



HAL
open science

Real-World 3D Data Analysis: Toward Efficiency and Interpretability

Romain Loiseau

► **To cite this version:**

Romain Loiseau. Real-World 3D Data Analysis: Toward Efficiency and Interpretability. Machine Learning [cs.LG]. École des Ponts ParisTech, 2023. English. NNT : 2023ENPC0028 . tel-04481526

HAL Id: tel-04481526

<https://pastel.hal.science/tel-04481526>

Submitted on 28 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.



École des Ponts
ParisTech



IGN

INSTITUT NATIONAL
DE L'INFORMATION
GÉOGRAPHIQUE
ET FORESTIÈRE



THÈSE DE DOCTORAT de l'École des Ponts ParisTech

Real-World 3D Data Analysis: Toward Efficiency and Interpretability

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

Spécialité de doctorat: Informatique

Thèse préparée au sein de l'équipe A3SI (Imagine, École des Ponts
ParisTech) du Laboratoire d'Informatique Gaspard Monge (LIGM) et
de l'équipe STRUDEL du Laboratoire en Sciences et Technologies
de l'Information Géographique (LASTIG)

Thèse soutenue le 27 septembre 2023, par
Romain LOISEAU

Composition du jury:

Edmond BOYER
Research Director, Meta Reality Labs

Rapporteur

Bertrand LE SAUX
Senior Scientist, Φ -lab (ESA/ESRIN)

Rapporteur

Zorah LAEHNER
Postdoctoral Researcher, University of Siegen

Examinatrice

Jean PONCE
Research Director, INRIA

Examineur

Oriane SIMEONI
Research Scientist, valeo.ai

Examinatrice

Mathieu AUBRY
Researcher, LIGM/École des Ponts ParisTech

Directeur de thèse

Loic LANDRIEU
Researcher, LIGM/École des Ponts ParisTech - LASTIG/IGN

Co-directeur de thèse

Ce manuscrit est dédié à mon père Benoit Loiseau.

Abstract

This thesis explores new deep learning approaches for modeling and analyzing real-world 3D data. 3D data processing is helpful for numerous high-impact applications, such as autonomous driving, territory management, industry facilities monitoring, forest inventory, and biomass measurement. However, the high annotation and processing cost of large-scale 3D data makes it particularly hard to use machine learning approaches. Furthermore, the difficulty of visually analyzing 3D data worsens the lack of interpretability of deep learning methods, limiting their adoption.

The computer vision community has proposed numerous methods to analyze 3D data for tasks such as shape classification, scene segmentation, and scene decomposition. Instead of leveraging world knowledge through hand-crafted descriptors, modern deep learning relies on large datasets for superior performance. However, they typically require considerable annotations and computations, and their decisions are often difficult to understand. In this thesis, we propose contributions that address all these limitations.

The first contribution of this thesis is an efficient deep architecture for analyzing dynamic LiDAR scans in real-time. Our approach explicitly models the acquisition geometry of rotating LiDAR sensors widely used for autonomous driving. In contrast to methods that process data acquired through entire rotations of the sensor, our model considers small angular rotations. Our proposed architecture achieves *state-of-the-art* accuracy with processing time reduced by five times and model size by fifty times compared to recent methods.

The second contribution is a deep learning method to summarize extensive 3D shape collections with a small set of 3D template shapes. We learn end-to-end a small number of 3D prototypical shapes that can be transformed to reconstruct an input point cloud. Our approach uses compact and interpretable representations in 3D space that can be viewed and manipulated, facilitating the annotation of unstructured 3D shape collections. This allows us to achieve *state-of-the-art* results for few-shot semantic segmentation.

The third contribution expands the above unsupervised analysis framework to parse large real-world 3D scans into interpretable parts. We introduce a probabilistic reconstruction model that decomposes an input 3D point cloud into a small set of learned prototypical shapes. Our network can automatically determine how many prototypes to use for each scene. Our visually interpretable representations outperform *state-of-the-art* unsupervised parsing methods in terms of decomposition accuracy. Furthermore, our model trains without any manual annotations.

This thesis also introduces two open-access and large-scale annotated real-world LiDAR datasets: HelixNet and the Earth Parser Dataset. HelixNet is

the largest open-source dataset of rotating LiDAR scans with dense annotations. It provides point-level sensor metadata crucial for precisely measuring the latency of semantic segmentation methods for autonomous driving. The Earth Parser Dataset consists of seven large aerial LiDAR scans, which can be used to evaluate the performance of real-world 3D processing in diverse and realistic environments.

We hope our work will encourage further research on efficient and interpretable models for real-world 3D data processing.

Keywords: Deep learning, unsupervised learning, interpretable methods, efficient machine learning, real-time processing.

Résumé

Cette thèse explore de nouvelles approches d'apprentissage profond pour l'analyse des données 3D du monde réel. Le traitement des données 3D est utile pour de nombreuses applications telles que la conduite autonome, la gestion du territoire, la surveillance des installations industrielles, l'inventaire forestier et la mesure de biomasse. Cependant, l'annotation et l'analyse des données 3D peuvent être exigeantes. En particulier, il est souvent difficile de respecter des contraintes liées à l'utilisation des ressources de calcul ou d'annotations. La difficulté d'interpréter et de comprendre le fonctionnement interne des modèles d'apprentissage profond peut également limiter leur adoption.

Des efforts considérables ont été déployés pour concevoir des méthodes d'analyse des données 3D, afin d'effectuer des tâches telles que la classification de formes ou la segmentation et la décomposition de scènes. Les premières analyses automatisées s'appuyaient sur des descripteurs créés à la main et incorporaient des connaissances préalables sur les acquisitions du monde réel. Les techniques modernes d'apprentissage profond ont de meilleures performances, mais, sont souvent coûteuses en calcul, dépendent de grands ensembles de données annotées, et sont peu interprétables. Les contributions de cette thèse répondent à ces limitations.

Notre première contribution est une architecture d'apprentissage profond pour l'analyse efficace de séquences LiDAR en temps réel. Notre approche prend en compte la géométrie d'acquisition des capteurs LiDAR rotatifs, que de nombreuses pipelines de conduite autonome utilisent. Par rapport aux travaux antérieurs, qui considèrent des rotations complètes des capteurs LiDAR individuellement, notre modèle traite l'acquisition par petits incréments. L'architecture que nous proposons a une performance comparable à celle des meilleures méthodes, tout en réduisant le temps de traitement de plus de cinq fois, et la taille du modèle de plus de cinquante fois.

Notre deuxième contribution est une méthode d'apprentissage profond permettant de résumer de vastes collections de formes 3D à l'aide d'un petit ensemble de formes 3D. Nous apprenons un faible nombre de formes prototypiques 3D qui sont alignées et déformées pour reconstruire les nuages de points d'entrée. Notre représentation compacte et interprétable des collections de formes 3D permet d'obtenir des résultats à l'état de l'art de la segmentation sémantique avec peu d'exemples annotés.

Notre troisième contribution développe l'analyse non supervisée pour la décomposition de scans 3D du monde réel en parties interprétables. Nous introduisons un modèle de reconstruction probabiliste permettant de décomposer un nuage de points 3D à l'aide d'un petit ensemble de formes prototypiques apprises. Nous surpassons les méthodes non supervisées les plus récentes en matière de précision de décomposition, tout en produisant des représentations

visuellement interprétables. Nous offrons des avantages significatifs par rapport aux approches existantes car notre modèle ne nécessite pas d'annotations lors de l'entraînement.

Cette thèse présente également deux jeux de données annotés du monde réel en accès libre, HelixNet et Earth Parser Dataset, acquis respectivement avec des LiDAR terrestres et aériens. HelixNet est le plus grand jeu de données LiDAR de conduite autonome avec des annotations denses, et fournit les métadonnées du capteur pour chaque point, cruciales pour mesurer précisément la latence des méthodes de segmentation sémantique. Le Earth Parser Dataset se compose de sept scènes LiDAR aériennes, qui peuvent être utilisées pour évaluer les performances des techniques de traitement 3D dans divers environnements.

Nous espérons que ces jeux de données, et ces méthodes fiables tenant compte des spécificités des acquisitions dans le monde réel, encourageront la poursuite de la recherche vers des modèles plus efficaces et plus interprétables.

Mots-clés : apprentissage profond, réseau de neurones, traitement en temps réel, apprentissage non supervisé, méthodes interprétables, méthodes efficaces.

Acknowledgements

I want to express my gratitude to all the people who helped me during this journey. Science is a never-ending path, and writing this thesis is the occasion to look back and thank the wonderful people with whom I shared part of the expedition.

I start by thanking my advisors, Loïc Landrieu and Mathieu Aubry. Three years ago, you decided to take me as one of your students for the next few years. I want to thank both of you for the mutual faith and respect our relationship was based on.

Thank you, Loïc, for your rigor, for your elegant ways of reformulating problems, and for all the time we spent writing. I started in a bad state, and I'm not finished learning. However, the way you are constantly challenging all your scientific productions is very inspiring. From other things, I will miss the time we spent debugging our *TikZ* code and beamer presentations.

Thank you, Mathieu, for sharing your view of research through your always insightful comments and ideas. Thanks for the pieces of advice and the long discussions we had about the future of a young graduated student.

Writing papers is a challenging task, and I had the chance to collaborate with excellent people during those years.

Tom Monnier started his thesis a few months before I arrived, and he helped me a lot during my first times in research. His experience and pedagogical approach to computer vision were very insightful. Thank you for the wise bits of advice about the thesis you shared.

In the middle of the thesis, while drinking a beer with Baptiste Bouvier, we had the weird idea of applying the methods I developed to his research field and seeing what would happen. Associated with Yann Teytaut and Elliot Vincent, we had the fantastic opportunity to collaborate at the crossroads of our respective research fields and to learn a lot from each other. Thanks a lot for your engagement and the creativity we had to deploy to make it work.

Elliot Vincent, it was the first time, among others, we collaborated. Thanks for your always very inspiring implication and down-to-earth reflections.

This thesis was only possible with inspiring teachers sharing their love for science. Loïc Laferté was my math teacher during my last years in high school. Thank you for giving your students such pedagogical lessons. We ended up seeing ourselves more than expected, and I am very proud and grateful to have you as a father-in-law. I would also like to thank Renaud Keriven, the researcher who made me want to do this thesis. I followed your advice to apply for the MVA master, for my position as a civil servant, and for this thesis. We met a few times since my internship in your team, and you always shared your experience generously. Thanks a lot for the inspiring humility with which you tackle scientific problems.

Most of my time at the IGN was shared with Emile Blettery, Helen Mair Rawsthorne, Yanis Marchand, and Damien Robert. We now know that a thesis is a journey of days and nights. Thank you for your always well-intentioned support, even in the most challenging moments.

I also want to give a special thank you to Clément Mallet, head of the LASTIG, with whom we spent lots of time discussing the vital yet fragile link between the scientific community and the decision-makers.

Of course, I can't thank everyone individually. However, I really appreciated all the research, sport, and coffee time with Mathieu Brédif, Laurent Caraffa, Paul Chapron, Florent Geniet, Lâ mân Lelégard, Samuel Mermet, Julien Perret, Ewelina Rupnik, Bruno Vallet, Mohamed Ali-Chebbi, Luc Baudoux, Grégoire Grzeczko wicz, Melvin Hersent, Ekaterina Kalinicheva, Solenn Tual, Charles Villard and Mehdi Zrhal.

I also had the opportunity to share excellent research time with people at IMAGINE. Mathis Petrovich, Tom Monnier, Elliot Vincent, Antoine Guédon, Yanis Siglidis, Théo Deprelle, François Darmon, Georgy Ponimatkin, Michaël Ramamonjisoa, Nicolas Dufour, Robin Champenois, Nguyen Nguyen, Zeynep Sonat Baltaci, thank you for all the discussions we had, around the coffee machine or in the offices, sharing both tips and tricks to do good research and more personal life challenges.

I would also like to thank David Picard, head of IMAGINE, with whom we discussed a lot about how science is done. Discussing our research fields' biases with you during our RER trips back to the west of Paris was always very pleasant.

Research could only be done with special help from the administrators. Isabelle Simunic, Stéphanie Bonnel, and Alain Sombris, thank you for supporting the research teams.

To my friends and family. I am very grateful for having you around me. Thank you, Augustin, Jean-Charles, Joseph, and Laetitia, for being such wonderful friends. You are always here when needed, and I really enjoy the time we spend having dinner, walking in Paris, sharing thoughts, painting minis, and discovering whiskies together. I hope it will last. I would also like to thank my parents for everything they gave me. I was pretty annoying as a child as I would always question everything and ask for well-built answers. Thank you for teaching me how to use this gift wisely. I still have a lot to learn.

Dear Mom, dear Thibaut, Mathilde, and Quitterie, we went through hard times, but our strength lies in staying close and constantly getting up. Thank you for our ever-lasting love, which overflows outside the family.

Finally, Emma, I can't thank you enough. The past ten years have been enlightened by your presence next to me. Thank you for your unconditional support and love.

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 1 |
| 1.1 | Goals | 1 |
| 1.2 | Motivations | 2 |
| 1.3 | Challenges | 7 |
| 1.4 | Contributions | 9 |
| 1.5 | Publications and Research Activities | 10 |
| 1.6 | Thesis Outline | 12 |
| 2 | Related Work | 15 |
| 2.1 | 3D Data Representation | 15 |
| 2.2 | 3D Tasks | 25 |
| 2.3 | 3D Point Cloud Datasets | 30 |
| 3 | Online Segmentation of LiDAR Sequences | 35 |
| 3.1 | Introduction | 37 |
| 3.2 | Related Work | 38 |
| 3.3 | A Dataset for Online LiDAR Segmentation | 39 |
| 3.4 | Fast LiDAR Segmentation with Transformers | 43 |
| 3.5 | Evaluating Online Semantic Segmentation | 50 |
| 3.6 | Conclusion | 55 |
| 4 | Exploring Shape Collections | 57 |
| 4.1 | Introduction | 59 |
| 4.2 | Related Work | 61 |
| 4.3 | Modeling Shape Collections | 62 |
| 4.4 | Experiments | 69 |
| 4.5 | Conclusion | 79 |
| 5 | Discovering Prototypes in Aerial Scans | 83 |
| 5.1 | Introduction | 85 |
| 5.2 | Related Work | 86 |
| 5.3 | Method | 87 |
| 5.4 | Results | 96 |
| 5.5 | Conclusion | 107 |

| | |
|--|------------|
| 6 Conclusion | 109 |
| 6.1 Contributions | 109 |
| 6.2 Perspectives | 110 |
| Bibliography | 115 |
| List of Figures | 139 |
| List of Tables | 140 |
| | |
| A Résumé long en français | 143 |
| A.1 Introduction | 143 |
| A.2 État de l’art | 146 |
| A.3 Segmentation automatique de flux LiDAR | 149 |
| A.4 Exploration de collections de formes 3D | 151 |
| A.5 Découverte de prototypes 3D dans des scans aériens | 153 |
| A.6 Conclusion | 155 |

Chapter 1

Introduction

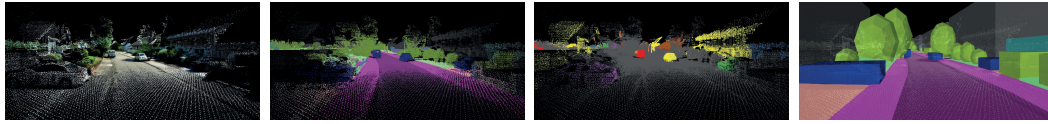
From autonomous driving to territory management, 3D data processing has numerous impactful applications. This thesis presents real-time, unsupervised, and interpretable methods for understanding 3D data. This chapter outlines the goals, motivations, challenges, and contributions of our work.

1.1 Goals

Our work is related to 3D computer vision, and spans autonomous driving, shape modeling, or building reconstruction; see Figure 1.1. In this thesis, we aim to design real-time, unsupervised, and interpretable 3D perception methods.

Real-time 3D data processing. Our first goal is to develop accurate methods for processing real-world 3D acquisitions in real time. In the case of embarked sensors, 3D data streams are particularly challenging to analyze due to the sensor’s complicated geometry, the high acquisition rate, and the real-time constraint. We aim at accelerating embarked data processing with models that have only a fraction of current models’ parameters, while preserving high performance.

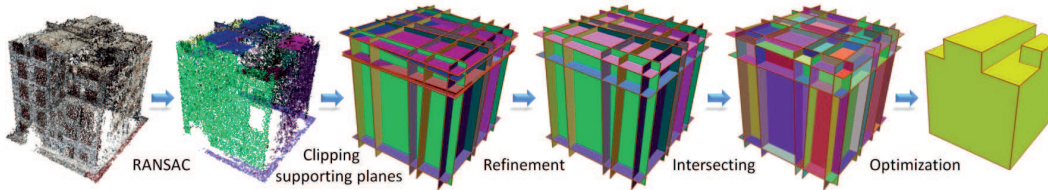
Unsupervised 3D data processing. Our second goal is to gain insight about large scenes or collections of 3D objects without manual annotations. While it is now easier than ever to acquire large quantities of 3D data, they remain particularly costly to annotate, visualize, and understand. To address



(a) LiDAR processing on the KITTI-360 [186] dataset.



(b) Shapes from ABC [158], commonly used ones, and 3D shape decomposition [317].



(c) Lightweight reconstruction of polygonal surfaces from point clouds [236].

Figure 1.1: **3D Representations and Applications.** 3D data can take various forms, from unordered point clouds to highly structured models. They are used for a variety of tasks such as autonomous driving, graphic design and building models.

this issue, we propose to design unsupervised methods for summarizing large shape collections in a viewable format. Our methods aim to assist in understanding, manipulating, and annotating large-scale 3D data. We aim at enabling one to easily perform downstream unsupervised tasks, such as clustering or few-shot semantic segmentation.

Interpretable 3D data processing. Our third goal is to design approaches whose underlying mechanisms are easily understandable. Contrary to deep learning methods that rely on abstract representations, our goal is to learn representations in input space that can be easily manipulated and annotated. Such methods allow us to explore and better understand data, which may prove pivotal to their adoption.

1.2 Motivations

Our work is motivated by the challenges and opportunities of real-world 3D data processing. First, we recognize the potential of 3D processing tools to improve actions and designs of public policies. Second, our objective is to con-

tribute to the development of deep learning models that (i) meet the stringent requirements of applications such as autonomous driving, (ii) contribute to reducing the environmental impact of computer vision by reducing its computing needs, and (iii) improve the acceptability and comprehension of learning-based methods by increasing their interpretability.

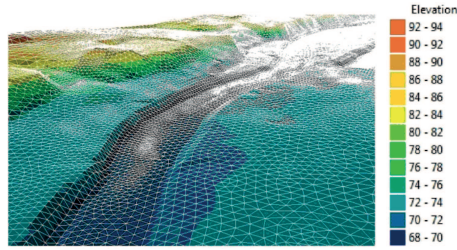
1.2.1 Data-Driven Public Policies

Public policies can greatly benefit from a better understanding of the territory. Figure 1.2 gives examples of applications which we detail in the following.

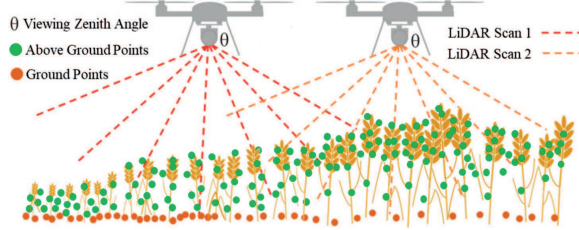
Risk prevention. 3D data can significantly help risk prevention. Detailed 3D maps are useful for identifying areas at risk of flooding [235, 337, 2], landslides [147, 110], or forest fires [217]. Carefully monitoring those areas is crucial to develop early warning systems and emergency response strategies. 3D models of buildings and infrastructure can assist in identifying areas at risk of damage during disasters [379, 333]. Such models can also help estimate the structural integrity of infrastructures with 3D data, and design evacuation plans to save lives and minimize damage [220].

Agriculture. 3D data can be used to monitor crop growth and health, detect pests and diseases, and optimize irrigation and fertilization [265, 21]. Real-world scans can also help to monitor permanent grasslands, groves, and other natural habitats to identify areas at risk of degradation or loss, and limit soil artificialization [45]. 3D acquisitions can also support public services by guiding and monitoring agricultural public policies, such as European’s Common Agricultural Policy (CAP) [65].

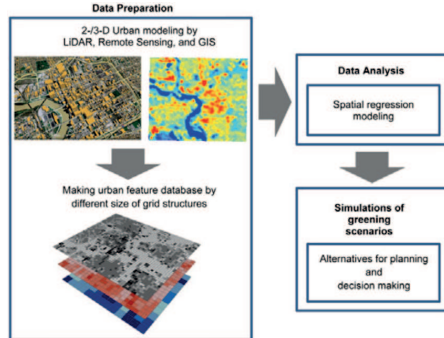
Forestry. 3D data can be used for forest inventory, biomass measurement, biodiversity estimation, habitat cartography, and plant health monitoring [348, 153]. LiDAR data can be used as a surrogate for various ground measurements and can be collected in large areas with less effort than traditional field measurements [108]. 2D aerial and satellite images acquire only part of the real world, as the canopy blocks ground acquisitions. With aerial LiDAR systems acquiring point clouds from above the canopy while still capturing the



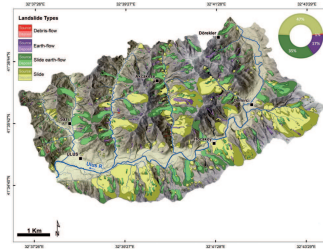
(a) Flood risks estimation.

Figure from [165].

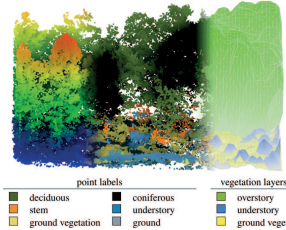
(c) Agriculture monitoring.

Figure from [21].

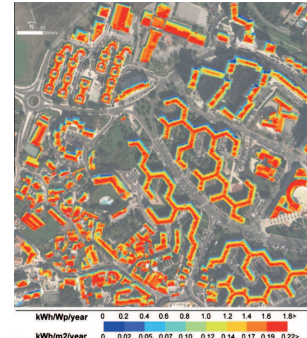
(e) Heat island modelization.

Figure from [60].

(b) Landslide prevention.

Figure from [110].

(d) Forest monitoring.

Figure from [153].

(f) Photovoltaic potential map.

Figure from [40].

Figure 1.2: 3D Data Usage for Public Policies. Public action can benefit from the acquisition and processing of 3D data in various ways, from risk management to environmental monitoring.

ground, analyzing the multilevel data structure expands the possibilities for forest monitoring.

