nécessiter de colorisation des nuages de points, de maillage ou de capteurs de profondeur : uniquement des nuages de points, des images et leurs poses.

Chapitre 6 : Conclusion. Résumé des contributions de cette thèse et discussion des orientations futures pour la vision informatique 3D efficace.

E-2 État de l'Art

Nos contributions pour le traitement efficace de grands nuages de points s'appuient sur diverses parties de la littérature détaillées dans ce chapitre. Nous présentons ici les principales familles de modèles d'apprentissage profond 3D et examinons les stratégies existantes pour l'apprentissage profond efficace, en mettant l'accent sur les méthodes spécifiques aux nuages de points. Nous développons également des directions de recherche basées sur les superpoints, qui ont inspiré notre travail dans les chapitres Chapitre 3 et Chapitre 4. Enfin, nous présentons un aperçu des approches liées au cadre d'apprentissage multimodal proposé dans le chapitre Chapitre 5.

Les architectures d'apprentissage profond 3D peuvent être largement catégorisées selon la représentation des données sur laquelle elles opèrent : basées sur des voxels, sur des images et sur des points. Les méthodes basées sur des images projettent des nuages de points 3D en plusieurs vues 2D, exploitant la vision informatique 2D pour analyser la scène. Les méthodes basées sur

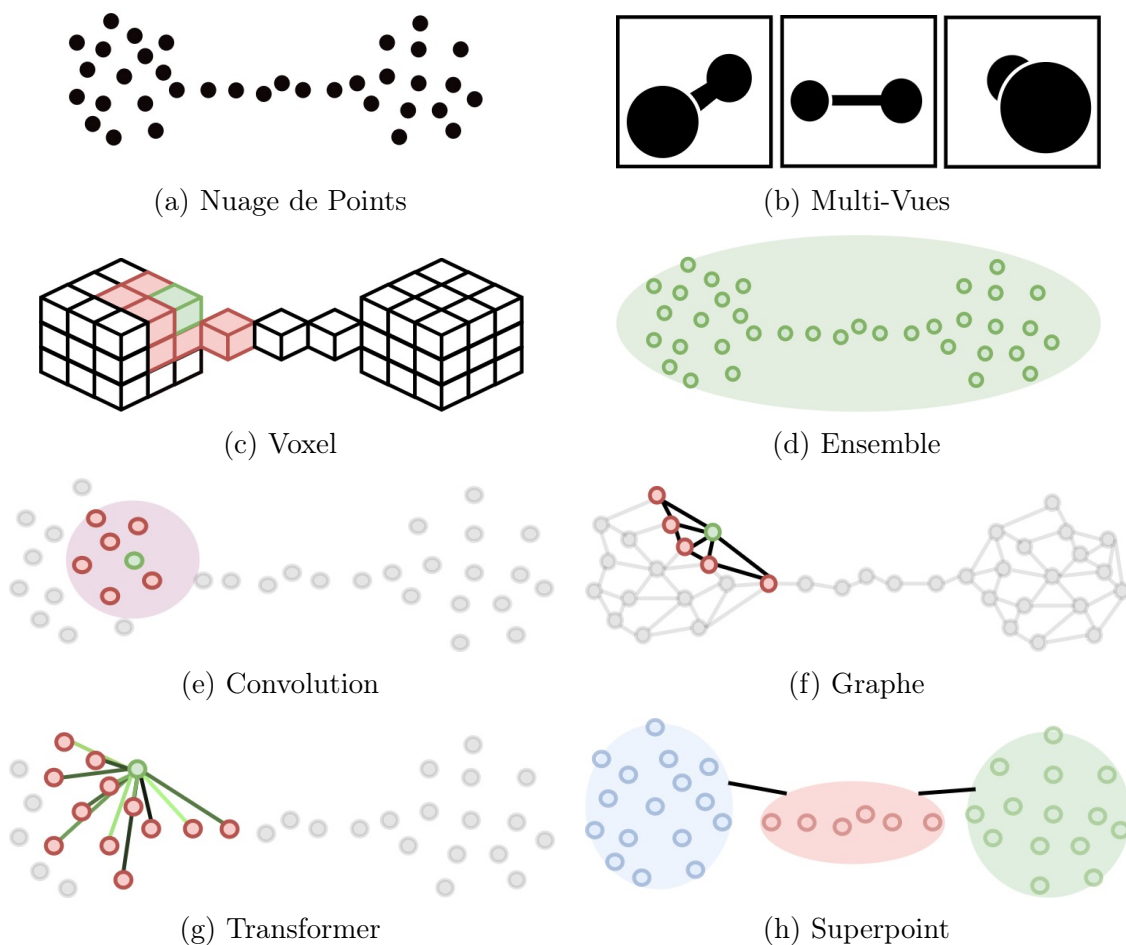


FIGURE E-2 – **Méthodes d'apprentissage profond 3D.** Une variété de méthodes ont été proposées pour extraire des descripteurs à partir de nuages de points E-2a. Certaines approches éliminent des informations en rendant des vues 2D des points E-2b, ou en discrétisant le nuage dans une grille de voxels E-2c. D'autres méthodes opèrent directement sur des ensembles non ordonnés de points E-2d, ou généralisent les convolutions discrètes 2D à l'espace continu 3D E-2e. D'autres alternatives utilisent des réseaux neuronaux graphiques E-2f ou des *transformers* E-2g pour raisonner sur un voisinage local. Étroitement liées à notre travail, les approches basées sur les superpoints E-2h raisonnent sur une partition de la scène.

des voxels convertissent les nuages de points 3D en grilles de voxels 3D. Les méthodes basées sur des points opèrent directement sur le nuage de points sans perte d'information explicite par discrétisation ou projection. Ces méthodes peuvent être divisées en méthodes basées sur des ensembles, des convolutions, des graphes, des *transformers* et des superpoints. Dans cette thèse, nous proposons des méthodes à l'intersection de ces catégories, avec un modèle