Urban management. Urban management can also benefit from 3D modeling [334]. Digital twins of cities [162], merging 2D and 3D acquisitions with vectorized information on population and circulation for example, can help evaluate urban expansion, provide economic intelligence tools [359, 372], calculate and prevent the risks of heat islands [60, 225].

Energy transition. With the growing need for low-carbon energy, 3D data can also be used in renewable energy production sites. This includes calculating the photovoltaic potential of entire countries with aerial LiDAR acquisitions [202, 40] and simulating the installation of wind turbines and barrages with 3D acquisitions.

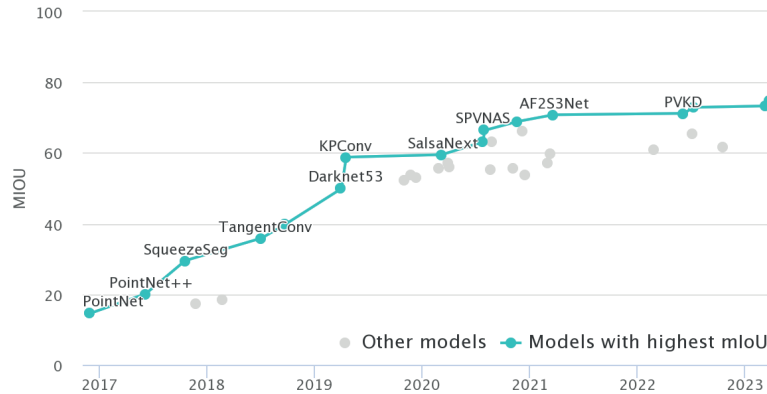
As 3D sensors become ubiquitous, 3D data is becoming increasingly relevant to better understand human activities and their environmental impact. Such applications to public policies are among the key motivations of this thesis, especially for works presented in Chapters 3 and 5.

1.2.2 Efficient and Interpretable Data Processing

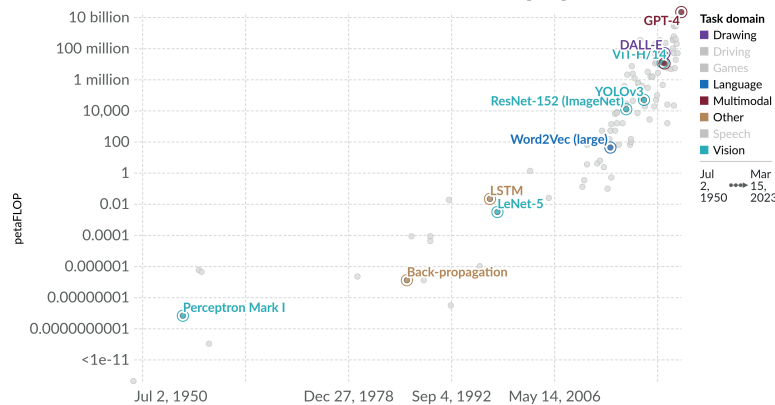
In recent years, data processing capacities have increased alongside the computation and annotation requirements of deep learning methods, leading to a growing interest in computer vision, see Figure 1.3. As a result, it is essential to explore more efficient artificial intelligence methods. In this section, we start by highlighting how current machine learning methods can fail to meet application-specific constraints. We then discuss the concerning increase in energy consumption of artificial intelligence systems. Finally, we explain how public usage is influenced by models’ understanding and acceptability.

Autonomous driving. While self-driving car technology has advanced rapidly in recent years [362, 114], fully autonomous navigation in real-world environments remains beyond the reach of current systems. 3D perception pipelines in autonomous driving must have a low latency—14 meters are traveled in one second at 50km/h speed—and process the data faster than it is acquired to match real-time requirements. To meet this challenge, academic and industry researchers must carefully select the sensors, processing methods, and computing devices used for real-time 3D perception. Recent advances in algorithms’ performance and efficiency [365] have made the use of sensors capturing a vehicle’s surroundings at high frequency viable. Jointly, the development of more efficient hardware has been widely explored. Field Programmable Gate Arrays (FPGAs) helps to handle real-time requirements [261], and LiDAR¹ sensors in-

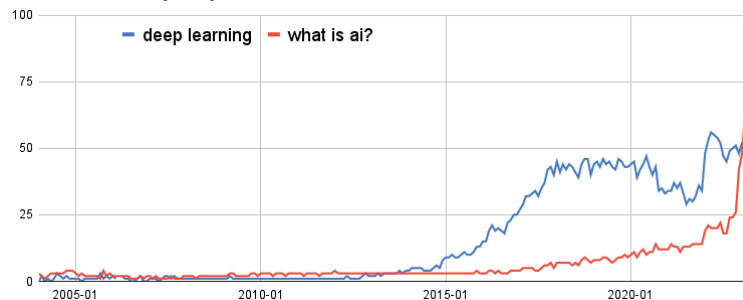
¹“Light Detection And Ranging” (LiDAR) is a remote sensing technology that measures distance by illuminating a target with a laser light and measuring the time of flight of the



(a) The performance of 3D perception algorithm is increasing steadily; here on the widely used large-scale 3D dataset SemanticKITTI [23]. *Figure from [63].*



(b) The computational needs of Artificial Intelligence (AI) systems is increasing exponentially. The models' computational cost increases exponentially during the pre-deep learning era, and recent architectures' computational demand is growing even faster. *Figure from [105].*



(c) The public interest and curiosity for AI and deep learning in particular has reached new levels. We show here the online research interest for terms “Deep learning” and “What is ai?” over the years [109]. Alongside the increasing interest in deep learning, the exponential increase in users who want to know more about AI shows the public's concerns. The “research interest” reflects the proportion of searches for a given keyword in a specific period, compared to the period where the usage rate of this keyword is the highest (value of 100). Therefore, a value of 50 means that the keyword has been used half as often as its peak use.

Figure 1.3: **Global Trends in AI's Efficiency and Interest.** The widespread adoption of AI systems is driven by their increasingly remarkable performance and the availability of massive computing resources. Mitigating the adverse environmental effects of data processing techniques and ensuring public acceptance is key for developing sustainable technologies.

creasingly match the requirements of autonomous driving pipelines [184].

Environmental impact. However impressive, recent deep learning advances have also led to a substantial increase in their computational costs [80, 286, 105]. The energy consumption of deep learning model training and inference raises concerns about their environmental impact. Our aim is to design more frugal AI methods, *i.e.* methods that use less processing power, to help mitigate the adverse effects of energy consumption needs. Measuring the environmental footprint of algorithms is not straightforward, and there is currently no consensus on how to evaluate it. Therefore, users use a variety of proxies, such as Floating-point Operations (FLOPs) per inference, or training or inference time, to assess the efficiency of deep learning methods.

Moreover, working toward models that consume less and toward better evaluations of the method’s resource needs is crucial for artificial intelligence sustainability and acceptability [7, 201].

Acceptability. We believe that computer vision tools should be efficient, but also interpretable and transparent to facilitate their adoption. Smaller and more easily explainable models can increase their acceptability and adoption in fields such as the medical domain [18], for example, and therefore their positive impact. By working towards more interpretable computer vision models, we hope to increase their transparency and make them accessible to a broader range of users.

1.3 Challenges

This thesis focuses on three main challenges: first, the difficulty of handling complex real-world 3D scenes with varying levels of clutter and occlusion; second, the scarcity of annotated datasets; and third, the challenge of interpretable deep learning.

1.3.1 Real-World Scene Complexity

Real-world scenes are often complex. They usually contain objects of multiple scales and varying geometries that can move during acquisition. Such reflected light with a sensor.

scenes are also often captured by sensors with limitations in their acquisition capabilities, such as occlusions, limited resolutions, and measurement errors. The amount of data to be processed, more than 10^6 points/seconds for rotating LiDARs for example, can also be challenging. These limitations complicate data processing and analysis, which require sophisticated algorithms to extract meaningful information. Facing this challenge is critical to understand the complexity of 3D scenes for various applications such as robotics, autonomous driving, and virtual reality. More generally, going beyond toy, synthetic, and object-centric data is crucial to prove useful methods.

1.3.2 Lack of Annotated Datasets

Unlike 2D images, 3D data require complex tools to segment and label individual points or surfaces. This process can be labor-intensive and time-consuming, especially for large datasets or complex scenes. Thus, the scarcity of publicly available annotated 3D datasets limits the ability of researchers to develop and evaluate algorithms. To design more robust evaluation schemes, 3D computer vision research needs more efficient and accurate 3D annotation tools and more diverse annotated datasets for various applications, such as object recognition, scene understanding, and robotics.

Moreover, even if unsupervised techniques can compensate for the lack of annotated datasets, designing deep learning methods that do not rely on large annotated datasets is often challenging.

1.3.3 Deep Learning Interpretability

Building deep interpretable models is a challenging task due to the complexity of models that can have millions of parameters and the lack of transparency in their decision-making process. This lack of interpretability can lead to mistrust in the model's output and make identifying errors difficult. In fields such as healthcare and finance, where decisions based on artificial intelligence are critical, this lack of transparency can pose legal and ethical challenges. To address these challenges, it is essential to develop new techniques that can provide insight into the inner workings of models. While difficult to design, these techniques include visualization tools and the creation of explainable frameworks that provide transparency and accountability in the decision-making process.

By developing more interpretable models, we can improve the reliability and acceptability of deep learning models.

1.4 Contributions

In this thesis, we present two contributions to enhance the efficiency and interpretability of real-world 3D data analysis.

Our first contribution is an architecture specifically designed to process rotating LiDAR acquisitions in real time. Our model takes into account the geometry of the sensor, and achieves *state-of-the-art* semantic segmentation performance on LiDAR sequences, while reducing the latency by more than five times and the number of parameters by fifty times compared to existing approaches.

Our second contribution is an unsupervised and interpretable pipeline for processing 3D point clouds. A key feature of our method is learning data representations directly in 3D space. This allows us to visualize, manipulate, and annotate the learned shapes, contributing to the interpretability of the network’s reasoning. Our method unfolds into two main aspects. First, it facilitates structuring, analyzing, and visualizing large shape collections. Second, it can parse large real-world 3D scans composed of recurrent objects by discovering meaningful and viewable prototypical shapes.

In addition to these two technical contributions, we also provide two open-source datasets. Indeed, the lack of diversity in the datasets used to design and test the current *state-of-the-art* methods, hampers the effectiveness of 3D computer vision pipelines. For example, autonomous driving datasets are often acquired only in suburban areas [23, 186], and ignore dense urban areas, historical centers, and natural areas. Increasing the diversity of datasets is essential to ensure the generalization of algorithms to a broader range of environments. In this thesis, we introduce two open-source annotated datasets acquired in the French territory:

- HelixNet [194], the largest open-source dataset of rotating LiDAR scans with dense annotations, which provides point-level sensor metadata crucial for precisely assessing real-time semantic segmentation methods for autonomous driving,

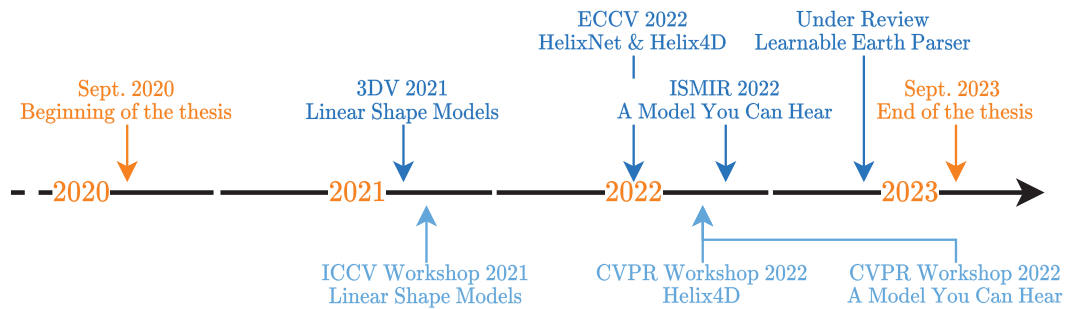


Figure 1.4: **Thesis Timeline.** From 2020 to 2023, we wrote four articles and presented three of them in international conferences and workshops.

- and the Earth Parser Dataset [197], which consists of seven large aerial LiDAR scans, and can be used to evaluate the performances of real-world 3D processing techniques in diverse environments including cities, crop fields, a power plant, forests, and a marina.

Both datasets are part of wider programs deployed by the Institut national de l’information géographique et forestière — French Mapping Agency (IGN), aiming at describing the French territory for public policies: Stereopolis [248] and LiDAR-HD [142]², respectively. Our goal in annotating, curating, and sharing parts of these datasets is to enable the computer vision community to easily evaluate the robustness of 3D analysis methods through diverse and challenging settings.

1.5 Publications and Research Activities

The manuscript presents three main projects, two of which have been published in international conferences and workshops, while the third is under review for an upcoming international computer vision conference. In addition, an extension of our work from three-dimensional data to audio data has been presented at an international Music Information Retrieval (MIR) conference and an international computer vision conference workshop. Figure 1.4 shows a timeline of the thesis.

The implementations accompanying the projects have been released in

²The French national program LiDAR-HD [142] deployed by the IGN aims at acquiring and diffusing a 3D cartography of French territory. The dense point cloud will be acquired by aerial LiDAR between 2020 and 2024, containing at least 20 points per square meter, or 10^{13} points in total.

open-source³ and interactive visualizations of the results have been made available on dedicated project pages⁴. The proposed datasets have been published on Zenodo and have already been downloaded over 150 times, with more than 7TB of downloaded data.

I also contributed to the research community by presenting my work to several research teams (INRIA team Willow, ISAS-CEA, Centre Borelli, ENAC-EPFL), groups (GDR ISIS, ISPRS Congress), public administrations (French Ministry of Ecology, IGN, BKG), and private companies (SAMP.ai, Deezer Research). Additionally, I have been actively involved in the research activities of the teams I was part of, including organizing research meetings regularly in the LASTIG, teaching computer vision lessons at the Ecole des Ponts ParisTech (15h) and the Ecole Nationale des Sciences Géographiques (25h), and organizing a 3D deep learning for remote sensing tutorial at ISPRS 2022 Congress.

The contributions of the thesis can be summarized as follows:

Publications at International Conferences

- 📖 Romain Loiseau, Mathieu Aubry, Loic Landrieu, “Online Semantic Segmentation of LiDAR Sequences: Dataset and Algorithm”, ECCV, 2022.
- 📖 Romain Loiseau, Baptiste Bouvier, Yann Teytaut, Elliot Vincent, Mathieu Aubry, Loic Landrieu, “A Model You Can Hear: Audio Identification With Playable Prototypes”, ISMIR, 2022.
- 📖 Romain Loiseau, Tom Monnier, Mathieu Aubry, Loic Landrieu, “Representing Shape Collections with Alignment-Aware Linear Models”, 3DV, 2021.

Publications at International Workshops


- 📖 Romain Loiseau, Mathieu Aubry, Loic Landrieu, “Helix4D: Online Semantic Segmentation of LiDAR Sequences”, CVPR Transformer for Vision Workshop, 2022.
- 📖 Romain Loiseau, Baptiste Bouvier, Yann Teytaut, Elliot Vincent, Mathieu Aubry, Loic Landrieu, “A Model You Can Hear: Audio Classification With Playable Prototypes”, CVPR Sight and Sound Workshop, 2022.

³<https://github.com/romainloiseau>







⁴<https://romainloiseau.fr>

-  [Romain Loiseau](#), Tom Monnier, Mathieu Aubry, Loic Landrieu, “Representing Shape Collections with Alignment-Aware Linear Models”, ICCV Learning 3D Representations for Shape and Appearance Workshop, 2021.




Publications Under Review

-  [Romain Loiseau](#), Elliot Vincent, Mathieu Aubry, Loic Landrieu, “Learnable Earth Parser: Discovering 3D Prototypes in Aerial Scans”, arXiv, 2023.

Open-Source Implementations

| | |
|--|------|
|  romainloiseau/EarthParserDataset | 19 ★ |
|  romainloiseau/LearnableEarthParser | 25 ★ |
|  romainloiseau/a-model-you-can-hear | 33 ★ |
|  romainloiseau/HelixNet | 39 ★ |
|  romainloiseau/Helix4D | 47 ★ |
|  romainloiseau/deep-linear-shapes | 28 ★ |

Datasets

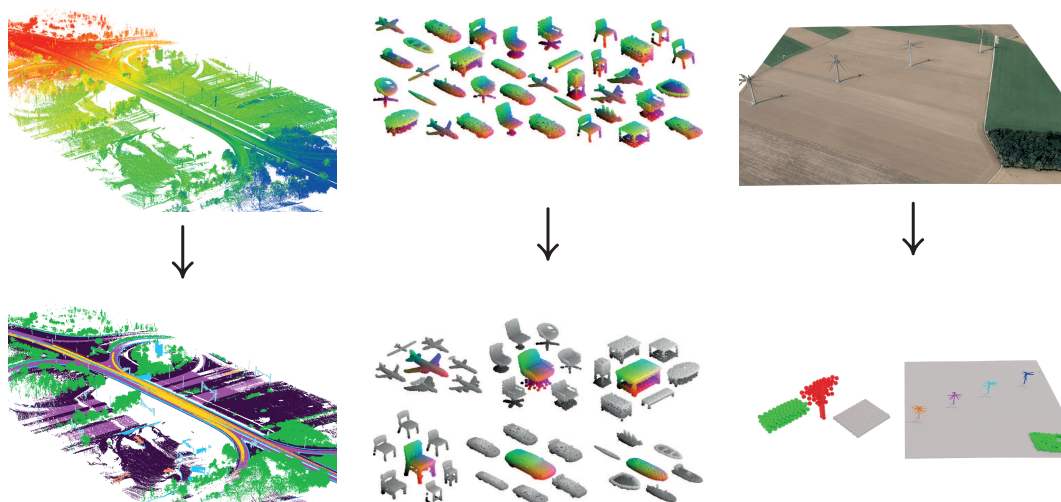
-  [Earth Parser Dataset](#): 7 aerial LiDAR scans, covering 7.7km² with annotations in diverse urban and natural environment.
-  [HelixNet](#): 10B annotated 3D points captured by a moving vehicle for autonomous driving. 568 

1.6 Thesis Outline

As shown in Figure 1.5, this thesis is structured as follows:

Chapter 2. Related Work. We present an overview of 3D data representations, tasks, methods, and datasets commonly used in the field.

Chapter 3. Online Segmentation of LiDAR Sequences. We propose a novel 3D point cloud semantic segmentation approach that takes advantage of the geometry of the LiDAR sensor to achieve real-time inference while achieving *state-of-the-art* segmentation performances. We also introduce an open-source annotated dataset to evaluate and design real-time segmentation methods.



(a) Chapter 3. Semantic segmentation of embarked LiDAR acquisitions.

(b) Chapter 4. Shape discovery in large unordered 3D collections.

(c) Chapter 5. Understanding real-world uncurated aerial LiDAR scans.

Figure 1.5: **Thesis Outline.** This thesis discusses 3D data processing from semantic segmentation to clustering and shape discovery.

Chapter 4. Exploring Shape Collections. We propose learning alignment-aware viewable and deformable prototypes to gain insights into large unordered collections of 3D shapes in an unsupervised setting. We demonstrate *state-of-the-art* results for shape clustering and few-shot semantic segmentation.

Chapter 5. Discovering Prototypes in Aerial Scans. We extend the 3D prototype learning framework introduced in Chapter 4 to large-scale real-world 3D aerial acquisitions. We also introduce an open-source aerial LiDAR dataset acquired in various environments, such as forests, urban areas, and marinas.

Chapter 6. Conclusion. We summarize the contributions of our work. We also propose future research directions toward efficient and interpretable real-world 3D data processing.

Chapter 2

Related Work

Recent advances in 3D data collection and processing have benefited many applications, such as self-driving cars, territory management, and medical data analysis. This chapter presents some of the key concepts and challenges of 3D data processing. We discuss the different types of 3D data representations, present some of the main tasks and approaches to solve them, and describe some of the most predominant 3D datasets.

2.1 3D Data Representation

In this section, we discuss different ways to represent 3D data, from raw data representations to handcrafted and learned descriptors.

2.1.1 Geometric Representation

3D data can be encoded in many different manners, such as point clouds, meshes, voxels-grids, implicit functions, and primitives-based representations. Figure 2.1 gives examples of different types of representations.

Point clouds. 3D point clouds are a popular representation of 3D objects or scenes and consist of a set of points in 3D space. Each point is associated with its 3D coordinates and may also include attributes such as color, normal, and intensity. Point clouds are typically obtained with laser sensors or photogrammetry. Here, we present the most common ways to produce point clouds.

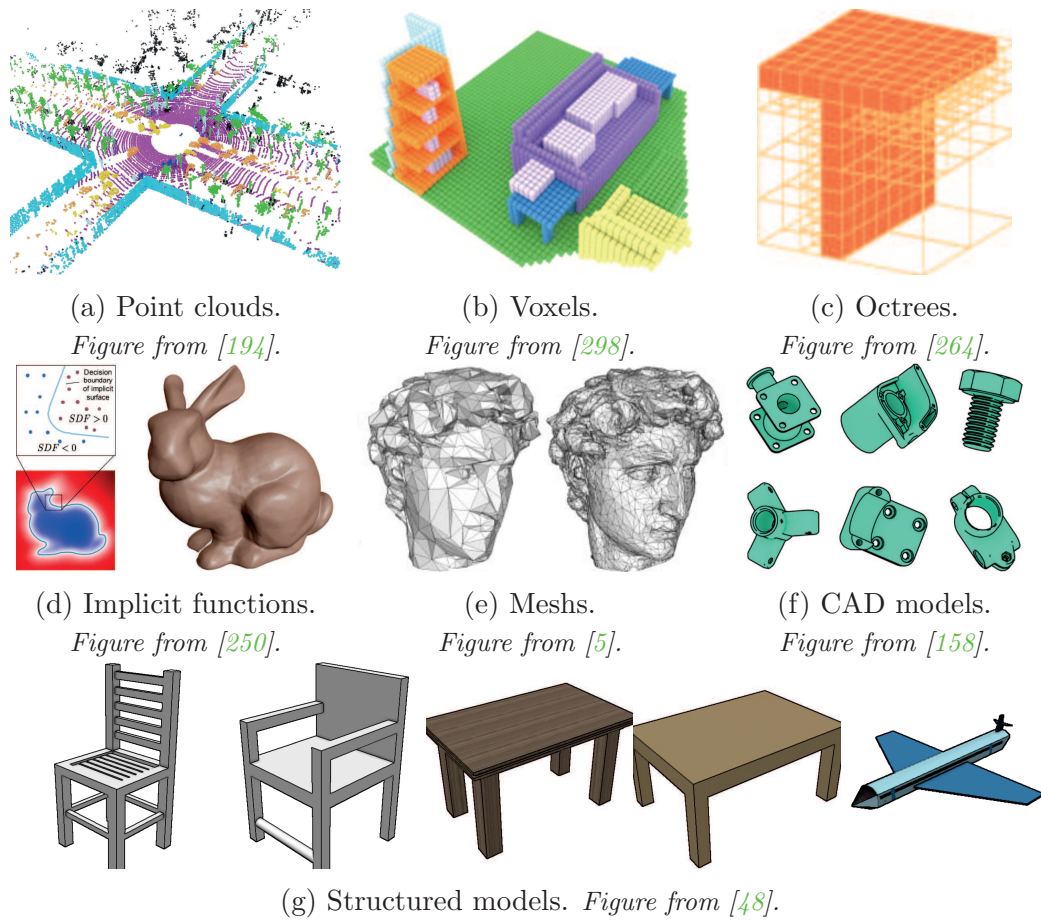


Figure 2.1: **3D Geometric Representations.** 3D data can be represented in various ways.

- **LiDAR sensors** acquire point clouds directly through light Time-of-Flight (ToF) measurement. They acquire precise point clouds, but due to their relatively high price point, their usage was reserved for long time for specific limited applications [178]. As the size and cost of LiDARs decrease and their reliability increases, the popularity of these sensors grows steadily. It is now possible to scan entire countries using airborne LiDARs [238, 142], smartphone manufacturers are equipping their smartphones with LiDAR sensors [8], and automotive suppliers are designing LiDAR sensors dedicated to autonomous driving [323].
- **Depth cameras** acquire point clouds by emitting and analyzing infrared light. They use structured light emitters and 1D or 2D arrays of photodetectors to obtain a depth image [136]. Their high spatial and temporal resolution makes them suitable for 3D data acquisitions [169]. However, because they use infrared light emitters, they are specifically suited for

indoor rather than outdoor applications, in which the sunlight prevents their use.

- **Sets of images** can also be used to generate point clouds [243]. Such methods often rely on matching pixels from images whose camera positions and orientations are known or estimated to deduce the corresponding 3D point position. Keypoint matching can be achieved by comparing handcrafted image descriptors such as Scale-Invariant Feature Transform (SIFT) [199] or Speeded-Up Robust Features (SURF) [22], or learned image features [312, 226, 203, 81, 204, 290]. The main advantage of these methods is that they rely on simple camera acquisitions without the need for specialized equipment. However, the resulting point cloud may lack precision and detail [34]. Reconstructing specular materials can also result in inaccurate 3D reconstructions due to light reflection [143, 271, 3, 278].

The use of 3D point clouds has shown promising results for 3D scene and object analysis. During the past decade, point clouds have been proven to be particularly suited to deep learning-based approaches. Neural networks have been designed to process unordered point sets [257, 256], to leverage graph neural networks to model local relationships [335], and to perform new learning-based operations to improve point cloud processing [183, 311].

However, a notable limitation is the absence of structured connectivity information within point clouds, which significantly hampers the computational efficiency of local feature extraction for large point clouds. Additionally, point clouds inherently lack the ability to represent surfaces accurately, as achieving infinite resolution becomes computationally infeasible. They also fail to provide valuable insights into the interior and exterior characteristics of objects. Moreover, they can have variable local densities and are incompatible with physical simulation. Despite these formidable obstacles, researchers have demonstrated the utility of point clouds in various computer vision tasks.

Meshes. Meshes are commonly used to represent a 3D surface. They consist of connected polygons, typically triangles. They are suitable for 3D rendering, as they are easy to manipulate for relighting and simulation, and compatible with modern graphics pipelines. 3D meshes can be generated from point

clouds using algorithms such as Delaunay triangulation [75, 86, 215], from multiple views [133, 321], or from implicit functions using the marching cubes algorithm [198]. Vertices can be associated with information such as normals or colors. Textures can also be applied using UV maps, which provide a two-dimensional representation of the 3D surface. Meshes are well suited for many applications, from industrial design to large-scale modeling, as they can satisfy properties such as water-tightness or closeness under certain constraints.

However, creating a high-quality mesh from a noisy point cloud can be challenging [228, 291, 212]. Recent work has explored learning-based approaches to address this problem by directly regressing meshes from point clouds [115, 353] or images [107]. Mesh-based representations can also be memory intensive to manipulate, especially for high-resolution models. They can also suffer from topological deficiencies such as non-manifold edges or vertices [84].

Voxels. Akin to 2D pixel representations, 3D voxels are a widely used 3D representation in which the 3D space is partitioned into a regular grid of small cubes. Each voxel usually stores an occupancy value. Voxels can also be associated with other properties, such as color or semantic labels. Because convolutions are naturally defined for voxels, several computer vision tasks use this representation.

However, the cubic complexity of this representation makes the processing computationally expensive. This limits the resolution of the processed 3D shape and makes voxel representations unsuitable for large-scale scenes. Indeed, at high-resolution, scanned surfaces represent a zero-measure set of occupied voxels. To address this problem, more efficient data structures have been proposed, such as octrees [218]. Octrees provide a flexible and hierarchical representation of the 3D space by dividing it into octants, where each octant’s resolution adapts to the data’s local density. Using octrees leads to a more efficient representation of 3D data and reduces the computational cost of 3D voxel-based methods, making them more practical for real-world applications [264, 358, 138].

Implicit functions. Implicit functions have emerged as a powerful alternative to voxel-based representations for 3D shape modeling and reconstruction. They provide a continuous representation of a shape by mapping the 3D space

to an occupancy value or a distance—possibly signed—to the closest shape boundary. Such continuous functions allow for infinite and adaptive resolution, and can be approximated with neural networks. They can be used for 3D shape reconstruction and completion from camera observations [250, 56, 222, 103] and from a variety of inputs, such as low and high-resolution voxels, sparse and dense point clouds, and complete or incomplete shapes [58, 55]. Subsequently, they have been used to capture local 3D geometric details [54], animate people in clothing based on body pose [314], and generalize to shapes with unseen 3D transformations [259].

However, their adoption has been limited by their high computational cost, particularly for high-resolution inputs. Moreover, they are often used for 3D generation or rendering, and their applications to 3D analysis are often limited.

Structured models. Simple parametric shapes can be combined to represent complex 3D objects. They serve as building blocks for more complex 3D models, providing high-level information on the underlying shapes. Using parametric shapes to represent objects or scenes can provide high-level structural information to represent digitized 3D data. Extracting a more compact shape representation with structured models can be used for shape analysis, segmentation, and synthesis [251, 253, 236]. Moreover, primitive-based representations can facilitate efficient and scalable 3D generative modeling [181, 176]. However, those models are mostly useful for getting shape representations, and can have difficulties modeling details at varying scales.

The most common structured models for modeling shapes with varying levels of detail are Computer-Aided Design (CAD) models. CADs typically comprise both geometric shapes and parameterized constraints that support the design process. CAD models has significantly impacted the design process of manufactured objects by making them accessible.

This thesis primarily concentrates on real-world LiDAR acquisitions, thus placing a major emphasis on point clouds. However, depending on the use cases, we explored a range of 3D geometric representations in the development of our methods. Although the geometry of objects and scenes carries informative cues, achieving automated analysis requires discriminative descriptors.

2.1.2 Classical Descriptors

Understanding 3D data structures begins with characterizing shapes. To achieve accurate results in subsequent tasks, shape descriptors must be discriminative, robust, and affordable to compute. 3D descriptors can be based on local, spectral, or scene-related features. Here, we focus on local descriptors that provide comprehensive information about each point, voxel, or region within a 3D scene or object. It is worth noting that the same principles apply to global descriptors, which describe complete 3D objects.

Local descriptors. Various features can be derived from the local geometry of a point cloud or a mesh.

- **Planarity, linearity, and verticality** can be computed from the eigenvalues of a point neighborhood covariance matrix [254, 342, 117]. These features can provide more detailed information about the local geometry of a point cloud by varying [31] or optimizing [76] the neighboring scales.
- **Surface normals'** directions and orientations convey essential information about the underlying object or scene. Mesh-based representations provide local surface normal directions for each face, but their orientation may suffer from uncertainties regarding the interior/exterior boundaries of the scene. Alternatively, surface normals can be derived by computing the gradient of an implicit function on the surface of an object [318]. The normal direction of 3D points can be estimated by performing Principal Component Analysis (PCA) on the relative position of its neighbors. However, estimating point-cloud normals remains challenging due to noise or ambiguity in the orientation [160, 282, 223].
- **Histogram-based** descriptors have also proven to be useful for describing 3D shapes. For point clouds with normals, Spin image [150, 151] computes a 2D histogram of local point density in a polar coordinate system tangent to the object's surface. First designed for images, the shape context descriptor [24] was extended to meshes with histograms in geodesic polar coordinates [159]. MeshHOG [363] computes a histogram of gradients of a mesh texture. 3-dimensional Histograms of Oriented Gradients (HOG) have also been used in various computer vi-

sion tasks [281, 72, 170, 1]. Additionally, non-spherical neighborhoods can be used to encode features such as height distributions [229].

Spectral descriptors. The first spectral descriptors used the eigenvalues and eigenfunctions of the Laplace-Beltrami operator on a surface. Being robust to near-isometric transformations of the input, they are suitable for describing non-rigid shapes. The Global Point Signature (GPS) [275] proposes to describe a point by evaluating the set of normalized eigenfunctions sorted according to their eigenvalues. As small modifications of the shape can change the order of eigenfunctions, the GPS can only be used to compare two shapes that are identical with respect to an isometric deformation. The Heat Kernel Signature (HKS) [302] describes shapes with heat diffusion equations, by measuring the temperature of a point after heating only this point for a given time. HKS has been extended to compute scale-invariant features to better represent non-rigid 3D models [41] and to other filter families such as the Schrödinger wave equation with the Wave Kernel Signature (WKS) [14].

Spectral descriptors have proven to be effective for shape matching with functional maps [242]. Functional maps optimize correspondences between the eigenfunctions of the Laplace-Beltrami operator for two shapes and ensure that the maps preserve descriptors across deformed shapes.

Other descriptors describe a 3D location with respect to an entire scene. For example, using the elevation, or position with respect to the road, can give great insight for understanding outdoor scenes. These handcrafted features are often specific to the application for which they are designed.

Given those 3D descriptors, deciding the one to use depends on the input data and the specific application [338, 233, 123, 124].

2.1.3 Deep Learning Features

Neural networks successfully tackled numerous challenges in 3D shape analysis. Deep learning features have been successfully used for tasks as diverse as classification [207], segmentation [311], shape generation [115, 310], matching [244], denoising [132] and compression [138]. Deep learning pipelines often directly output the final prediction from a raw input, making the notion of

descriptor poorly defined. However, deep learning-based methods typically comprise an encoder and a decoder. The encoder is often a set of operations designed to extract low-resolution high-level information about objects or scenes, while the decoder produces the final output. Here and in Figure 2.2, we present the deep learning techniques that are most relevant to this thesis.

3D convolutions. Convolutional Neural Networks (CNNs) have revolutionized 2D image analysis [161], but their application to 3D data has first been limited by their computational complexity.

- **Discrete convolutions.** The direct approach of using 3D convolutions on voxelized representations of real-world data is inefficient, due to the large amount of their empty voxels leading to the computation of useless operations. However, implementation efforts to perform convolutions on sparse data structures such as MinkowskiNet [59], SparseConvNet [112], and SPVConv [307] have emerged as promising solutions to this problem, enabling efficient discrete convolution.
- **Continuous convolutions.** KPConv [311] and ConvPoint [37] leverage local information while avoiding suffering from discrete convolution computational drawbacks. These continuous formulations of convolutions operate on point clouds without intermediate representation. They learn convolution weights along with the position of kernel points regarding center points, thus producing a continuous generalization of convolutions that does not rely on grids, see Figure 2.2b.
- **Convolutions on non-Euclidian spaces.** Convolutions applied directly on the surface of an object have the advantage of being invariant to isometric deformations, deformations that preserve geodesic distances on the shape. This is particularly useful when analyzing shapes that can be deformed, such as human bodies. Geodesic-CNNs (GCNNs) generalize CNNs on non-Euclidean manifolds. GCNNs create local patches with polar coordinates defined along the surface geodesics, see Figure 2.2c. Then, local filters are applied on these patches to extract discriminant features [213]. However, the resulting features are invariant with respect to the orientation of the patch, leading to losing directional information. Multi-Directional-GCNNs [255] improves on this work by keeping

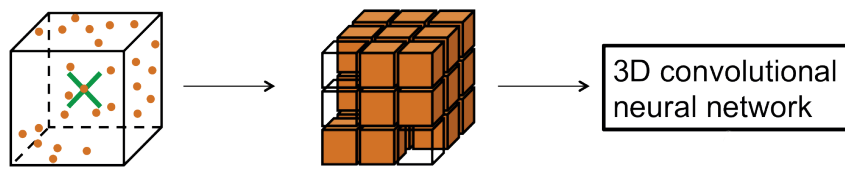
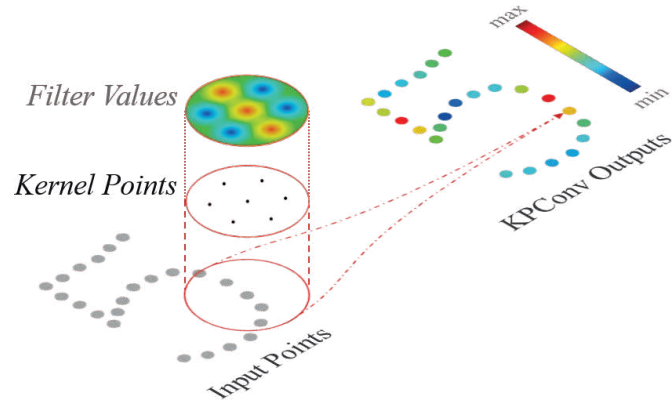
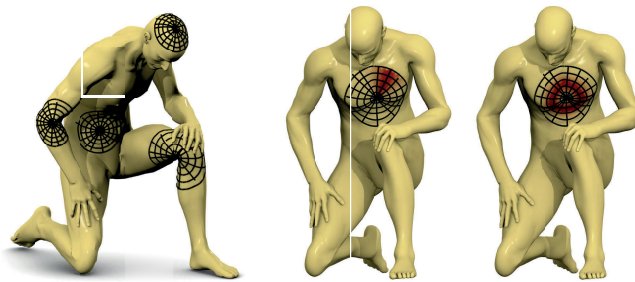
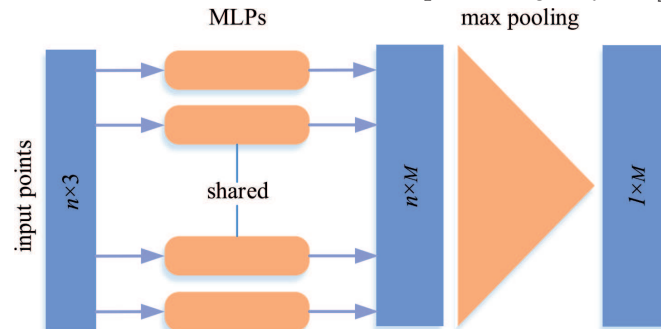
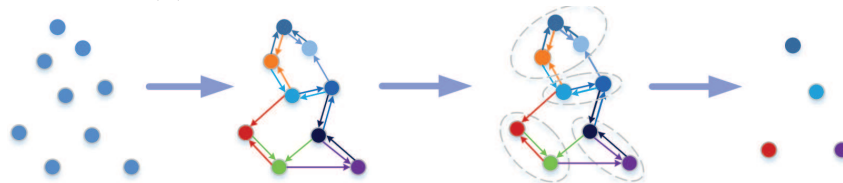
(a) Discrete convolutions. *Figure from [298].*(b) Continuous convolutions. *Figure from [311].*(c) Convolutions on non-Euclidian spaces. *Figure from [213].*(d) Pooling-based features. *Figure from [125].*(e) Graph-based networks. *Figure from [125].*

Figure 2.2: **3D Deep Learning Paradigms.** Choosing among the deep learning-based 3D processing methods depends on each specific input data and application.

patches directional information until the last layer of the network, leading to more discriminative and robust features. Anisotropic-CNNs (ACNNs) [35] uses anisotropic heat kernels, see Section 2.1.2, to extract local mesh features for shape correspondences. While GCNN and ACNN use fixed patches with polar coordinates or heat kernels, mixture model networks [232] generalize convolutions on manifolds and graphs by directly learning the coordinate system of extracted patches. MeshCNN [130] jointly uses convolutions on mesh edges while learning which edges to collapse for classification and segmentation.

Pooling-based features. PointNet [257] proposes to learn 3D representations by applying pointwise Multi-Layer Perceptrons (MLPs) and pooling operations on unordered point sets. The key idea is that unordered sets require operations invariant to the orders of points. They applied point-level linear layers and object-level pooling to learn 3D representation. However, they do not generalize to complex scenes because they do not exploit local 3D structures. Hierarchical representations [256] addressed this issue by learning and aggregating features on different scales. DeepSets [364] generalize neural networks on sets for a variety of modalities, including point clouds, with the same permutation-invariant principle.

Graph convolutions. Graph neural networks have also been used to analyze 3D point clouds. Message passing over graph edges [106] is well suited for 3D data and has been widely used on meshes [328, 224] and point clouds [292, 258, 335]. Graph neural networks applied to superpoint representations [249, 117, 97, 188, 167] also enable fast and accurate semantic segmentation on large-scale 3D point clouds [168].

More recently, attention mechanisms [326] have been leveraged for point clouds with 3D transformer networks, and achieved *state-of-the-art* performances for various tasks [227, 373, 122, 164, 260]. Indeed, those transformer architectures are a way to learn to generate dynamic graphs.

Capsule networks. Capsule networks [134] group neurons into “capsules” that represent objects or parts of objects. They were first developed to detect and link parts of objects in images [276]. The 3D capsule approach [375]

explicitly tries to learn shape representations invariant to 3D transformations and can be applied to many tasks. 3D capsules have been extended to learn spatial relationships [341], to be equivariant to rotations [376], and to model non-rigid 3D shapes [377].

Deep functional maps. Deep functional maps learn descriptors for performing shape matching with functional map [242]. In particular, FMNet [189] trains a siamese network from ground truth dense correspondences. Other approaches train deep functional maps to learn descriptors without supervision by minimizing the distortion of geodesic distances [129] or heat kernels [15]. Deep functional maps have also been used for semantic segmentation [356, 12] and partial shape matching [287].

2.2 3D Tasks

In this section, we discuss the 3D computer vision tasks that are most pertinent for our research, see Figure 2.3. We give an overview of classification, clustering, semantic segmentation, and primitive fitting. We focus mainly on deep learning approaches, which are the most relevant to our work.

2.2.1 Shape Classification

The goal of shape classification is to assign a class to 3D objects regardless of their pose or deformation, see Figure 2.3a. Early attempts at point cloud classification demonstrated promising but limited performance [46]. They relied on extracting handcrafted features from 3D neighborhoods given as input to discriminative learning algorithms [180, 366]. However, a new set of methods based on deep learning emerged with the growing computing and acquisitions' capabilities. At first, the use of discrete 3D CNNs was promising but computationally heavy [214, 99, 128]. PointNet [257] and its hierarchical version PointNet++ [256] showed significant improvements by operating directly on point clouds by using permutation invariant pooling operators. Point convolutions [183, 13, 311], graph convolutions [335], discrete convolutions on voxels [298, 264, 59, 112], and self-organizing network [173] have also been explored for shape classification. More recently, transformer networks

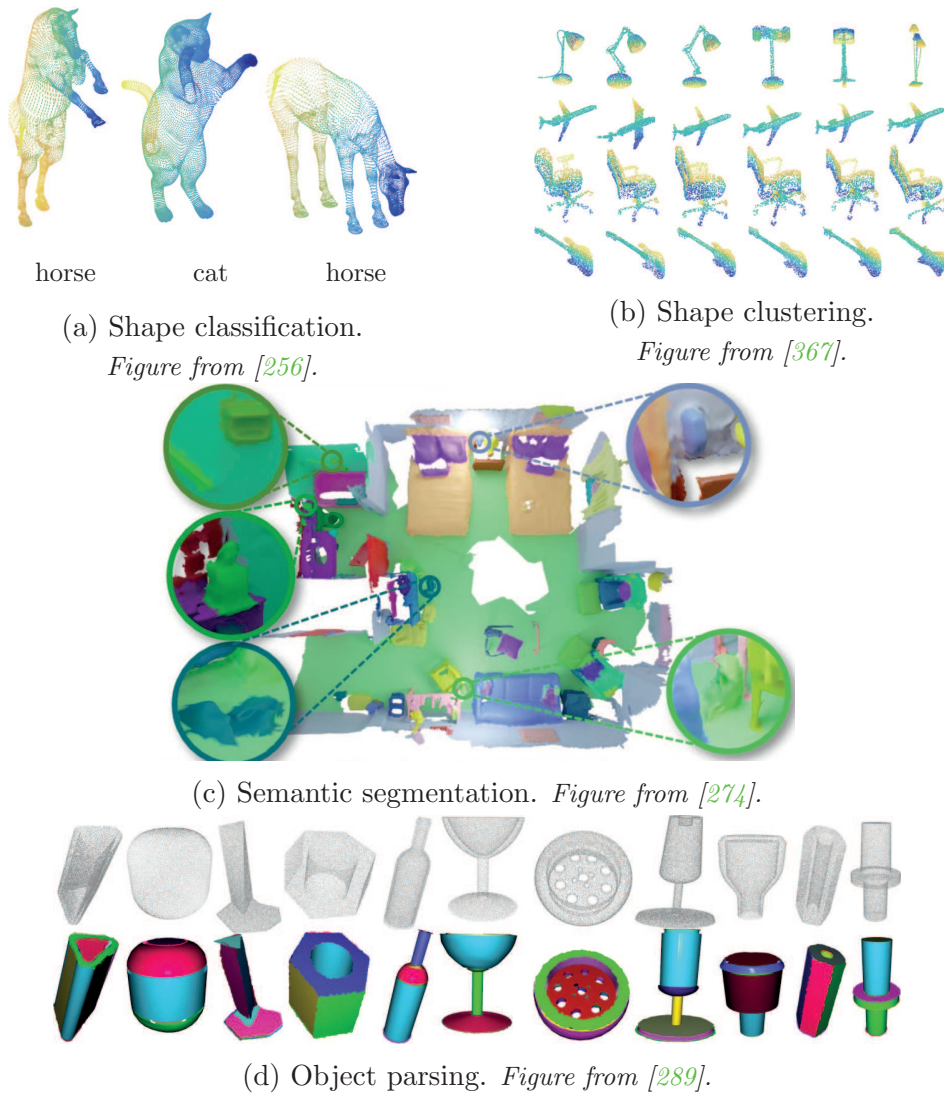


Figure 2.3: **3D Tasks.** Our work is related to several computer vision tasks aiming at understanding 3D data.

have emerged as the new paradigm for shape classification [373, 122]. Finally, shape classification is often used on synthetic shape collections as a pretext task to demonstrate the robustness of developed methods.