efficace et scalable héritant des méthodes de *transformers* et de superpoints, et une approche d'apprentissage multimodal 2D-3D inspirée des approches basées sur des images et des voxels.

L'analyse efficace des nuages de points 3D est essentielle compte tenu de la taille considérable de certains scans 3D. Elle présente d'abord une vue d'ensemble des directions de recherche pour l'apprentissage profond efficace, couvrant la compression, l'entraînement, l'automatisation, l'implémentation et l'architecture. Les stratégies incluent la compression pour réduire la taille des réseaux neuronaux, l'optimisation des processus d'entraînement, l'automatisation de la recherche d'hyperparamètres, des mises en œuvre plus efficaces, et la conception de nouveaux blocs de construction. En ce qui concerne l'efficacité des nuages de points 3D, plusieurs architectures sont explorées, telles que PointNet, RandLANet, et PointNeXt, montrant des performances rapides et des analyses 3D efficaces. Des représentations de données efficaces, comme les convolutions sur grilles de voxels épars, sont également étudiées pour réduire l'empreinte mémoire des modèles tout en conservant une résolution élevée. Cependant, les méthodes basées sur des superpoints, exploitant la similarité locale des nuages de points denses, permettent une réduction significative de l'entrée, assurant une efficacité inégalée.

Les superpixels, appliqués aux images, sont utilisés pour simplifier la détection d'objets et la segmentation sémantique. De manière similaire, les superpoints, appliqués aux nuages de points 3D, sont efficaces pour l'over-segmentation et la segmentation sémantique. Les méthodes basées sur les superpoints peuvent être de type regroupement ou graphiques, s'adaptant à la géométrie 3D sans hypothèses sur le nombre de superpoints. Pour la segmentation sémantique, SPG utilise des convolutions graphiques pour apprendre les relations entre les superpoints. Les partitions hiérarchiques dans le Chapitre 3 sont adaptatives à la géométrie locale, permettant une modélisation précise

des interactions entre les objets. Pour la segmentation panoptique dans le Chapitre 4, notre approche utilise un regroupement graphique rapide, améliorant ainsi l'efficacité et la scalabilité par rapport aux méthodes existantes.