2.2.2 Shape Clustering

Clustering shapes amounts at organizing 3D models into semantically meaningful groups. It is challenging because of the potential objects' deformation and misalignment. Methods based on handcrafted descriptors [200, 366] or auto-encoders [313, 269] trained using a reconstruction loss [353, 237, 382] often failed to effectively cluster 3D models.

Self-supervised features. Recently, to answer the difficulty of annotating 3D data, self-supervised learning techniques have shown promising results for pre-training models. Some methods learn features through reconstruction [353, 131, 279]. Other works learn 3D features using contrastive learning across different representations [370]. Inspired by text and image masked autoencoders, some recent methods pre-train point-cloud-based architectures via occlusion completion [247, 351, 331, 361, 352].

Self-supervised methods have shown promising success in shape clustering by using descriptors invariant to transformations [367, 277]. To better process unaligned shapes collections, rotation-invariant representations have also been explored [357, 52, 371, 179].

2.2.3 Semantic Segmentation

While shape classification and clustering expect to predict a single class for a shape, 3D semantic segmentation consists in predicting dense labels for an object or scene. It is a crucial part of understanding one’s surroundings, and hence has many applications in robotics, autonomous driving, and augmented reality. Pre-deep learning methods used discriminative algorithms to map handcrafted point descriptors to class predictions [339] with Gaussian Mixture Models [166], Support Vector Machines [285] or Random Forests [51]. Deep learning methods have been the dominant paradigm in the past decade. U-Net-style [270] neural networks allow one to predict dense labels with any feature extraction framework. First designed for medical images [270] and volumes [61], the decoder combines both upsampled decoded features and high-resolution features from intermediate encoder layers, see Figure 2.4.

Using this architecture on 3D point clouds, PointNet++ [256] pioneered deep learning indoor semantic segmentation on ScanNet [73]. For large-scale scenes, 2D projections of 3D can be segmented with 2D convolutional neural networks and reprojected back to the point cloud [38]. Moreover, local 2D representations of 3D point clouds could benefit from local geometry [300, 309]. Convolutions in their continuous [311] or sparse [112, 59] formulation have also been shown to be robust and versatile building blocks for semantic segmentation pipelines. Landrieu *et al.* [168] pools points into geometrically homogeneous superpoints to efficiently process large-scale scenes. Despite the

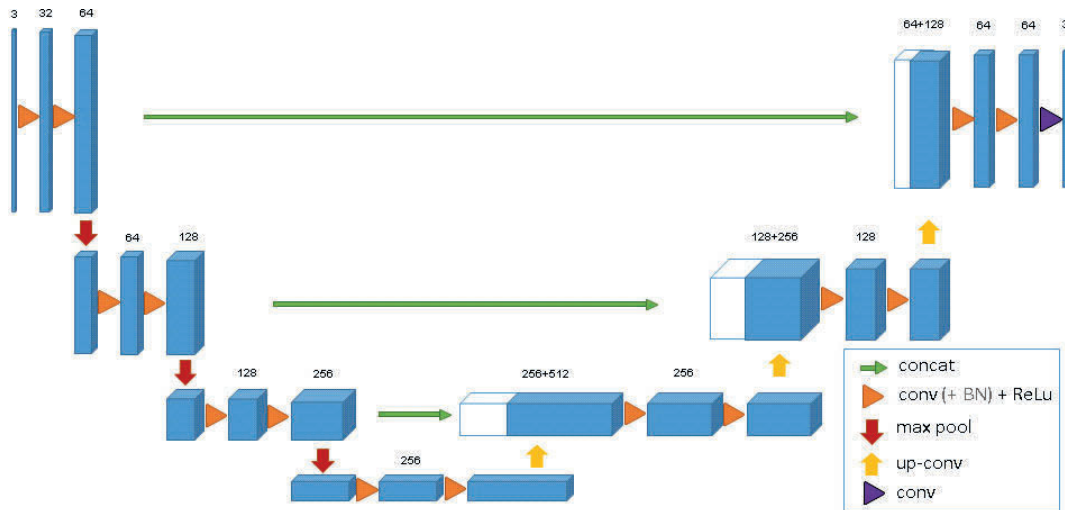


Figure 2.4: **U-Net Architecture.** U-Net-style networks extract features of decreasing spatial resolution with a sequence of convolutional and down-sampling layers. The resulting abstract low-resolution feature representation is then upsampled with unpooling operations, while simultaneously preserving high-resolution details by associating the encoded features at each intermediate resolution. *Figure from [61].*

computational complexity of attention mechanisms, transformers have recently shown great performances [373, 122, 354]. Additionally, while some methods rely solely on point clouds, DeepViewAgg [267] merges 2D and 3D features with an attention mechanism for point clouds’ semantic segmentation.

Although published in a short period of time, the literature was quickly adapted for specific applications. For example, semantic segmentation on LiDAR datasets for autonomous driving has been addressed using range images [185, 304, 69] and Bird-Eye-View (BEV) projections [368]. Sparse convolutions [113, 59] have also facilitated working with discrete convolutions in Cartesian [307, 57] or cylindrical [381, 135] coordinates.

Few-shot segmentation. Methods that can perform segmentation while being trained on very few annotated samples have gained attention due to their potential to reduce annotation costs and handle new classes. However, the task is challenging and requires leveraging prior knowledge and incorporating effective transfer learning strategies. Gadelha *et al.* [95] extract shape regions by learning convex decompositions of 3D shapes. Groueix *et al.* [116] and Wang *et al.* [332] propagate annotations through shape matching by learning to deform 3D objects. More recent approaches use learned 3D features to

match 3D models in larger scenes [374]. Furthermore, few-shot 3D semantic segmentation may benefit significantly from the advancing abilities of self-supervised networks to learn meaningful 3D features.

2.2.4 Primitive Based Analysis

Primitive-based point cloud decomposition. Modeling shapes as a set of primitives such as blocks [268], cylinders [28] or superquadrics [20] has a rich history in vision and graphics [152]. Classical applications include reverse engineering [25], shape completion [283, 305] and editing [96]. A variety of methods have been developed to find primitives in unstructured 3D scenes, including seed growing techniques [171, 172], genetic algorithms [53], RANSAC [89]-based technique [111, 284, 182, 262], and probabilistic methods [190, 346].

Primitive discovery with deep learning. Deep learning has recently become the dominant paradigm for primitive discovery. For such task, supervised methods [384, 175] are limited by the availability of annotated datasets. Following the seminal work of Tulsiani *et al.* [317], some unsupervised approaches only rely on a reconstruction loss to learn primitive decomposition [252, 253]. Even if these methods provide accurate shape decomposition, they are often limited to a restricted set of parametric shapes and evaluated on synthetic datasets.

The number of primitives used for scene or object decomposition determine a tradeoff between compacity and precision. This has been explored with recurrent networks [384, 174, 288], capsule networks [375], reinforcement learning strategies [317] or probabilistic models [252]. Primitive discovery also relates to approaches that consists in learning prototypical shapes directly instead of being restricted to a set of parametric primitives [78].

However, most of the above methods are trained and evaluated on well-curated synthetic datasets, such as ModelNet [347], ShapeNet [48], or D-FAUST [33]. They are typically evaluated on instances of known classes, which questions their robustness to real-world data.

2.3 3D Point Cloud Datasets

The 3D computer vision literature considers a variety of synthetic and real-world datasets. In this work, we focus on large-scale 3D point clouds acquired in realistic settings. We consider, for example, unstructured collections of 3D models and large aerial or terrestrial LiDAR acquisitions; see Figure 2.5. We present in this section the most relevant datasets for this thesis, and therefore we do not discuss all types of 3D datasets such as indoor RGB-D datasets [350, 297, 9, 73, 49], datasets of scanned objects [322, 345] or datasets for human motion analysis [6, 146, 32, 206, 29].

We first introduce the main object-centric synthetic datasets employed in the computer vision community. Second, we describe ways to simulate large-scale real-world datasets from existing 3D models. Third, we describe some of the datasets used in autonomous driving and collected directly from embarked 3D sensors. Fourth, we present several 3D aerial datasets, which are acquired from LiDARs mounted on airborne platforms and offer unique challenges for point cloud processing.

2.3.1 Collections of Structured 3D Models

Synthetic 3D datasets offer a compelling alternative to expensive real-world acquisitions. Synthetic datasets are constructed from virtual 3D models and scenes, offering numerous advantages for developing new 3D data processing methodologies. One significant benefit is that ground truth annotations can be easily created for various tasks such as classification or semantic segmentation, making them ideal for supervised learning. Additionally, it is easier to control noise, occlusion, and lighting conditions on synthetic datasets.

ShapeNet [48] and ModelNet [347] are annotated datasets of 3D shapes that span a variety of categories, from chairs and cars to airplanes and tables. A subset of ShapeNet has been densely annotated [280], decomposing each object into several parts. For example, the table’s legs and plates are annotated as different classes. The more recently released 3D-FUTURE [94] and ABO [64] datasets compile high-quality CAD models of household objects with more geometric details and informative textures. Another relevant dataset is ABC [158], a large unannotated collection of one million CAD models that represent shapes designed for industrial purposes. For the purpose of our the-

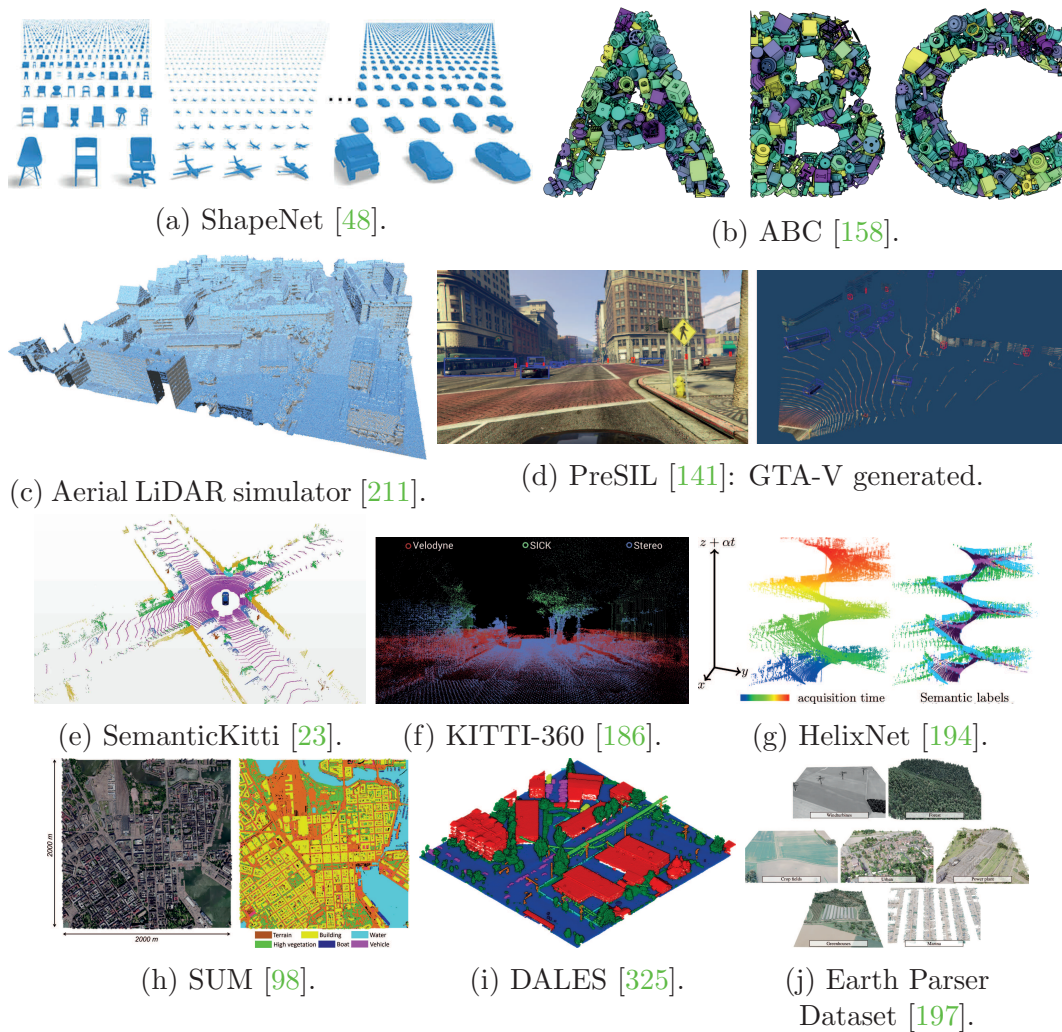


Figure 2.5: **3D Datasets.** As research is pushing the limits of 3D data analysis, the results are driven in part by the growing availability of high-quality datasets.

sis, we can easily use those structured models in our point-cloud processing pipelines by sampling points on the surfaces.

2.3.2 Large-Scale Simulated Datasets

To expand beyond simple shapes, more realistic datasets have been created using video game engines or by simulating aerial LiDAR acquisitions from existing meshes. These datasets enable the training and evaluation of computer vision models in more complex and diverse environments. For example, the PreSIL dataset [141] contains simulated 2D and 3D sequences captured from the popular video game “Grand Theft Auto V”, providing realistic urban

scenes with complex interactions between vehicles and pedestrians. Simulating aerial LiDAR [211, 208] or autonomous driving sensors [83] acquisitions from existing 3D models also provides realistic 3D urban environments for training semantic segmentation and object detection models [79]. These datasets enable researchers to explore new computer vision applications in challenging simulated scenarios which approximate real-world ones.

2.3.3 Autonomous Driving 3D Datasets

As autonomous driving becomes an increasingly realistic prospect, multiple datasets have been proposed to evaluate the performance of perception algorithms [239, 68]. In addition to cameras, rotating LiDAR sensors have become one of the most prevalent sensors mounted on autonomous vehicles due to their high accuracy, low latency, and steadily decreasing prices [273]. KITTI [101] and its annotations from SemanticKITTI [23] is one of the most widely used dataset for semantic scene understanding using LiDAR sequences. Released more recently, KITTI-360 [186] and the Waymo Open Dataset [303] contain images and LiDAR acquisitions with 2D and 3D annotations. ApolloScape [139], DublinCity [383], and TerraMobilita/iQmulus [324] have been acquired with LiDAR setups that offer scans of urban environments with high precision and density. However, their acquisitions are incompatible with real-time road perception. Several prominent datasets such as NuScene [43] or the very large ONCE dataset [209] provide only object-level annotations (*i.e.* boxes), which do not allow training models to predict dense labels. Even if those acquisitions are often highly precise, the amount of acquired data, the cost of annotating 3D data, and the diversity of sensors, make it challenging to design generic methods for autonomous driving perception.

2.3.4 Aerial LiDAR Datasets

The increase in the availability of aerial LiDAR technology has led to the multiplication of open datasets [240, 355, 98, 383] of varying sizes from 1 to 10 km² [325, 294]. These large-scale aerial scans are particularly challenging due to the high number of acquired points, and the occlusion between real-world objects, resulting in raw imperfect 3D scans. Moreover, these scans are often limited to dense urban environments and do not capture the diversity of real-

world terrains. Even if some specialized datasets focus on forested areas [153, 340], current aerial LiDAR datasets still do not cover the diversity of the real world evenly.

Chapter 3

Online Segmentation of LiDAR Sequences: Dataset and Algorithm

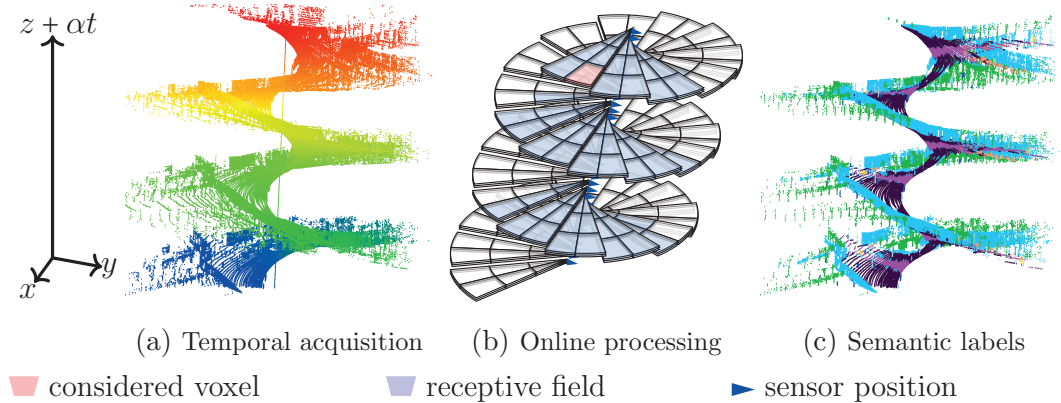


Figure 3.1: **Online LiDAR Segmentation.** The 3D point sequences of rotating LiDAR data of our proposed dataset HelixNet follow a complex helix-like structure in space and time, represented in (a) by using the vertical axis for both time and elevation. We propose an efficient spatio-temporal transformer to process angular slices of data centered on the sensor’s position. The slices are partitioned into voxels, each attending other voxels from past slices to build a large spatio-temporal receptive field (b). Our proposed model can segment the LiDAR point stream (c) with *state-of-the-art* accuracy and in real-time.

Abstract

Roof-mounted spinning LiDAR sensors are widely used by autonomous vehicles. However, most semantic datasets and algorithms used for LiDAR sequence segmentation operate on 360° frames, causing an acquisition latency incompatible with real-time applications. To address this issue, we first introduce HelixNet, a 10 billion point dataset with fine-grained labels, timestamps, and sensor rotation information necessary to accurately assess the real-time readiness of segmentation algorithms. Second, we propose Helix4D, a compact and efficient spatio-temporal transformer architecture specifically designed for rotating LiDAR sequences. Helix4D operates on acquisition slices corresponding to a fraction of a full sensor rotation, significantly reducing the total latency. Helix4D reaches accuracy on par with the best segmentation algorithms on HelixNet and SemanticKITTI with a reduction of over 5× in terms of latency and 50× in model size. The code and data are available at: <https://romainloiseau.fr/helixnet>.

This chapter’s work was initially presented in:

- [Romain Loiseau](#), Mathieu Aubry, Loic Landrieu, “Online Semantic Segmentation of LiDAR Sequences: Dataset and Algorithm”, ECCV, 2022.
- [Romain Loiseau](#), Mathieu Aubry, Loic Landrieu, “Helix4D: Online Semantic Segmentation of LiDAR Sequences”, CVPR Transformer for Vision Workshop, 2022.

3.1 Introduction

Due to their low acquisition latency and high precision, rotating LiDAR sensors are among the most prevalent sensors for autonomous vehicles [273]. The acquired sequences of 3D points exhibit a complex structure in which the temporal and spatial dimensions are entangled through the rotation of the sensor around a reference point in motion; see Figure 3.1. However, this structure is often not reflected in the formatting of open-access LiDAR datasets [23, 149, 186], which are discrete sequences of range images, or frames, each corresponding to a 360° degree arc around the sensor. Consequently, most LiDAR semantic segmentation methods operate on one or several such frames at the same time, in the image [69] or point cloud [381, 368, 307] format. However, waiting for an entire frame to be acquired introduces an unavoidable latency of more than 100ms on top of the processing time, excluding applications for high-speed or urban driving. In this chapter, we address this issue by introducing (i) HelixNet, the largest available LiDAR dataset, whose fine-grained point information allows for the realistic real-time evaluation of segmentation methods, and (ii) Helix4D, a spatio-temporal transformer designed for the efficient segmentation of LiDAR sequences.

Our dataset HelixNet, has several key advantages compared to standard datasets such as SemanticKITTI [23], see Table 3.1. By organizing points with respect to sensor rotation and reporting their precise release times, we can accurately benchmark the real-time readiness of leading *state-of-the-art* LiDAR sequence segmentation algorithms. Furthermore, the pointwise sensor orientation allows us to split the data into slices of acquisition corresponding to a fraction of the sensor’s rotation. These slices can be processed sequentially by our proposed network Helix4D, resulting in a lower acquisition latency and a more realistic scenario for autonomous driving. Based on a spatio-temporal transformer designed explicitly for LiDAR sequences, Helix4D is more than 50 times smaller than the current best semantic segmentation architectures and reaches *state-of-the-art* performance with significantly reduced latency.

3.2 Related Work

Semantic segmentation of rotating LiDAR data. A first set of methods designed for rotating LiDAR sequences processes the data in range image format [185, 304, 69]: 2D projections showing the distances between the sensor and acquired 3D points. Taking advantage of advances in the implementation of sparse convolutions [113, 59], a second set of methods projects the point cloud into fine grids in polar [368], Cartesian [307, 57] or cylindrical [381, 135] coordinates. A third kind of approach exploits the temporal dimension of LiDAR acquisitions by *stacking* contiguous frames [59, 16]. Observing that cylindrical partitions better capture the geometry of rotating LiDAR acquisition, our proposed method Helix4D builds on the proposed Cylinder3D [381] architecture by adding an attention mechanism which models the spatio-temporal acquisition.

Attention-based networks. Due to their remarkable performance and scalability, transformers [326] have quickly been adapted from text processing to images [82, 191, 299, 44], videos [10], or meshes [187]. Transformers are also well suited to handle unordered sets, such as 3D point clouds [122, 373]. In particular, their scalability can be leveraged to achieve large receptive fields [210, 245] and more discriminative features [27] than purely convolutional approaches. Transformers can also efficiently process complex temporal [154, 330] and spatio-temporal [85] sequences. In the wake of hybrid convolution-transformer models [121, 62, 71], our proposed model Helix4D combines efficient cylindrical convolutions with a simplified spatio-temporal transformer architecture operating at low resolution.