Dans le Chapitre 5, notre méthode propose d'apprendre à agréger des informations à partir de vues arbitrairement orientées du même objet 3D, en fonction de leurs conditions d'observation. Des méthodes utilisant des mécanismes d'attention pour apprendre des représentations multimodales ont été étudiées, notamment pour fusionner des informations textuelles et visuelles, ainsi que des vidéos. Certains travaux récents montrent la pertinence de l'utilisation de l'attention pour fusionner des modalités, comme celui de Li et al., qui fusionne la sémantique 2D et l'occupation 3D pour la complétion de scène. Dans le domaine de l'analyse de scènes 2D/3D avec l'apprentissage profond, des réseaux dédiés à la modalité 3D ont atteint des performances impressionnantes, mais nécessitent souvent une profondeur réelle pour chaque pixel, ce qui limite leur applicabilité dans des environnements réels. Notre travail dans le Chapitre 5 présente la première approche multimodale de bout en bout qui apprend à agréger des informations multi-vues en fonction des conditions d'observation.

E-3 Segmentation Sémantique Efficace à Grande Echelle

Nous introduisons une nouvelle architecture novatrice de transformer basée sur les superpoints pour une segmentation sémantique efficace de vastes scènes 3D. Notre méthode intègre un algorithme rapide pour partitionner les nuages de points en une structure hiérarchique de superpoints, rendant notre prétraitement 7 fois plus rapide que les approches existantes basées sur les superpoints. De plus, nous exploitons un mécanisme d'auto-attention pour capturer les relations entre les superpoints à plusieurs échelles, conduisant à des performances de pointe sur trois jeux de données de référence difficiles :