Roof-mounted rotating LiDAR datasets. This chapter focuses on semantic segmentation algorithms for roof-mounted rotating LiDAR sensors. In previously released datasets, the 3D points are given as sets called “frames”, each representing a full 360 degrees rotation of the sensor. In Table 3.1, we report several key characteristics of other existing datasets that are either organized in complete sensor rotations [149, 101, 23, 186, 104] or in an arbitrary number of rotations [272, 306]. On the contrary, the 3D points of HelixNet are given with respect to the sensor rotation and in the order in which they are made available. This last point proves crucial for evaluating the precision

Table 3.1: **Embarked LiDAR Datasets with Semantic Point Annotations.** With over 8.8B annotated 3D points, HelixNet is 70% larger than SemanticKITTI, and includes more diverse scenes spanning 6 different French cities. Contrary to other datasets, HelixNet arranges points with respect to the sensor rotation and contains fine-grained information about their release time.

| Dataset | labels | frames | classes | span | format |
|-------------------------|--------|--------|---------|----------|-----------------|
| HelixNet (Ours) | 8.85B | 78k | 9 | 6 cities | sensor rotation |
| SemanticKITTI [101, 23] | 5.2B | 43k | 19 | 1 city | frame |
| Rellis3D [149] | 1.5B | 13k | 16 | 1 city | frame |
| KITTI-360 [186] | 1.0B | 81k | 37 | 1 city | frame |
| A2D2 [104] | 387M | 41k | 38 | 3 cities | frames |
| Paris-Lille-3D [272] | 143M | N/A | 50 | 2 cities | multi-frame |
| Toronto3D [306] | 78M | N/A | 8 | 1 city | multi-frame |

and latency of segmentation algorithms in a setting that is compatible with real-time inference. Moreover, our proposed dataset HelixNet is 70% larger than SemanticKITTI [101, 23], and spans 6 cities and various environments.

3.3 HelixNet: A Dataset for Online LiDAR Segmentation

We introduce HelixNet, a new large-scale and open-access LiDAR dataset intended for the evaluation of real-time semantic segmentation algorithms. In contrast to other large-scale datasets, HelixNet includes fine-grained data on sensor rotation and position, as well as point release time. We report the localization of the sequences of HelixNet in detail in Figure 3.4.

General characteristics. As seen in Figure 3.2, HelixNet contains 20 sequences of 3D points, each corresponding to 6 to 7 minutes of continuous acquisition, for a total of 129 minutes. Scanning was performed by an HDL-64E Velodyne rotating LiDAR [144] mounted on a mobile platform [248]. As shown in Figure 3.3, HelixNet covers multiple cities and a wide variety of environments such as a university campus, dense historical centers, and a highway interchange. With a total of 10 billion points across 78 800 frames and 8.85

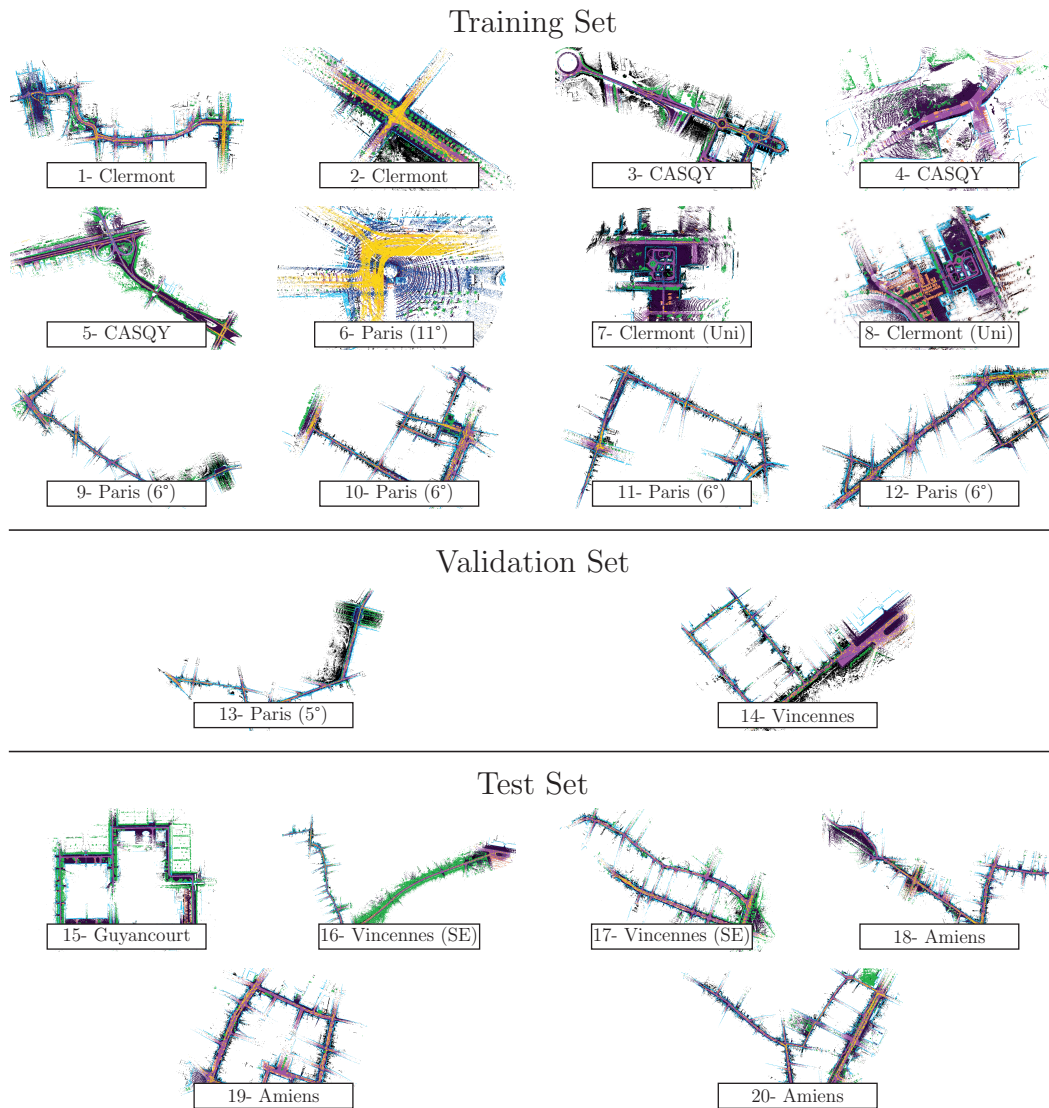


Figure 3.2: **Coverage from HelixNet.** We split the acquisitions into 12 training, 2 validation, and 6 testing sequences. HelixNet contains diverse scenes in various urban environments from static or mobile sensors.

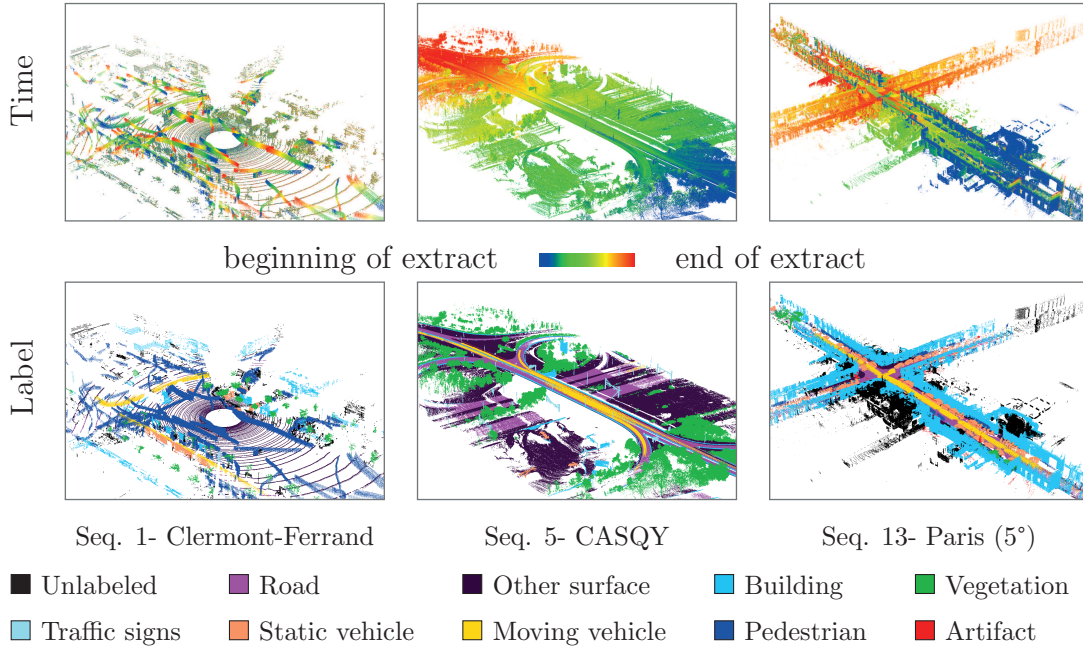


Figure 3.3: **Extracts from HelixNet.** Our proposed dataset contains various urban scenes from motorway to pedestrian plazas and historical centers. In the first row, we represent extracts of 15 to 30 seconds of acquisition colored according to the point release time. In the second row, we represent the point semantic labels.

billion individual labels, HelixNet is the largest densely annotated open-access rotating LiDAR dataset by a factor of 1.7 as shown in Table 3.1. HelixNet follows the file format of SemanticKITTI [23] allowing researchers to evaluate existing code with minimal effort.

We use a 9-classes nomenclature: *road* (16.4% of all points), *other surface* (22.0%), *building* (31.3%), *vegetation* (8.5%), *traffic signs* (1.6%), *static vehicle* (4.9%), *moving vehicle* (2.1%), *pedestrian* (0.9%), and *acquisition artifact* (0.05%). Points without labels correspond to either *un-annotated* (6.2%) parts of the clouds due to their ambiguity, or point without echos (6.1%). Compared to fine-grained classes such as the ones used by SemanticKITTI [23] or Paris-Lille3D [272], our focused nomenclature limits class imbalance and makes macro-averaged metrics more stable.

Each point is associated with the 9 following values: (1-3) Cartesian coordinates in a fixed frame of a reference, (4-6) cylindrical coordinate relative to the sensor at the time of acquisition, (7) intensity, (8) fiber index, and (9) packet output time. As detailed in the next paragraph, the last two features are not typically available in large-scale datasets and cannot be inferred. However,

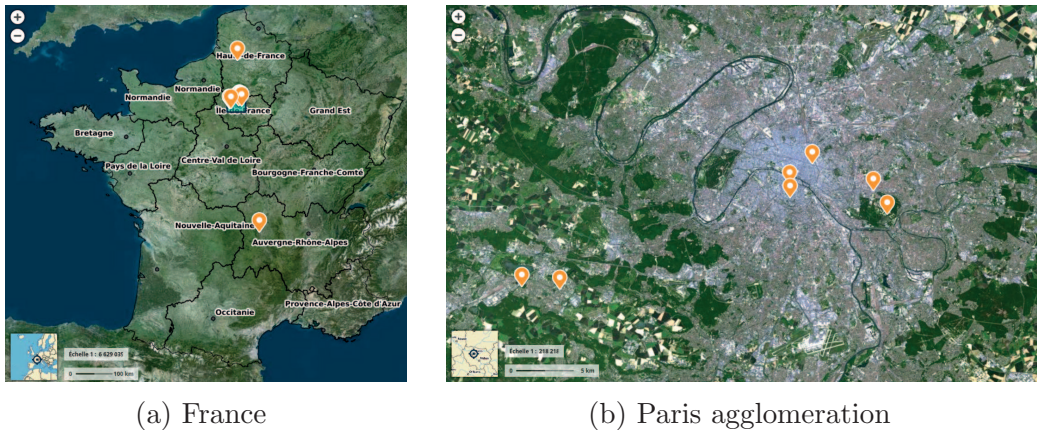


Figure 3.4: **Localization of the Sequences.** HelixNet’s data were acquired in 6 different cities with 4 of them in the Paris agglomeration, spanning a large variety of landscapes and urban configuration.

they play a key role in measuring latency in a real-time setting.

Sensor-based timing and grouping. A rotating LiDAR consists of a set of lasers—or fibers—arranged on a rotating sensor head. The lasers send periodic pulses of light whose return times give the position of the impact points relative to the sensor. In the context of autonomous driving, these sensors are typically deployed on a moving platform and capture 3D points with centimetric accuracy. The sensor releases the data stream as a discrete temporal sequence of *packets* of 3D points. For an HDL-64E LiDAR, each packet contains 6×64 points, corresponding to around 1° rotation of the sensor. To represent the real-time operational setting of autonomous driving, we associate with each point the timestamp of its *packet output event*, *i.e.* the instant the packet is available and not the acquisition time of the point. The latency between the acquisition of the first point and the complete transfer of its packet is $278\mu\text{s}$. Although small compared to acquisition and inference times, this more rigorous timing constitutes a step towards a more realistic evaluation setting of segmentation algorithms of LiDAR sequences.

On top of its absolute position, we associate with each individual point its cylindrical coordinates relative to the position of the sensor at the exact time of its acquisition. This differs from other datasets such as SemanticKITTI [23], which gives the relative position of all points but the absolute position of the sensor only once per frame. While sensor movement can be interpolated, the vehicle trajectory might not be linear and the sensor head rotates. For

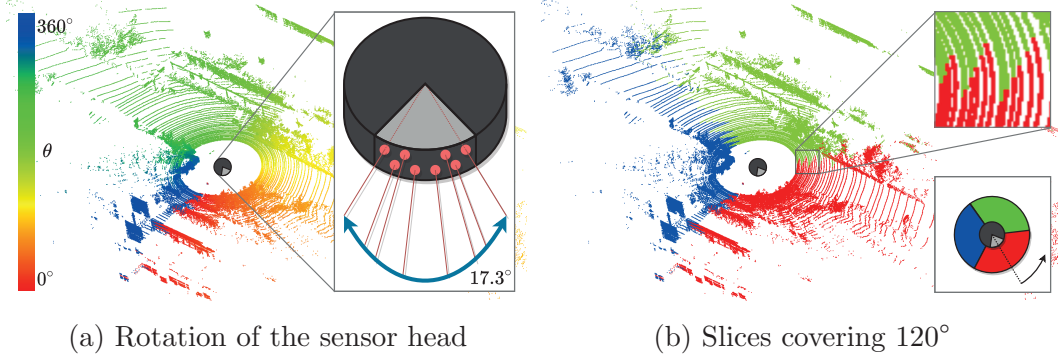


Figure 3.5: **Sensor Acquisition Geometry.** We represent in (a) the acquisition of a rotating sensor, which is split into $\frac{1}{3}$ turn slices in (b). As the laser emitters position forms an angle of over 17.3° around the sensor head, taking slices with respect to the sensor rotation θ results in a jagged profile.

comparison, at 50km/h, the sensor moves more than 1.4m during each rotation.

LiDAR sequences are typically split into frames containing points that cover a 360° degree arc around the sensor. However, the acquisition geometry makes this grouping artificial. Indeed, the fibers (*i.e.* the individual lasers) do not all face the same direction: they are arranged around the sensor’s heads at different angles, with a range of more than 17.3° . This means that the points within a packet are not vertically aligned but present a jagged profile as seen in Figure 3.5. In order to obtain frames with *straight edges* such as those of SemanticKITTI [23], we would have to consider an acquisition over a sensor rotation of 377° , adding a further 5ms of latency. Contrary to other datasets, HelixNet contains the index of the emitter of each point and organizes the points with respect to the angle of the sensor itself. This allows us to easily build frames or frame portions that are directly consistent with the rotation *of the sensor head itself*. This is important for measuring the real latency of segmentation methods and, as described in the next section, contributes to the efficiency of our proposed network.

3.4 Helix4D: Fast LiDAR Segmentation with Transformers

We consider a sequence of 3D points acquired by a rotating LiDAR on a mobile platform, which we split into chronologically ordered slices of acquisition. As represented in Figure 3.6, we process each slice with a U-Net architecture [270]

with cylindrical convolutions [381]. At the lowest resolution, a spatio-temporal transformer network connects neighboring voxels in space and time, resulting in a large receptive field. We first describe the construction of slices, then our cylindrical U-Net, and finally the transformer module.

3.4.1 Temporal Slicing

Instead of processing the data frame-by-frame, we propose to split the sequence into slices covering a fixed portion of the sensor rotation, resulting in a shorter acquisition time and a lower latency. Each point i of the sequence is characterized by the angular position θ_i of the sensor head at its exact time of acquisition. The points are sorted in chronological acquisition order *i.e.* $\theta_i \leq \theta_j$ if $i < j$. We partition the sequence into groups of contiguous points called slices, acquired during a portion $\Delta\theta \in]0, 2\pi]$ of a full rotation of the sensor itself. Choosing $\Delta\theta = 2\pi$ corresponds to the classic frame-by-frame setting and implies an acquisition latency of 104ms in HelixNet or SemanticKITTI [23]. A slice size of $\Delta\theta = 2\pi/5$ leads to an acquisition latency of 21ms, which is more conducive to real-time processing of driving data.

3.4.2 Cylindrical U-Net

Inspired by the Cylinder3D model [381], we first discretize each slice along a fine cylindrical partition `grid(1)`. Each point i is associated with a descriptor x_i^{point} based on its intensity, relative position with respect to the sensor in Cartesian and cylindrical coordinates, and its offset with respect to the center of its voxels in `grid(1)`. We compute the point feature f_i^{point} by applying a shared MLP $\mathcal{E}^{\text{point}}$ to x_i^{point} for all points i in the slice. The resulting f_i^{point} are then maxpooled with respect to the voxels of `grid(1)` to serve as input to a convolutional encoder $\mathcal{E}^{\text{grid}}$. The network $\mathcal{E}^{\text{grid}}$ is composed of sparse cylindrical convolutions [113] and strided convolutions for downsampling. $\mathcal{E}^{\text{grid}}$ produces a set of L sparse feature maps $f^{\text{grid}(1)}, \dots, f^{\text{grid}(L)}$ with decreasing resolutions:

$$f_i^{\text{point}} = \mathcal{E}^{\text{point}}(x_i^{\text{point}}) \quad (3.1)$$

$$f^{\text{grid}(1)}, \dots, f^{\text{grid}(L)} = \mathcal{E}^{\text{grid}}(\text{maxpool}(f^{\text{point}})) \quad , \quad (3.2)$$

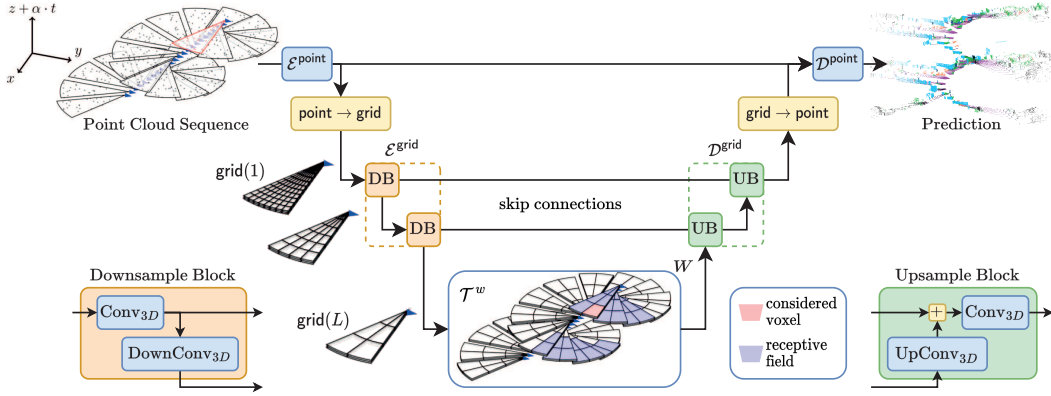


Figure 3.6: **Helix4D Architecture.** A point sequence is split into angular slices, whose points are encoded by $\mathcal{E}^{\text{point}}$ and pooled along a fine-grained cylindrical partition. A convolutional encoder $\mathcal{E}^{\text{grid}}$ yields feature maps at lower resolutions. We apply W consecutive spatio-temporal transformer blocks \mathcal{T}^w on the coarse voxels, with attention spanning across current and past slices. The resulting features are up-sampled to full resolution with a convolutional decoder $\mathcal{D}^{\text{grid}}$ using the encoder’s maps at intermediate resolutions through skip connections. Finally, the grid features are allocated to the points, which are classified by $\mathcal{D}^{\text{point}}$.

where maxpool is performed with respect to $\text{grid}(1)$. At the lowest resolution $\text{grid}(L)$, we apply the transformer-based module \mathcal{T} presented in the next subsection to the feature map $f^{\text{grid}(L)}$ to obtain the coarse cylindrical map $g^{\text{grid}(L)}$:

$$g^{\text{grid}(L)} = \mathcal{T}(f^{\text{grid}(L)}) . \quad (3.3)$$

The decoder $\mathcal{D}^{\text{grid}}$ combines cylindrical convolutions and strided transposed convolutions to map $g^{\text{grid}(L)}$ to a feature map $g^{\text{grid}(1)}$ at the highest resolution, and uses the maps $f^{\text{grid}(L-1)}, \dots, f^{\text{grid}(1)}$ through residual skip connections. We concatenate for each point i the descriptor $g^{\text{grid}(1)}(i)$ of its voxel in $\text{grid}(1)$ and its point feature f_i^{point} . Finally, the point decoder $\mathcal{D}^{\text{point}}$ associates a vector of class scores c_i^{point} with each point i :

$$g^{\text{grid}(1)} = \mathcal{D}^{\text{grid}}(g^{\text{grid}(L)}, f^{\text{grid}(L-1)}, \dots, f^{\text{grid}(1)}) \quad (3.4)$$

$$c_i^{\text{point}} = \mathcal{D}^{\text{point}}([g^{\text{grid}(1)}(i), f_i^{\text{point}}]) , \quad (3.5)$$

where $[\cdot]$ is the channelwise concatenation operator. The network is supervised by the cross-entropy and Lovász-softmax [26] losses directly on the point

prediction, without class weights.

Our approach differs from Cylinder3D [381] by relying on simple $3 \times 3 \times 3$ sparse cylindrical convolutions instead of asymmetrical convolutions and dimension-based context modeling. Furthermore, we do not use voxel-wise supervision.

Our simplified architecture results in a lighter computational and memory load, but can still learn rich spatio-temporal features thanks to the addition of the transformer module described below.

3.4.3 Spatio-Temporal Transformer

We denote by \mathcal{V} the set of non-empty voxels at the lowest resolution $\text{grid}(L)$ for all slices of the considered sequence. We associate with each voxel v of \mathcal{V} a feature f_v^{voxel} defined as the value of $f^{\text{grid}(L)}$ at v . We remark that f^{voxel} can be ordered as a non-strictly ordered time sequence, and propose to successively apply W independent transformer blocks $\mathcal{T}^1, \dots, \mathcal{T}^W$ whose architecture is described below. We denote by g^{voxel} the resulting spatio-temporal voxel representation:

$$g^{\text{voxel}} = \mathcal{T}^W \circ \dots \circ \mathcal{T}^1(f^{\text{voxel}}) . \quad (3.6)$$

We associate each voxel v of \mathcal{V} with the absolute position (X_v, Y_v, Z_v) of its center, the release time T_v of its first point, and the index I_v of the sensor rotation of its corresponding slice. In order to use a sparse attention scheme, we define for each voxel v a spatio-temporal mask $M(v)$ characterized by a radius R and a set of rotation offsets $P \subset \mathbb{N}$:

$$M(v) = \{u \mid \|(X_v, Y_v, Z_v) - (X_u, Y_u, Z_u)\| < R, I_v - I_u \in P\} . \quad (3.7)$$

In the context of autonomous driving, we choose $R = 6\text{m}$ and $P = \{0, 5, 10\}$. With a standard rotation speed of 10Hz, this corresponds to slices 0.5 and 1 seconds in the past along with the current one. See Figure 3.7 for an illustration of the receptive field and attention maps.