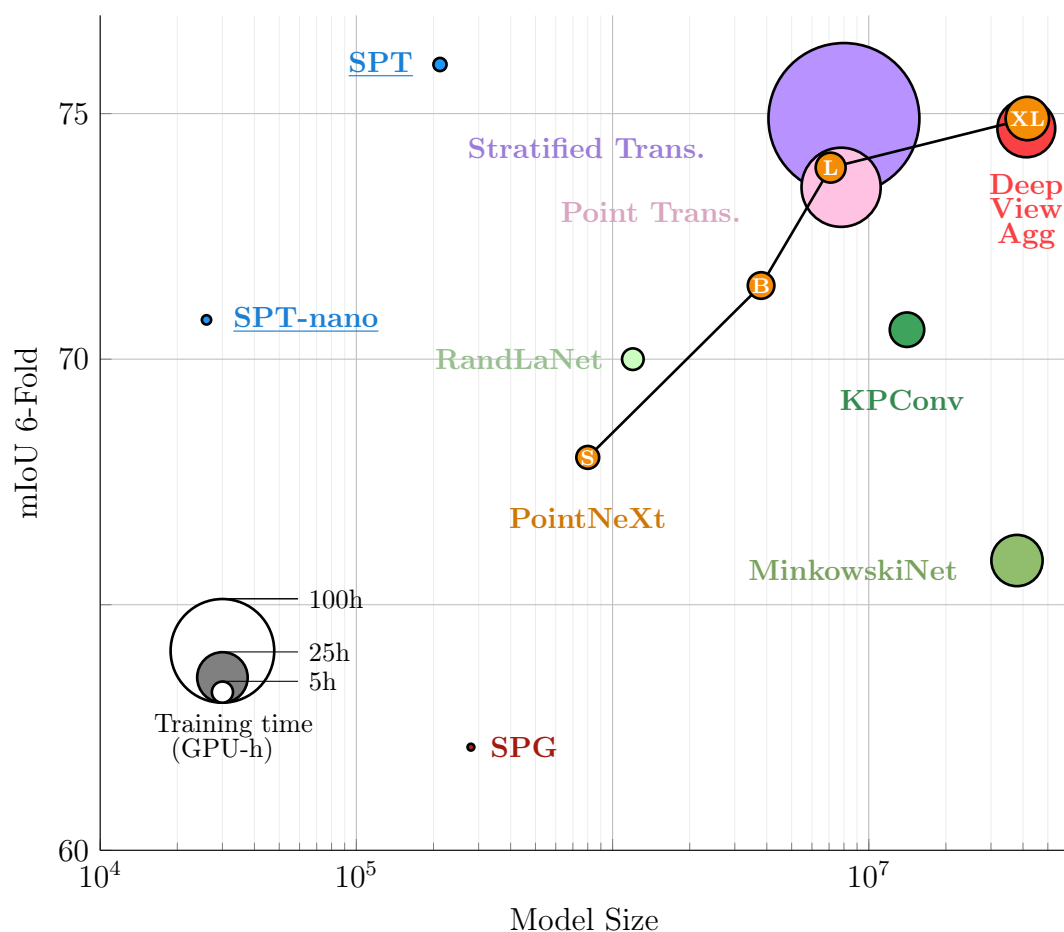


FIGURE E-3 – **Taille du modèle vs Performance.** Nous visualisons les performances de différentes méthodes sur S3DIS (validation croisée) en relation avec leur taille de modèle en échelle logarithmique. La superficie des marqueurs indique le temps GPU nécessaire pour l’entraînement sur un seul *fold*. Notre méthode proposée Superpoint Transformer (SPT) atteint l’état de l’art avec une réduction allant jusqu’à 200 fois de la taille du modèle et 70 fois du temps d’entraînement (en heures GPU) par rapport aux méthodes récentes. Le modèle encore plus petit, SPT-nano, atteint des performances correctes avec seulement 26k paramètres.

S3DIS (76.0% mIoU, validation 6-fold), KITTI-360 (63.5% sur Val) et DALES (79.6%). Avec seulement 212k paramètres, notre approche est jusqu’à 200 fois plus compacte que d’autres modèles de pointe tout en maintenant des performances similaires. De plus, notre modèle peut être entraîné sur un seul GPU en 3 heures pour un *fold* de S3DIS, nécessitant de 7 à 70 fois moins d’heures GPU que les méthodes les plus performantes. Notre code et nos poids sont accessibles sur https://github.com/drprojects/superpoint_transformer.

E-4 Segmentation Panoptique Efficace à Grande Echelle



FIGURE E-4 – **Segmentation Panoptique à Grande Échelle**. Nous présentons les résultats de SuperCluster pour l'ensemble de la Zone 5 de S3DIS [14] (plafond supprimé pour la visualisation) avec 9.2 millions de points (78 millions avant sous-échantillonnage) et 1863 objets "things". Notre modèle peut traiter une telle scène en une seule inférence sur un seul GPU V100-32GB en 3.3 secondes et atteindre un PQ de pointe de 46.3.

Nous présentons une méthode hautement efficace pour la segmentation panoptique de vastes nuages de points 3D en redéfinissant cette tâche comme un problème de regroupement de graphes scalable. Cette approche peut être entraînée en n'utilisant que des tâches auxiliaires locales, éliminant ainsi l'étape coûteuse en ressources d'appariement d'instances pendant l'entraînement. De plus, notre formulation peut facilement être adaptée au paradigme des superpoints, augmentant encore son efficacité. Cela permet à notre modèle de traiter des scènes avec des millions de points et des milliers d'objets en une seule inférence. Notre méthode, appelée SuperCluster, atteint une

nouvelle performance de pointe en segmentation panoptique pour deux jeux de données de scènes d'intérieure : 46,3 PQ (+4.0) pour S3DIS Area 5 et 54,5 PQ (+21.0) pour ScanNetV2. Nous établissons également la première performance de pointe pour deux benchmarks de cartographie mobile à grande échelle : KITTI-360 et DALES. Avec seulement 209 000 paramètres, notre modèle est plus de 30 fois plus petit que la meilleure méthode concurrente et s'entraîne jusqu'à 15 fois plus rapidement. Notre code et nos modèles pré-entraînés sont disponibles sur github.com/drprojects/super_cluster.

E-5 Apprentissage sur Nuages de Points et Images Arbitrairement Localisées

Des travaux récents sur la segmentation sémantique 3D proposent d'exploiter la synergie entre images et nuages de points en traitant chaque modalité avec un réseau dédié et en projetant les descripteurs 2D apprises sur les points 3D. Fusionner d'importants nuages de points et des images soulève plusieurs défis, tels que la construction d'une correspondance entre points et pixels et l'agrégation de descripteurs entre plusieurs vues. Les méthodes actuelles nécessitent une reconstruction de maillage ou des capteurs spécialisés pour récupérer les occultations et utilisent des heuristiques pour sélectionner et agréger les images disponibles. En revanche, nous proposons un modèle d'agrégation multi-vues entraînable de bout en bout exploitant les conditions de vision des points 3D pour fusionner les descripteurs d'images prises à des positions arbitraires. Notre méthode peut combiner des réseaux 2D et 3D standard et surpasse à la fois les modèles 3D opérant sur des nuages de points colorisés et les réseaux hybrides 2D/3D sans nécessiter de colorisation, de maillages ou de cartes de profondeur réelles. Nous établissons une nouvelle référence en segmentation sémantique intérieure/extérieure à grande