Simplified transformer block. We now define a single transformer block \mathcal{T}^w with H heads operating on a sequence of voxel features f^{voxel} of dimension

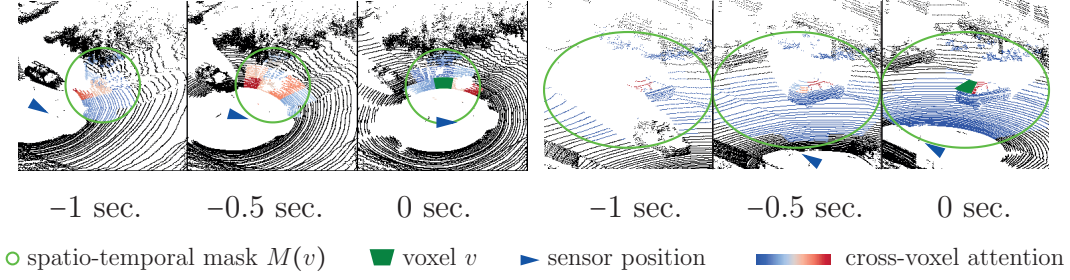


Figure 3.7: **Spatio-Temporal Attention.** We represent the spatio-temporal mask and attention score of one head of the transformer for two different voxels. The network gathers information from different frame offsets P as the sensor moves.

D. For each head h and each voxel v , we apply the following operations:

- (i) A single linear layer \mathcal{L}^h generates both a key k_v^h of dimension K and a value val_v^h of dimension D/H .
- (ii) For all voxels u in the mask $M(v)$, we define the compatibility score $y_{u,v}^h$ as the cross-product between keys and with a learned relative positional encoding $\text{PE}^h(u, v)$.
- (iii) The cross-voxel attention $a_{u,v}^h$ is obtained with a scaled softmax.
- (iv) The values val_u^h of voxels in $M(v)$ are averaged into a vector \tilde{f}_v^h using their respective cross-voxel attention as weights.
- (v) The vectors \tilde{f}_v^h are concatenated channelwise across heads and added to the input of the block to define its output.

These operations can be summarized as follows:

$$k_v^h, \text{val}_v^h = \mathcal{L}^h(f_v^{\text{voxel}}) \quad (3.8)$$

$$y_{u,v}^h = (k_v^h)^\top (k_u^h + \text{PE}^h(u, v)) \quad \text{for } u \in M(v) \quad (3.9)$$

$$\{a_{u,v}^h\}_{u \in M(v)} = \text{softmax}\left(\{y_{u,v}^h\}_{u \in M(v)} / \sqrt{K}\right) \quad (3.10)$$

$$\tilde{f}_v^h = \sum_{u \in M(v)} a_{u,v}^h \text{val}_u^h \quad (3.11)$$

$$\mathcal{T}^w(f^{\text{voxel}})_v = f_v^{\text{voxel}} + [\tilde{f}_v^1, \dots, \tilde{f}_v^H]. \quad (3.12)$$

Our design is similar to the classical transformer architecture but uses keys as queries to save memory and computation. We also do not use feed-forward

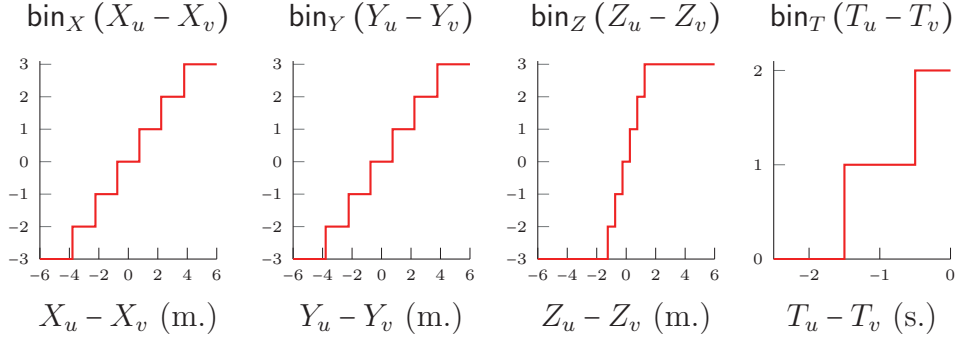


Figure 3.8: **Relative Positional Encoding Bins.** Each of the four dimension are split into irregular bins inspired by Wu *et al.* [344].

networks after averaging the values: the only learnable part of a block \mathcal{T}^w is its linear layers \mathcal{L}^h and its relative positional encoding PE^h .

Since g^{voxel} only requires information about the voxels of the current and past slices, it can be computed sequentially for all slices in the order in which the sensor releases them. For a given slice, the voxel map $g^{\text{grid}(L)}$ for non-empty voxels is given by the values of g^{voxel} , and set to zero otherwise. To save computation at inference time, we store in memory the keys, values, and absolute positions of the voxels in past slices with a fixed buffer of $\max(P)$ rotations. This allows us to allocate a large spatio-temporal receptive field to each voxel without supplementary computations.

Relative positional encoding. We propose to learn relative positional vectors $\text{PE}^h(u, v)$ that encode the spatio-temporal offset $(X_u, Y_u, Z_u, T_u) - (X_v, Y_v, Z_v, T_v)$ between voxels u and v for each transformer block w independently. Inspired by the work of Wu *et al.* [344], we first discretize the offsets along each dimension $d \in \{X, Y, Z, T\}$ with B_d irregular bins. For each dimension d and head h , we learn B_d weight vectors of size K . We define the functions $\text{PE}_d^h: \mathbb{R} \mapsto \mathbb{R}^K$ that map the d -dimension of an offset to the vector associated with its corresponding bin. The positional encoding between two voxels u and v is the sum of the vectors corresponding to their discretized offsets in each dimension:

$$\begin{aligned} \text{PE}^h(u, v) &= \text{PE}_X^h(X_u - X_v) + \text{PE}_Y^h(Y_u - Y_v) \\ &\quad + \text{PE}_Z^h(Z_u - Z_v) + \text{PE}_T^h(T_u - T_v). \end{aligned} \quad (3.13)$$

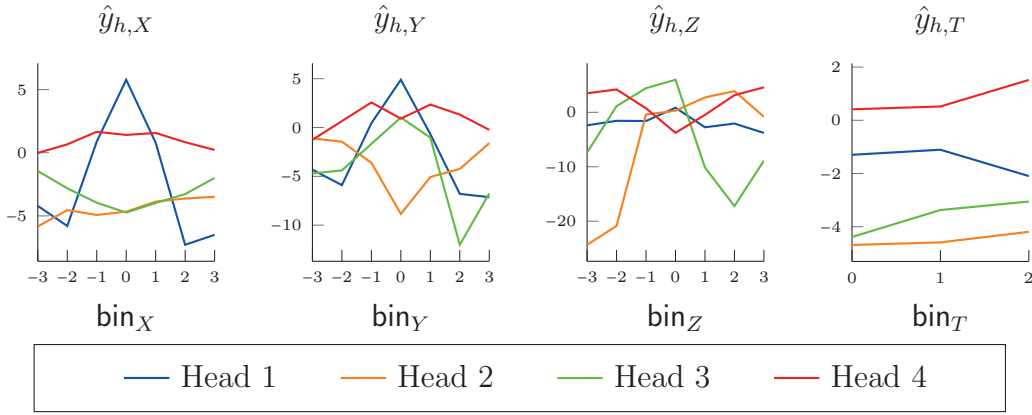


Figure 3.9: **Positional Encoding.** We plot the average compatibility with respect to the offset learned for the second encoder block of Helix4D within different dimensional bins. The compatibility is the cross product between voxel keys and the positional encoding, averaged over 3 rotations of HelixNet (see Equation 3.15).

Relative positional encoding vectors are used directly in the calculation of the compatibility score, as given in Equation 3.9. We use $B_X = B_Y = B_Z = 7$ and $B_T = \text{Card}(P) = 3$ irregular bins for the relative positional encoding. For an offset x along dimension $d \in \{X, Y, Z, T\}$, the corresponding bin for the pair of voxels u, v is given by the following piecewise function [344, Eq 18]:

$$\text{bin}_d(x) = \begin{cases} \lfloor x/\rho_d \rfloor & \text{if } |x| \leq \alpha\rho_d \\ \text{sign}(x) \times \min\left(\beta, \left\lfloor \alpha + \frac{\ln(|x|/(\alpha\rho_d))}{\ln(\gamma/\alpha)}(\beta - \alpha) \right\rfloor\right) & \text{if } |x| > \alpha\rho_d, \end{cases} \quad (3.14)$$

where $\lfloor \cdot \rfloor$ denotes the rounding operation, $\alpha = 2$, $\beta = 3$, and $\gamma = 1.25$, and $\rho_X = \rho_Y = 1.5\text{m}$, $\rho_Z = 0.5\text{m}$, and $\rho_T = 5 \times 104\text{ms}$. See Figure 3.8 for a visual representation of bins.

We represent in Figure 3.9 the compatibility $\hat{y}_{h,d}(b)$ between keys and relative positional encodings within each bin b , head h and dimension $d \in \{X, Y, Z, T\}$ averaged over 3 rotations:

$$\hat{y}_{h,d}(b) = \frac{1}{\text{Card}(\mathcal{V}_d(b))} \sum_{(v,u) \in \mathcal{V}_d(b)} (k_v^h)^\top (\text{PE}^h(u, v)), \quad (3.15)$$

where $\mathcal{V}_d(b) = \{(v, u) \in \mathcal{V} \mid \text{bin}(d_u - d_v) = b\}$. We observe differentiated behaviors between the heads: Head 1 focuses on near voxels, Head 2 and Head 3 focus on the area above/below the voxel, respectively, and Head 4 focuses on the

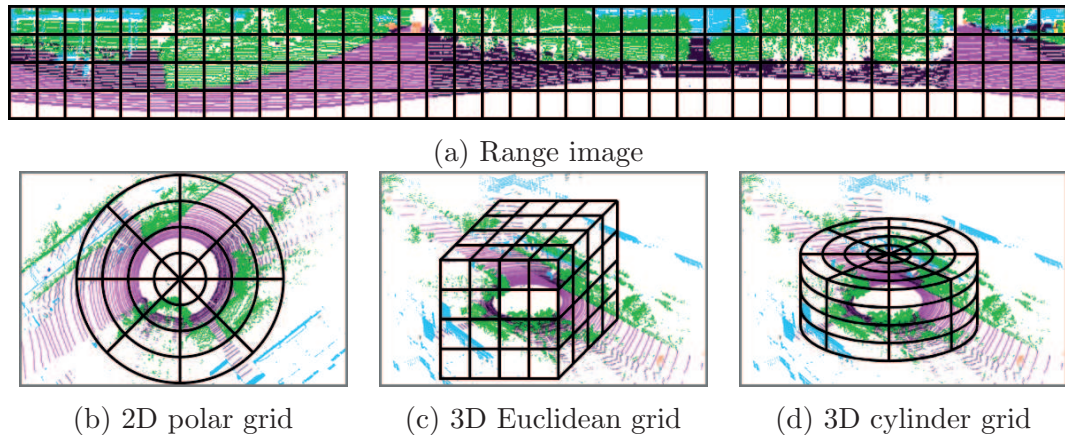


Figure 3.10: **Data Partitioning Schemes.** We represent commonly used partitioning: (a) SalsaNeXt [69] uses range images, (b) PolarNet [368] uses a bird-eye-view polar partition, (c) SPVNAS [307] uses a classic regular 3D grid, (d) Cylinder3D [381] uses a cylinder grid. Grids not to scale.

current frame.

3.5 Evaluating Online Semantic Segmentation

We evaluate the performance and inference time of our approach and other *state-of-the-art* methods in both online and frame-by-frame settings. We use our proposed dataset HelixNet and the standard SemanticKITTI dataset.

Online evaluation setting. We aim at evaluating the real-time readiness of rotating LiDAR semantic segmentation algorithms in the context of autonomous driving. The total latency of a model is determined by its inference speed and also the time it takes to acquire its input. Operating on full frames requires at least 104ms of acquisition, which is incompatible with realistic autonomous driving scenarios. Instead, we propose an online evaluation setting using the slices defined in Sec. 3.4.1. By default, we use a slice size of a fifth turn of the sensor head: $\Delta\theta = 2\pi/5$, corresponding to 21ms of acquisition.

Slices are processed sequentially. We define the inference latency of a segmentation method as the average time between the release of the last point of a slice and its segmentation. To meet the real-time requirement, inference must be faster than the acquisition of a slice. Slower processing would cause the classification to continuously fall behind. Although thinner slices directly reduce acquisition latency, they also make the real-time requirement more strict:

Table 3.2: **Semantic Segmentation Results.** Performance of Helix4D and competing approaches on HelixNet and on the validation set of SemanticKITTI*, in the frame-by-frame and online setting. We report the mean Intersection-over-Union (mIoU) and the inference time in ms. Methods meeting the real-time requirement are indicated with ✓ and those who do not with ✗. * SemanticKITTI is denoted as SK. Measuring the latency on this dataset requires making non-realistic approximations about the fiber position.

| Method | Size ×10 ⁶ | Full frame ● | | 104ms | ½ frame ► | | 21ms |
|-----------------------|--------------------------|--------------|-------------|-------------|-------------|-------------|-------------|
| | | HelixNet | SK* | Inf. (ms) | HelixNet | SK* | Inf. (ms) |
| SalsaNeXt [69] | 6.7 | 69.4 | 55.8 | 23 ✓ | 68.2 | 55.6 | 10 ✓ |
| PolarNet [368] | 13.6 | 73.6 | 58.2 | 49 ✓ | 72.2 | 56.9 | 36 ✗ |
| Pan. PolarNet [380] | 13.7 | — | 64.5 | 50 ✓ | — | 60.3 | 44 ✗ |
| SPVNAS [307] | 10.8 | 73.4 | 64.7 | 73 ✓ | 69.9 | 57.8 | 44 ✗ |
| Cylinder3D [381] | 55.9 | 76.6 | 66.9 | 108 ✗ | 75.0 | 65.3 | 54 ✗ |
| Helix4D (Ours) | 1.0 | 79.4 | 66.7 | 45 ✓ | 78.7 | 66.8 | 19 ✓ |

as a full turn must be processed in less than 104ms, a fifth turn must be in at most 21ms.

Adapting SemanticKITTI. SemanticKITTI [23, 101] contains 43 552 frames along 22 sequences of LiDAR scans densely annotated with 19 classes. In contrast to HelixNet, SemanticKITTI is not formatted with respect to the sensor rotation and only gives the acquisition time and sensor position once per frame. To measure the latency, we make the following approximations: (i) the fibers are assumed to be vertically aligned, meaning that the angle of the points is the same as the sensor’s; (ii) we interpolate the acquisition time of points between frames from their angular positions; (iii) we use the acquisition time as release time. To obtain the absolute positions of the voxels, we assume that the sensor jumps between the positions given by the LiDAR poses for each frame. In our open-source implementation, we provide an adapted dataloader allowing methods already running on SemanticKITTI to be evaluated in the online setting with minimal adaptation.

Adapting competing methods. To evaluate the semantic segmentation performance and latency of other segmentation algorithms in the online setting, we process the point clouds corresponding to each slice independently and sequentially. This approach restricts the spatial receptive field to the extent of the slices. However, as the sensor moves, it is not straightforward to add past

Table 3.3: **HelixNet Semantic Segmentation Scores.** We report the IoU for each class of HelixNet evaluated in the online setting with slices of 72° .

| Method | Road | Other surface | Building | Vegetation | Traffic signs | Static vehicle | Moving vehicle | Pedestrian | Artifact | Avg |
|-----------------------|-------------|---------------|-------------|-------------|---------------|----------------|----------------|-------------|-------------|-------------|
| SalsaNeXt [69] | 84.4 | 76.1 | 88.7 | 70.7 | 61.4 | 58.6 | 35.1 | 68.5 | 69.7 | 68.2 |
| PolarNet [368] | 86.2 | 77.9 | 91.2 | 77.9 | 63.2 | 64.8 | 35.4 | 68.1 | 84.8 | 72.2 |
| SPVNAS [307] | 80.5 | 77.1 | 93.0 | 81.8 | 68.0 | 60.9 | 36.9 | 71.7 | 59.0 | 69.9 |
| Cylinder3D [381] | 85.3 | 78.4 | 93.5 | 83.9 | 66.2 | 63.3 | 35.7 | 77.7 | 90.9 | 75.0 |
| Helix4D (Ours) | 87.8 | 82.5 | 94.0 | 84.4 | 68.9 | 72.3 | 46.4 | 78.8 | 93.3 | 78.7 |

slices whose relative positions may no longer be valid. By modeling the spatio-temporal offset between voxels, Helix4D does not suffer from this limitation.

We selected five segmentation algorithms with open-source implementations and trained models for SemanticKITTI. SalsaNeXt [69] uses range images, PolarNet [368] and panoptic PolarNet [380] a bird’s eye view polar grid, SPVNAS [307] a regular grid, and Cylinder3D [381] a cylindrical grid. See Figure 3.10 for a representation of each method’s partition. We do not consider methods that stack frames as their structure and resulting latency is incompatible with the online setting. When using SemanticKITTI, we evaluate the provided pretrained models on the validation set. On HelixNet, we retrain the models from scratch using the procedure of their official repository. We removed all test-time augmentations that resulted in prohibitive inference time. All methods are evaluated on the same workstation using a NVIDIA TESLA V100 32Go GPU.

Analysis. In Table 3.2, we report performance in frame-by-frame and online setting with slices of 72° , for Helix4D and competing methods, for HelixNet and SemanticKITTI. We observe that Helix4D yields *state-of-the-art* accuracy, with mIoU scores only matched by Cylinder3D [381]. However, Cylinder3D is 50 times larger in terms of parameters and twice slower, not meeting the real-time requirement even in the full frame setting. As reported in Table 3.3, distinguishing moving vehicles in HelixNet is particularly difficult. Our approach even largely outperforms Panoptic PolarNet despite this method using instance annotation as supervision, preventing us from evaluating on HelixNet. Helix4D yields significantly improved scores thanks to its

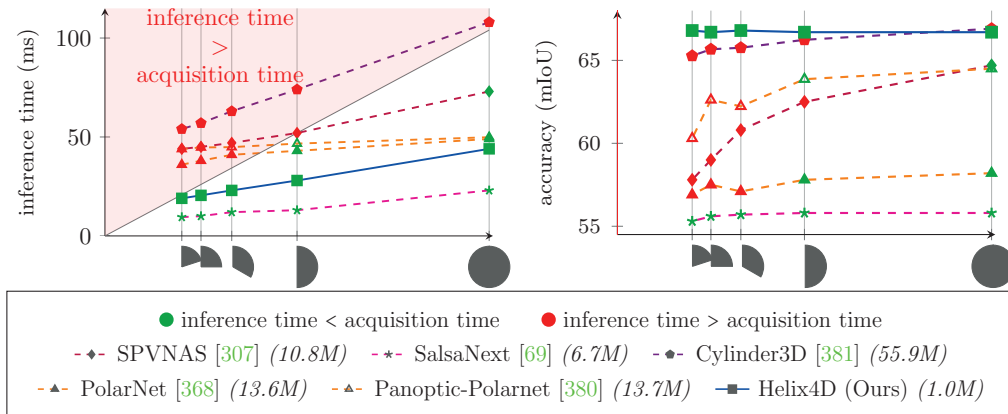


Figure 3.11: **Influence of Slice Size.** We plot the processing time (left, in ms) and precision (right, in mIoU) of different methods with respect to the considered size of slices, estimated on the validation set of SemanticKITTI [23]. Methods whose inference time is slower than the acquisition time of the slice (red shaded area) do not meet the real time requirement.

larger spatio-temporal receptive fields: 14m and 1000ms *vs.* 8m and 21ms for Cylinder3D for a fifth rotation. In the online setting, only two approaches meet the real-time requirement: SalsaNeXt [69] and Helix4D. Our approach outperforms SalsaNeXt by over 10 mIoU points in both the full frame and the on-line settings. In short, Helix4D is as accurate as the largest and slowest models with an inference speed comparable to that of the fastest and less accurate models. The total latency (acquisition plus inference time) of our model evaluated online is 40ms (21 + 19ms), and reaches the same performance as Cylinder3D evaluated on full frame with a latency of 212ms (104 + 108ms), an acceleration of more than 5 folds.

In Figure 3.11, we report the inference time and mIoU for different slice sizes. Due to various overheads, the inference time appears in an affine relationship with the size of slices, making the real-time requirement stricter for smaller slices. Due to its very design, the performance of Helix4D is not affected by the slice size. In contrast, competing methods perform worse with smaller slices.

Ablation study. We assess on SemanticKITTI the impact of different design choices by evaluating several alterations of our method, reported in Table 3.4.

(a) **Asymmetric Convolutions:** we replace the $3 \times 3 \times 3$ convolutions in our U-Net with the convolution design proposed by Cylinder3D [381]. We did

Table 3.4: **Ablation Study.** We report the speed and accuracy of several modification of our Helix4D on the validation set of SemanticKITTI.

| Method | Size $\times 10^3$ | Full Frame ● | 104ms | $\frac{1}{2}$ Frame ► | 21ms |
|------------------------------|-----------------------|--------------|-------------|-----------------------|-------------|
| | | mIoU | Inf. (ms) | mIoU | Inf. (ms) |
| Helix4D | 985 | 66.7 | 45 ✓ | 66.8 | 19 ✓ |
| (a) Asymmetric Convolutions | 1171 | 66.6 | 56 ✓ | 66.6 | 31 ✗ |
| (b) Cylindrical U-Net | 985 | 58.6 | 22 ✓ | 60.2 | 16 ✓ |
| (c) Slice-by-Slice | 985 | 62.9 | 29 ✓ | 62.6 | 19 ✓ |
| (d) w. Queries | 993 | 65.2 | 45 ✓ | 64.8 | 20 ✓ |
| (e) w/o. Positional Encoding | 983 | 64.3 | 41 ✓ | 64.1 | 18 ✓ |
| (f) Helix4D Tiny | 306 | 65.3 | 45 ✓ | 64.9 | 17 ✓ |

not observe a significant change in performance and an increase in run-time of 50%, failing the real-time requirement for slices of 72°.