FIGURE E-5 – **Fusion Multimodale et Multi-vues.** Nous proposons de fusionner les informations complémentaires entre les nuages de points et un ensemble d’images localisées. En utilisant un modèle de visibilité simple, nous pouvons projeter les descripteurs 2D sur les points 3D et utiliser les conditions de visualisation pour sélectionner les descripteurs des images les plus pertinentes. Nous représentons les images à leur position avec le symbole  et colorons les points 3D en fonction de l’image dans laquelle ils sont vus.

échelle sur S3DIS (74.7 mIoU en validation croisée) et sur KITTI-360 (58.3 mIoU). Notre pipeline complet nécessite uniquement des scans 3D bruts et un ensemble d’images et de poses. Notre code est accessible publiquement sur <https://github.com/drprojects/DeepViewAgg>.

E-6 Conclusion

Ce chapitre de conclusion récapitule nos contributions, avant d’esquisser les orientations de recherche futures que nous identifions comme prometteuses.

Cette thèse apporte des contributions significatives à la vision par ordinateur 3D, avec trois méthodes novatrices :

Chapitre 3 : Segmentation Sémantique Efficace à Grande Echelle. Une approche efficace pour la segmentation sémantique 3D, combinant superpoints et *transformers*, surpassant les méthodes concurrentes en performances tout en étant beaucoup plus compacte et rapide à entraîner.

Chapitre 4 : Segmentation Panoptique Efficace à Grande Echelle. Une formulation novatrice de la segmentation panoptique 3D comme un problème de partitionnement de graphe, avec un modèle léger (SuperCluster) atteignant des performances de pointe et capable de traiter de grandes scènes sur un seul GPU.

Chapitre 5 : Apprentissage sur Nuages de Points et Images Arbitrairement Localisées. Une méthode d'agrégation multi-vue pour la segmentation sémantique 3D à partir d'images et de nuages de points, dépassant les méthodes existantes sans nécessiter de colorisation ou de maillage des nuages de points.

Ces travaux ouvrent des perspectives pour des recherches futures dans le domaine de la vision 3D.

Apprentissage basé sur les superpoints.

- Expressivité du modèle : Pour améliorer l'expressivité de SPT, l'utilisation d'un encodeur de points plus expressif inspiré de KPConv ou MinkowskiNet est envisagée.
- Tokenisation des nuages de points : La partition des nuages de points en primitives géométriquement homogènes pourrait devenir une étape de prétraitement standard, similaire à la tokenisation pour le traitement du langage naturel.
- Partition apprenable : L'apprentissage de la partition en temps réel est envisagé, bien que la non-différentiabilité de l'étape de partition actuelle soit un défi.

- Raisonnement Super-X : L'extension du cadre à d'autres modalités, comme les images, est suggérée, ouvrant la voie à des adaptations pour des modalités multiples.

Apprentissage Multimodal 2D-3D.

- Aggrégation multi-vue multi-capteurs : L'adaptation de DeepViewAgg pour tenir compte de plusieurs capteurs d'image avec des propriétés optiques différentes est suggérée.
- Ajout de modalités : L'extension de DeepViewAgg à d'autres modalités complémentaires est proposée, telles que les capteurs radar, les cartes HD, les images de rue et les images satellite.
- Mapping robuste 2D-3D : L'intégration de la temporalité pour prendre en compte les mouvements d'objets et la robustesse aux erreurs de paramètres de la caméra est recommandée.
- Apprentissage multimodal Super-X : La combinaison des paradigmes Super-X et multimodal offre des possibilités intéressantes pour l'apprentissage auto-supervisé à grande échelle et l'adaptation naturelle à d'autres modalités.

En résumé, ces perspectives suggèrent des directions prometteuses pour de futurs développements en vision par ordinateur 3D et en apprentissage multimodal.