(b) **Cylindrical U-Net:** we replace the transformer by a $1 \times 1 \times 1$ convolution on the voxels of the lowest resolution. We observe a slight decrease in run-time and a significant drop of over 6 mIoU points. This result shows that the transformer is able to learn meaningful spatio-temporal features at low resolution.

(c) **Slice-by-Slice:** we restrict the mask $M(v)$ of each voxel to its current slice. This reduction in the temporal receptive field results in a drop of 4 mIoU points, without any appreciable acceleration.

(d) **w. Queries:** we modify our simplified transformer to associate a query for each voxel along with keys and values, and use key-queries compatibilities. This does not affect the run-time and slightly decreases the performance.

(e) **w/o. Positional Encoding:** we remove the relative positional encoding PE in the calculation of compatibilities in equation Equation 3.9. This leads to a slightly decreased run time, but decreases performance by more than 2.5 points. This illustrates the advantage of explicitly modeling the spatio-temporal voxel offsets.

(f) **Helix4D Tiny:** we replace the learned pooling in our U-Net with max-pools and use narrower feature maps for a total of $306k$ parameters. This method only performs two points under Helix4D with a third of its parameters.

3.6 Conclusion

In this chapter, we introduced a novel online inference setting for the semantic segmentation of sequences of rotating LiDAR 3D point clouds. Our proposed large-scale dataset HelixNet contains specific sensor information that allows a rigorous evaluation of the performance and latency of segmentation methods in our online setting. We also introduced Helix4D, a transformer-based network specifically designed for online segmentation, achieving *state-of-the-art* results with a fraction of the latency and parameters of competing methods. We hope that our open-source dataset and implementation will encourage the evaluation of future semantic LiDAR segmentation methods in more realistic settings and help to bridge the gap between academic work on 3D perception and the operational constraints of autonomous driving.

Acknowledgements. This work was supported in part by ANR project READY3D ANR-19-CE23-0007 and was granted access to the HPC resources of IDRIS under the allocation 2022-AD011012096R1 made by GENCI. We thank François Darmon, Tom Monnier, Mathis Petrovich and Damien Robert for inspiring discussions and valuable feedback.

Chapter 4

Representing Shape Collections with Alignment-Aware Linear Models

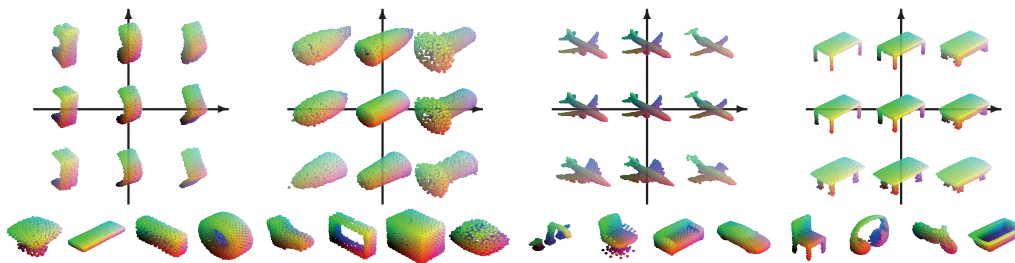


Figure 4.1: **Discovered Linear Models.** Our approach discovers without supervision linear shape models from large collections of shapes. We show two examples of two-dimensional families and eight additional prototypes discovered for ABC [158] (left) and ShapeNet [48] (right).

Abstract

In this chapter, we revisit the classical representation of 3D point clouds as linear shape models. Our key insight is to leverage deep learning to represent a collection of shapes as affine transformations of low-dimensional linear shape models. Each linear model is characterized by a shape prototype, a low-dimensional shape basis and two neural networks. The networks take as input a point cloud and predict the coordinates of a shape in the linear basis and the affine transformation which best approximate the input. Both linear models and neural networks are learned end-to-end using a single reconstruction loss. The main advantage of our approach is that, in contrast to many recent deep approaches which learn feature-based complex shape representations, our model is explicit and every operation occurs in 3D space. As a result, our linear shape models can be easily visualized and annotated, and failure cases can be visually understood. While our main goal is to introduce a compact and interpretable representation of shape collections, we show it leads to *state of the art* results for few-shot segmentation. Code and data are available at: <https://romainloiseau.fr/deep-linear-shapes>.

This chapter’s work was initially presented in:

- [Romain Loiseau](#), Tom Monnier, Mathieu Aubry, Loic Landrieu, “Representing Shape Collections with Alignment-Aware Linear Models”, 3DV, 2021.
- [Romain Loiseau](#), Tom Monnier, Mathieu Aubry, Loic Landrieu, “Representing Shape Collections with Alignment-Aware Linear Models”, ICCV Learning 3D Representations for Shape and Appearance Workshop, 2021.

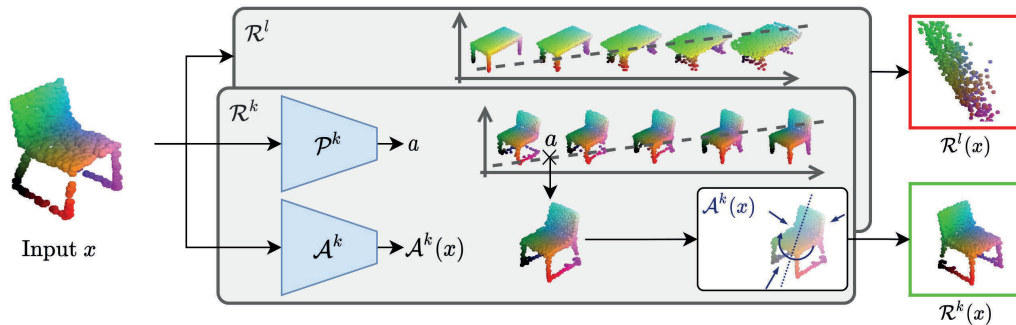


Figure 4.2: **Method Overview.** Given an input point cloud x , we predict for each shape model \mathcal{R}^k the element that best reconstructs the input: the projection network \mathcal{P}^k outputs the coordinates a of a shape in a linear family, and the alignment network \mathcal{A}^k predicts the parameters of an affine transformation $\mathcal{A}^k(x)$ which is applied to the selected shape. The input point cloud is then assigned to the shape model that best reconstructs it, here highlighted in green.

4.1 Introduction

Picture a company acquiring thousands of 3D scans of technical components; how to leverage, organize, or even simply visualize these 3D models? Deep shape analysis techniques have flourished over the last years [125] but, even when motivated by geometric intuitions, these methods and their results remain hard to interpret and interact with. Moreover, they are often limited by the availability of domain and application-specific annotations. Instead of pushing for even more complex architectures, we operate directly in 3D space and revisit the simple linear shape model with a deep learning perspective. As illustrated in Figure 4.1, we model a collection of 3D shapes with a set of low-dimensional linear shape models. Each linear model is defined by a prototype 3D point cloud and a set of vector basis that can be interpreted as fields of translation vectors for each point of the prototype. By adding a linear combination of this basis vector to the prototype, one can continuously move in a low-dimensional subspace of the shape space.

We face three key challenges when trying to represent 3D shape collections with such linear models. First, comparing shapes using Chamfer or Earth Mover distances has strong limitations for shape analysis, since they are impacted by simple rigid or affine shape transformations, which cannot be easily represented by linear models. Transformation-invariant distances such as the Gromov-Hausdorff Distance [221] can be defined to overcome this problem, but

they are typically very hard to work with. Second, finding the coordinates in the shape basis that best reconstruct a sample according to a given similarity measure is a difficult non-convex problem. Third, operations as simple as averaging are non-trivial for point clouds, and dimensionality reduction techniques such as Principal Component Analysis [343] do not directly apply.

In this work, we present an unsupervised approach that learns small sets of linear shape models to explain large collections of point clouds. We propose to solve this task with a clustering formulation directly in 3D space, where clusters are associated to linear shape families, each modeled as a reference prototype point cloud and a set of basis vectors that can be interpreted as displacement fields. We explore two ways of defining such displacement fields—either using a pointwise parametrization or an implicit one based on parametric differentiable functions of 3D space—and analyze their benefits. In addition, to predict the coordinates of a point cloud in the linear basis and account for shape transformations, we extend the idea from the work of Monnier *et al.* [230] on transformation-invariant image clustering to the setting of 3D shape alignment. By jointly learning linear shape families and parametric functions predicting both shape basis coordinates and alignment parameters, our approach is able to discover rich and meaningful shape models from a collection of point clouds without any supervision.

We believe that our method has strong advantages compared to recent unsupervised 3D shape analysis approaches. First, by manipulating objects directly in 3D space, our results are easy to interpret and visualize. Second, our linear shape models can serve as a mean to explore large collections of raw 3D point clouds. Finally, we show that despite its simplicity, our model yields competitive results for shape clustering and *state-of-the-art* results for few-shot shape segmentation.

Our contributions can be summarized as follows:

- we present an unsupervised method to represent large point cloud collections with a small set of linear families of shapes;
- we extend the DTI clustering framework to learn linear shape models by introducing projection networks;
- we analyse two different representations for linear shape modeling and

show the benefits of representing them with continuous functions of space rather than pointwise displacements;

- we demonstrate qualitative results for visualizing the large unstructured ABC dataset [158] and obtain *state-of-the-art* few-shot segmentation performances on the standard ShapeNetPart dataset [48].

4.2 Related Work

Shape reconstruction. Deprelle *et al.* [78] introduces a 3D shape reconstruction model obtained by combining and transforming learned elementary structures. This method shares similarities with ours as it allows learning prototypes of shape parts. However, Deprelle *et al.* uses a fixed number of prototypes to reconstruct inputs, and thus lacks interpretability on the presence and position of parts. Moreover, it focuses on reconstruction accuracy, and mainly follows the AtlasNet [115] deformation framework.

Linear shape modeling. The idea of representing a collection using a low-dimensional image basis was first developed for images of faces [295]. Popularized by the classical eigenfaces model [319], linear models have since been applied to various computer vision problems and data. A linear 3D face model was designed in [30] and applied to new view synthesis. [67] demonstrated applications to medical data. Non-rigid surface-from-motion can also benefit from linear shape basis decomposition to recover 3D shapes [39, 316, 74]. An application of linear modeling to unsupervised 3D keypoint discovery was recently demonstrated in [87]. These linear models are typically learned from a set of examples using principal component analysis, factorization techniques [315], or defined manually [336]. Furthermore, some recent work proposes to analyze shape collections through implicit representations [148, 378, 77]. In contrast, we propose a learning-based approach to model arbitrary unregistered shapes from large collections of examples, and we use several low-dimensional linear families.

4.3 Modeling Shape Collections

Our goal is to explain a collection of N point clouds x_1, \dots, x_N with a small set of K shape models. For simplicity, we assume that all point clouds have the same number M of points. We propose to solve this task with a clustering formulation described in Sec. 4.3.1. We then describe how we model alignment (Sec. 4.3.2) and linear shape families (Sec. 4.3.3) resulting in our final modeling. Finally, we present how we parametrize our linear shape models and give some training details (Sec. 4.3.4).

4.3.1 Method Overview

We build a set of K shape models $\mathcal{R} = \{\mathcal{R}^1, \dots, \mathcal{R}^K\}$. Each \mathcal{R}^k maps a sample point cloud x to a reconstructed point cloud $\mathcal{R}^k(x)$ which can be interpreted as the approximation of x by the corresponding model. We denote by d a distance between point clouds which measures the quality of a reconstruction. We use the Chamfer distance in all our experiments. We learn the shape models \mathcal{R} by minimizing the loss

$$\mathcal{L}(\mathcal{R}) = \sum_{x \in x_1, \dots, x_N} \min_{k=1}^K d(x, \mathcal{R}^k(x)) , \quad (4.1)$$

which can be interpreted as a clustering objective defined as the sum of the reconstruction errors with optimal cluster assignment.

Prototype model. The simplest form of \mathcal{R}^k is a constant function:

$$\mathcal{R}_{\text{proto}}^k(x) = c^k \in \mathbb{R}^{M \times 3} , \quad (4.2)$$

where each c^k can be seen as a prototype point cloud. Such prototype point clouds can be learned by minimizing \mathcal{L} with batch Stochastic Gradient Descent (SGD). This amounts to performing stochastic K-means [36] for 3D point clouds. Note that this is a weak reconstruction model, however, our goal is not to learn the most faithful reconstruction, but rather to summarize the collection.

4.3.2 Alignment-Aware Model

A clear limitation of the prototype model is that it does not take into account simple geometric transformations of the point clouds, such as rigid transformations. For example, point clouds can be close to a model’s prototype c^k according to the distance d , while a rotated or translated version of the same point cloud is far away. We would like both point clouds to be associated with the same shape model. To address this issue, we incorporate in each model \mathcal{R}^k an affine alignment component. In practice, we use neural networks \mathcal{A}^k —which we refer to as *alignment networks*—whose goal is to predict an affine transformation $\mathcal{A}^k(x)$ aligning the prototype c^k with a target point cloud x . This results in an alignment-aware model $\mathcal{R}_{\text{align}}^k$ defined by:

$$\mathcal{R}_{\text{align}}^k(x) = \mathcal{A}^k(x) [c^k] , \quad (4.3)$$

where the affine transformation $\mathcal{A}^k(x)$ is applied to each point of the prototype point cloud c^k . The alignment networks $\mathcal{A}^1, \dots, \mathcal{A}^K$ can be trained alongside the prototypes c^1, \dots, c^K by minimizing Equation 4.1. This model can be seen as an extension of the recent Deep Transformation-Invariant (DTI) clustering framework [230] developed for images to point clouds. Indeed, our alignment models can be understood as defining an approximation of an affine-invariant version of the distance d according to which the clustering is performed. In this part, we rather view these networks as an integral part of the shape models.

Note that different transformation models could be considered. In our experimental analysis, we study variations of the model using weaker transformations, such as rigid transformations or scaling, and show the benefits of the affine model. On the contrary, one could consider complex deformations parametrized by deep networks, such as the ones used in FoldingNet [353] or AtlasNet [115], which would surely lead to higher accuracy reconstructions. However, such transformations completely change the geometry of a point cloud and are hard to interpret.

4.3.3 Linear Shape Modeling

Our goal in this section is to model changes in objects more subtle than those that can be modeled by affine transformations, such as the angle of the wings

of an airplane, while maintaining the model interpretability. We propose to associate a linear shape family to each prototype point cloud.

Linear shape families. For each model k , we define a linear shape family as a pair formed by (i) a prototype point cloud c^k in $\mathbb{R}^{M \times 3}$ and (ii) a set v^k of D basis vectors $v^k = \{v_1^k, \dots, v_D^k\}$, where each $v_i^k \in \mathbb{R}^{M \times 3}$ associates to each point of the prototype a 3D vector and can be interpreted as displacement fields. Each (c^k, v^k) defines a continuous collection of shapes covered by translating the points of c^k along the directions defined by v^k . Each element u of the linear family (c^k, v^k) is characterized by a vector a in \mathbb{R}^D defining its coordinates in the linear shape family:

$$u = c^k + \sum_{i=1}^D a_i v_i^k . \quad (4.4)$$

The vector a can be interpreted as the set of amplitudes to apply to the displacement fields $\{v_1^k, \dots, v_D^k\}$. Note this formally describes an affine space but we follow the convention of previous works and refer to it as linear.¹ Also note that we do not explicitly enforce linear independence between basis vectors, but their high dimensionality ($M \times 3$) leads to such independence in practice.

Projection networks. If we had access to ordered point clouds, *i.e.* lists of M points in \mathbb{R}^3 where the i -th points are in correspondence, we would be able to use the L_2 distance to measure point clouds similarity. In this case, computing the coordinates of the element of the linear family closest to a target point cloud would simply amount to performing Euclidean projection. This is however not the case for unordered point clouds, for which the notion of distance is more complicated. For common point cloud similarity measures such as the Chamfer distance, finding the closest point cloud in a linear family is a difficult non-convex optimization problem. This task is made even harder by the fact that we use our alignment networks to transform the elements of the family before comparing them with the input cloud.

Therefore, we propose to leverage deep learning to estimate which element of a linear family is the closest to a target point cloud after alignment. More specifically, we associate to each linear family (c^k, v^k) a neural network \mathcal{P}^k which aims at associating to a given input sample the coordinates of the ele-

¹An analogy can be made with the face reconstruction model EigenFace [320]: c is equivalent to the *mean face*, and v to the *eigenfaces*.

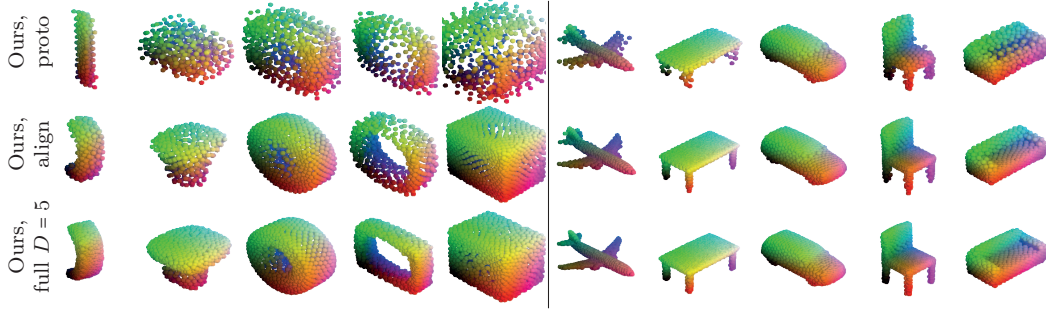


Figure 4.3: **Learned Prototypes and Comparisons.** We compare the prototypes from our different shape modeling discovered in ABC [158] (left, 5 shape models out of 10) and ShapeNetCore [48] (right, 5 shape models out of 55). Note how sharp the prototypes become when the shape modeling complexity increases, respectively with alignment-awareness and 5-dimensional linear families.

ment in the linear family minimizing the distance d . The output of the network $\mathcal{P}^k(x) \in \mathbb{R}^D$ is interpreted as the coordinates a of the point cloud defined in Equation 4.4. By analogy with the L_2 distance case, we refer to these networks as *projection networks*.

Full model. We define our final shape model \mathcal{R} as a collection of models $\mathcal{R}_{\text{full}}^k$ each composed of a linear family (c^k, v^k) , an alignment network \mathcal{A}^k and a projection network \mathcal{P}^k . Given a target point cloud x , our model reconstructs it by (i) selecting an element of the linear family (c^k, v^k) through the projection network \mathcal{P}^k , and (ii) aligning it with the target using the transformation predicted by the alignment network \mathcal{A}^k . More formally, we write each shape model as:

$$\mathcal{R}_{\text{full}}^k(x) = \mathcal{A}^k(x) \left[c^k + \sum_{i=1}^D [\mathcal{P}^k(x)]_i v_i^k \right], \quad (4.5)$$

where $[\mathcal{P}^k(x)]_i$ refers to the i -th component of $\mathcal{P}^k(x)$ and the affine transformation $\mathcal{A}^k(x)$ is applied to each point of the point cloud independently. Again, we optimize jointly the c^k, v^k, \mathcal{A}^k and \mathcal{P}^k to minimize the reconstruction loss defined in Equation (4.1).

4.3.4 Parameterization and Training Details

We first describe how we parametrize the linear families, then provide implementation details such as networks architecture and our curriculum learning

strategy.

Linear family parametrization. While the prototypical point cloud c^k is modeled directly using learnable parameters in $\mathbb{R}^{M \times 3}$, the basis vectors v_i^k can be parametrized in two different ways:

- *Pointwise parametrization:* for each model k , we represent v^k as vectors of learnable parameters of size $D \times (M \times 3)$ that can directly be interpreted as D pointwise displacement vectors of the prototype c^k .
- *Implicit parametrization:* we use implicit parametric functions of the 3D space modeled as neural networks to define the displacement fields. More precisely, for each model k and basis dimension i , we learn a parametric function $\mathcal{V}_i^k : \mathbb{R}^3 \mapsto \mathbb{R}^3$ mapping any point in the 3D space to a displacement direction. Writing $[c^k]_p$ the 3D coordinates of the p -th point of prototype c^k , the 3D coordinates $[v_i^k]_p$ of the i -th basis vector associated to the point p are $[v_i^k]_p = \mathcal{V}_i^k([c^k]_p)$.

Intuitively, the pointwise parametrization seems better suited for modeling complex and discontinuous transformations within a shape family such as the appearance/disappearance of object parts. On the contrary, the transformations learned with implicit parametrizations are derived from continuous functions of the 3D space and can be expected to be more regular.

We compare both settings in Section 4.4.2, and show that pointwise parametrizations provide better shape reconstructions, but that implicit parametrization yields more interpretable transformations, preserving semantic correspondences. Thus, unless specified otherwise, we use the implicit parametrization of the basis in the rest of the chapter.

Architecture. For each model k , the alignment network \mathcal{A}^k takes as input a point cloud and outputs a vector in \mathbb{R}^{12} corresponding to a linear 3D operator and a translation vector applied to each point of the model. The projection network \mathcal{P}^k also takes a point cloud as input and outputs a vector in \mathbb{R}^D that is interpreted as coordinates in the linear family (c^k, v^k) . These networks share a common PointNet [257] backbone encoder which acts as a global feature extractor. This shared encoder starts with a sequence of three linear layers with batch normalization [145] and ReLU activation acting on points independently

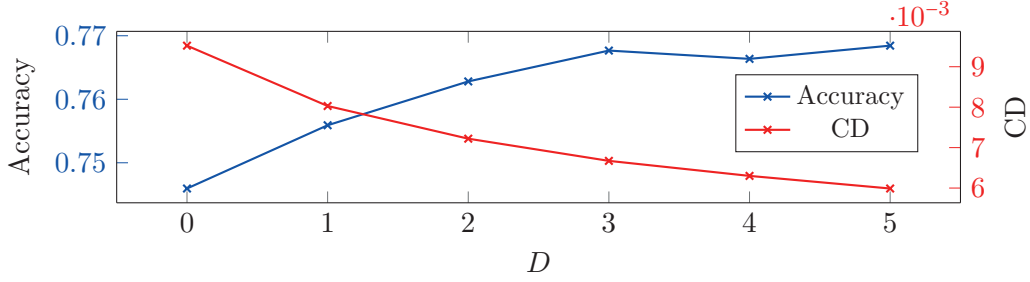


Figure 4.4: **Influence of Degrees of Freedom D on ModelNet.** The reconstruction error (CD) decreases with added degrees of freedom. In contrast, the clustering Accuracy stops increasing when $D \geq 3$, hinting that we have reached a sufficient level of complexity.

and sequentially generating representations of size 64, 128 and 1024, and ends with a max-pooling over all points. This encoder is then followed by $2 \times K$ MLPs corresponding to each prediction task (alignment or projection) and each shape model. Each MLP has one hidden layer of size 128. The implicit parametrizations $\mathcal{V}_i^k : \mathbb{R}^3 \mapsto \mathbb{R}^3$ are MLPs with 2 hidden layers of size 128.

Curriculum learning. Inspired by the curriculum learning strategy of [230], we propose to learn our models by gradually increasing the models complexity. We first learn raw prototype models ($\mathcal{R}_{\text{proto}}^k$), an optimization which corresponds to performing a gradient-based K-means algorithm in the 3D space. Second, we augment each model with alignment awareness ($\mathcal{R}_{\text{align}}^k$). Finally, we gradually increase the linear families dimension up to the desired one, resulting in our final shape model ($\mathcal{R}_{\text{full}}^k$). Curriculum learning allows the model to choose the number of displacement fields D according to the complexity of the studied dataset. Early stopping occurs when the benefit of adding a new degree of liberty (*i.e.* increasing D by one) does not meet a criterion on the loss or on a validation task, see Figure 4.4.

Alignment networks and basis vectors are initially set to identity and zero, respectively. When unfreezing a new module (alignment or a dimension of projection), the learning rate for the new weights is initially set to a tenth of the learning rate applied for the rest of the network, and gradually increased over 50 epochs to the global learning rate. This “warm-up” heuristic helps the network learn more smoothly from one step of the curriculum to the next.

Initialization strategy. As it is the case for many clustering algorithms, initialization can be critical. In our case, we initialize the prototype point clouds with samples of the training set chosen according to a k-means++ strategy [11] with respect to the Chamfer distance.

Cluster reassignment. To prevent empty clusters, we reassign at the end of each epoch any cluster that was selected fewer times than 20% of the expected size of clusters (N/K) in the evenly distributed cluster assignment of Equation 4.1. Clusters are reassigned by selecting and duplicating another cluster. The duplicated cluster is chosen with a probability proportional to the mean of its reconstruction error over the last epoch. To break the symmetry, we add Gaussian noise with variance 10^{-4} to both its prototype and vector basis. The alignment and projection networks are copied without adding noise. We decrease the reassignment threshold tenfold after each curriculum step in order to preserve less populated but expressive clusters.

Training strategy. We use the Adam optimizer [155] with a learning rate of 0.001, a batch size of 64, and neither weight decay nor data augmentation. Our model takes point clouds in $\mathbb{R}^{1024 \times 3}$ as input for all experiments, except for the few-shot segmentation task that takes point clouds in $\mathbb{R}^{2048 \times 3}$ as input.

Implementation details. Our implementation uses PyTorch, Torch-Points3D [50], and an efficient CUDA implementation of the Chamfer distance which significantly speeds up training.

Memory and speed. With $K = 10$ prototypes and $D = 5$, our model has 4.6M parameters. For comparison, the reconstruction models proposed by Wang *et al.* [332] and Groueix *et al.* [116] have respectively 2.6M and 10.0M parameters. Our model can be trained on a single NVIDIA GeForce RTX 2080Ti within a few hours on the 3991 samples of ModelNet10, and in less than a day on ShapeNetCore. Inference on all samples from ShapeNetCore ($\approx 50k$ shapes) takes less than 4 minutes.

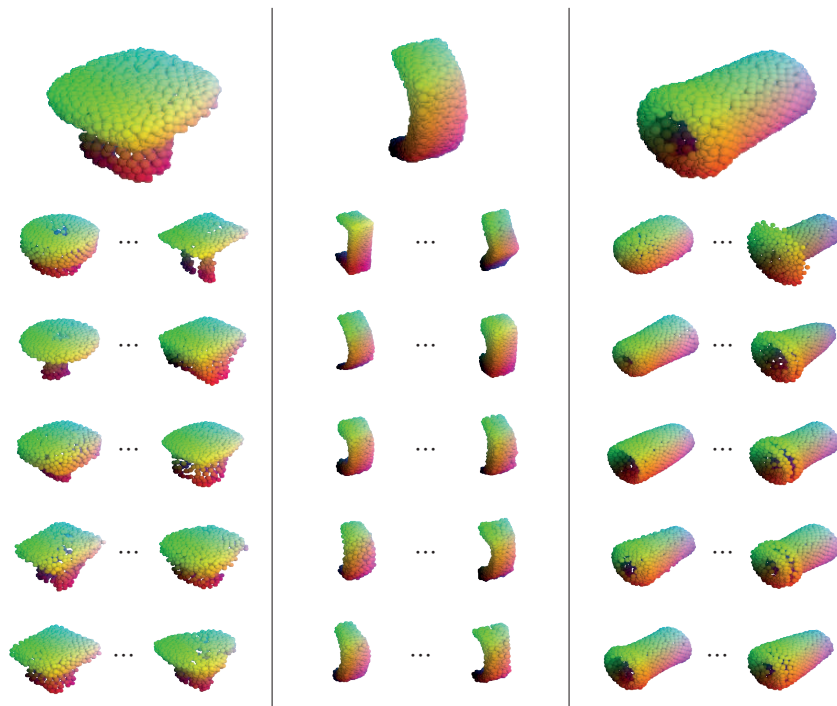


Figure 4.5: **Basis Vectors.** Examples of linear shape models obtained after training on ABC with $D = 5$. The prototype is represented at the top and each row corresponds to one of the dimension of the linear families. The models’ basis vectors correspond to complex morphological changes.

4.4 Experiments

In this section, we analyze the benefits of our method to represent shape collections, first qualitatively (Section 4.4.1) then quantitatively (Section 4.4.2). Finally, we demonstrate that it leads to results on par with state of the art for few-shot and low shot shape segmentation (Section 4.4.3).

4.4.1 Qualitative Results

We demonstrate the potential of our method for exploring large shape collections.

Datasets. The ShapeNet dataset [48] is a large collection of over 50K 3D models organized along 55 common object categories such as chairs, airplanes, or cars. The ABC dataset [158] is a very large collection of Computer-Aided Design (CAD) models of diverse mechanical object parts, such as screws or pipes. We used the first six chunks from this dataset and considered the connected components of each mesh as separate objects ($\approx 70K$ shapes). We

apply our approach using 55 shape models for ShapeNet and 10 for ABC. For both datasets, we uniformly sample points on the objects’ surface to obtain point clouds.

Prototypes. We present in Figure 4.3 examples of prototypes learned when successively adding different components of our method. The first line, denoted “Ours, proto”, represents the linear families’ prototypes learned during the first stage of our training ($\mathcal{R}_{\text{proto}}$). The second line, denoted “Ours, align”, displays the learned prototypes after the second stage of our training ($\mathcal{R}_{\text{align}}$), during which affine alignment networks are learned jointly with their model’s prototype. Finally, the third line denoted “Ours, full $D = 5$ ” illustrates the prototypes learned at the last stage of training ($\mathcal{R}_{\text{full}}$) alongside linear shape families of dimension 5 and their associated projection networks. We show the center of each linear shape model, defined by taking the median amplitude a_i in each dimension i when considering all point clouds associated with the model, *i.e.* point clouds for which this model outputs the best reconstruction.

The prototypes learned with the Chamfer distance (first line) appear noisy, hinting that they are not well aligned with the shapes they try to approximate. When adding alignment networks, we obtain the prototypes of the second line, which are much cleaner, outlining the interest of using a transformation-invariant model, as well as the fact that our approach can effectively learn such a model. Finally, the prototypes obtained with our full method are even sharper and smoother, indicating that linear shape families can better model the associated point clouds.

Our results on the ABC dataset outline the capacity of our full model to differentiate between different types of shapes, as prototypes correspond to different object types. By looking at the prototypes, one can grasp at a glance the diversity of shapes contained in this large-scale dataset.

Linear shape models. In Figure 4.5, we illustrate some of the linear shape models learned on ABC. The top row shows the center of the linear shape models, and the subsequent lines illustrate the five basis vectors. For each model and each basis vector, we represent two shapes whose amplitudes for the considered dimensions are set to the 5-th and 95-th percentile values of all point clouds associated to the model, while the other amplitudes remain at the

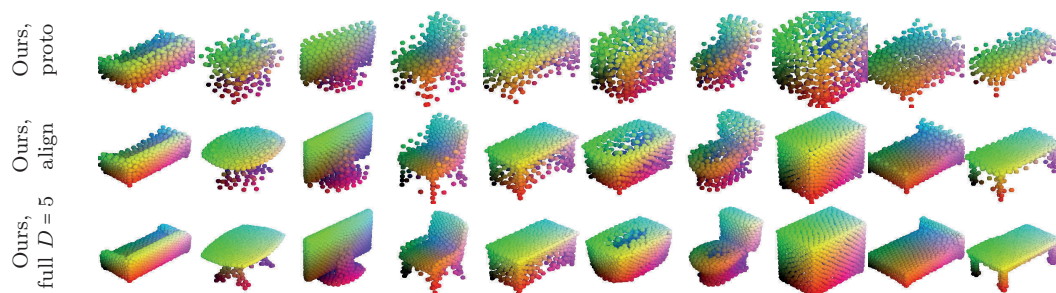
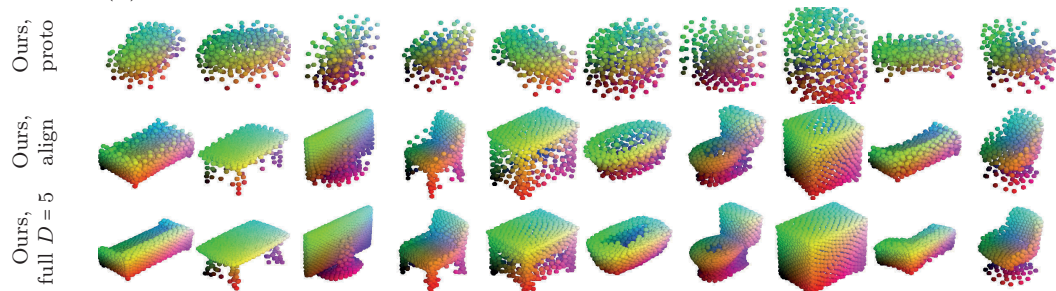
(a) 10 prototypes learned from the **aligned ModelNet10** dataset.(b) 10 prototypes learned from the **rotated ModelNet10** dataset.

Figure 4.6: **Modeling ModelNet10.** Prototype learned on ModelNet’s [347] aligned version (a) and with random z -axis rotations (b). In this figure, the models are manually rearranged to be in correspondence across the two experiments. Note how our model without alignment networks (“Ours, proto”) is unable to learn meaningful prototypes on un-aligned data. In contrast, our models with alignment networks learn sharp and informative prototypes despite the rotations. This shows that alignment networks allow our model to handle a raw, un-aligned dataset to produce a compact overview of its shape diversity.

median value. Again, we can see how the different dimensions give insights on the diversity of shapes within the dataset.

We represent in Figure 4.6 the models learned on ModelNet with random rotations. We observe that when alignment networks are used, the obtained prototypes are similar to the ones obtained on the aligned version of the dataset. This shows that our approach can be used successfully on raw, un-aligned datasets.

Reconstructions. In Figure 4.10, we show examples of reconstructed shapes from ShapeNet (airplanes, cars, and chairs) for three different linear shape families. As expected, the model is able to reconstruct objects precisely while remaining visually interpretable. We show some reconstruction results in Fig-

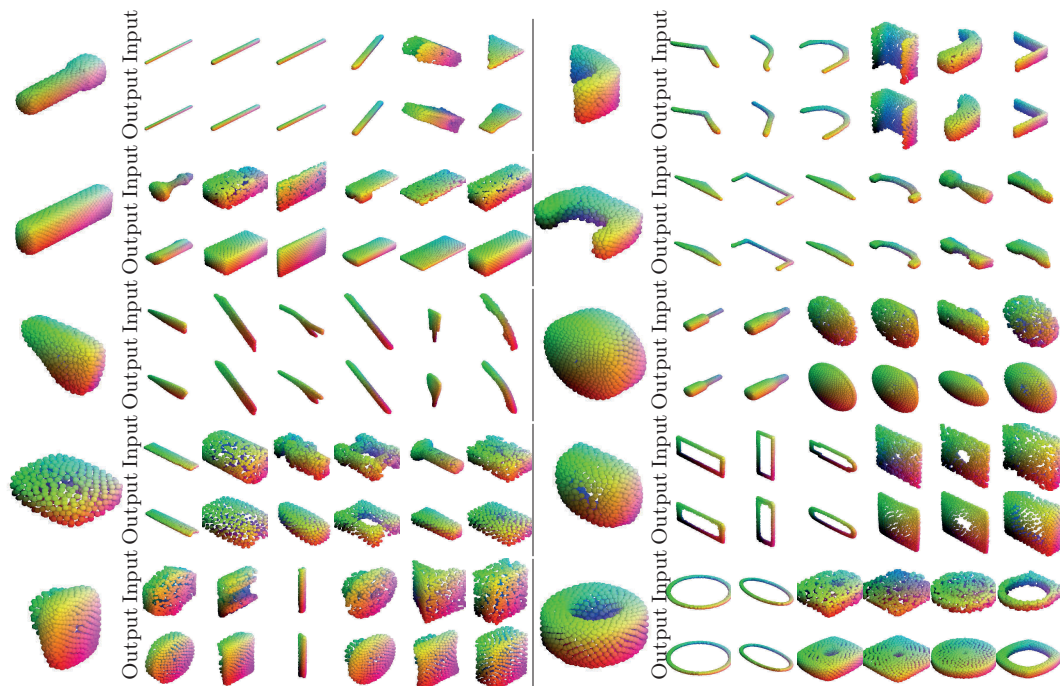


Figure 4.7: **Visualizing Reconstructions on ABC.** The left-most columns represent prototypes from all 10 linear models learned on the ABC dataset. For each prototype, we select 6 samples for which this model gives the best reconstruction (“Input”, top line). We then represent the associated reconstruction provided by the model (“Output”, top line). Each family represents a wide variety of morphologically homogeneous shapes: round rings, square rings, bent archs, cylinders, etc... Looking at the prototypes gives us a concise overview of the shape diversity.

ure 4.7 and Figure 4.8 for ABC and ShapeNet, respectively. For each model, we represent some sample shapes for which the model provides the reconstruction with the lowest error. Viewing our approach in terms of clustering, this amounts to showing elements from the clusters associated with each model. Note that in Figure 4.8, our linear models are associated with rich subsets of shapes which remain mostly semantically homogeneous.

4.4.2 Quantitative Analysis for Clustering and Reconstruction

The qualitative results described in the previous section outline the potential of our approach for visualizing and analyzing large, unstructured, and diverse shape collections. We now provide a more quantitative analysis of these results on the standard ModelNet10 dataset [347].

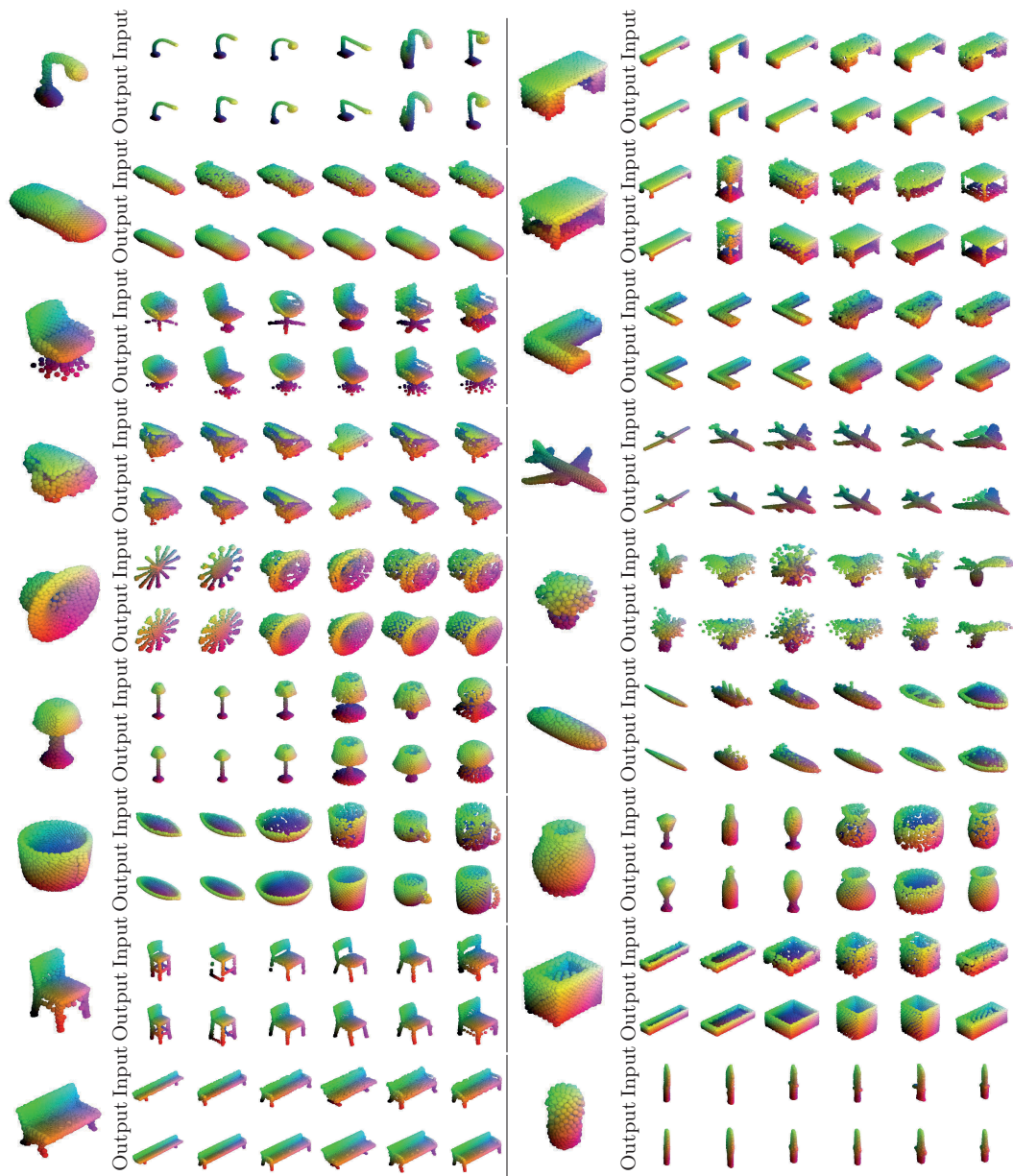


Figure 4.8: **Visualizing Reconstructions on ShapeNet.** The left-most columns represent prototypes from some of the 55 linear models learned on the ShapeNet dataset. For each prototype, we select 6 samples for which this model gives the best reconstruction (“Input”, top line). We then represent the associated reconstruction provided by the model (“Output”, top line). We observe that the samples associated with a given model are for the most part semantically homogeneous, and well represented by their prototype.

Data and evaluation. ModelNet10 contains 3991 train and 909 test aligned 3D point clouds obtained from CAD models of 10 different classes. We use this dataset both in its original aligned version and also with added random rotations around the z-axis to evaluate the capacity of our method to represent

Table 4.1: **Results on ModelNet10.** We present results with 10 linear shapes models, first for different restrictions of the alignment networks, then for different basis vector configurations. The steps in the curriculum training of our model are in **bold**. We report clustering accuracy in % (“Accuracy”) and the Chamfer distance multiplied by 10^3 (“CD”), Results are averaged over five runs.

| | | Accuracy | CD |
|----------------------|---------------------------------|-------------------|-------------------|
| Ours, proto | | 63.9 ± 1.5 | 20.0 ± 0.4 |
| ... with supervision | | 79.0 ± 0.2 | 23.5 ± 0.0 |
| Ours, align | Rigid transformation (6D) | 64.6 ± 5.2 | 16.2 ± 0.1 |
| | Trans. + Iso. Scaling (4D) | 71.5 ± 4.1 | 15.0 ± 0.1 |
| | Trans. + Aniso. Scaling (6D) | 74.1 ± 3.0 | 10.4 ± 0.1 |
| | Linear (9D) | 71.85 ± 4.7 | 11.1 ± 0.1 |
| | Affine (12D) | 75.9 ± 3.0 | 9.7 ± 0.0 |
| | ... with supervision | 88.9 ± 0.5 | 11.2 ± 0.0 |
| Ours, full | $D=1$ Pointwise parametrization | 74.3 ± 1.7 | 7.9 ± 0.0 |
| | Implicit parametrization | 77.5 ± 2.8 | 8.1 ± 0.0 |
| | ... with supervision | 89.7 ± 0.6 | 9.5 ± 0.0 |
| | $D=5$ Pointwise parametrization | 75.1 ± 1.7 | 5.7 ± 0.0 |
| | Implicit parametrization | 77.0 ± 3.4 | 5.9 ± 0.0 |
| | ... with supervision | 90.4 ± 1.0 | 7.8 ± 0.0 |
| FoldingNet [353] | | 76.3 ± 7.5 | 3.5 ± 0.0 |

unaligned data. Unless specified otherwise, the results are given for the original dataset. We trained the different variants of our method with 10 reconstruction models on train and test shapes of ModelNet10. We evaluate in Table 4.1 the clustering accuracy and reconstruction error measured by the Chamfer Distance. To measure the quality of the resulting clustering, we assign to each model the majority label of its associated point clouds from the train set. The accuracy of the classification is then defined by assigning to test shapes the label of the model giving the best reconstruction.

Alignment. We compute the performance of our models only defined by prototypes (“Ours, proto”), and then train models with alignments of different complexities (“Ours, align”). We first evaluate a model whose alignment networks are restricted to a rigid transformation (“Rigid transformation (6D)”), with rotations parametrized with quaternions. We also evaluate models with a

Table 4.2: **Non-Aligned Data.** Clustering Accuracy (“Accuracy”, in %) and reconstruction error (“CD”, Chamfer distance multiplied by 10^3) obtained with 10 linear shapes models on the rotated version of ModelNet10. Δ_{CD} is the difference of reconstruction error when training the same model on the aligned or unaligned datasets.

| | Accuracy | Δ_{Accuracy} | CD | Δ_{CD} |
|--------------------|----------------------------------|----------------------------|---------------------------------|----------------------|
| Ours, proto | 41.2 ± 3.4 | -22.7 | 30.1 ± 0.1 | -10.1 |
| Ours, align | 61.8 ± 3.3 | -14.1 | 11.0 ± 0.1 | -1.3 |
| Ours, full $D = 1$ | 65.2 ± 6.7 | -12.3 | 9.3 ± 0.0 | -1.2 |
| Ours, full $D = 5$ | 68.8 ± 7.9 | -8.2 | 6.7 ± 0.0 | -0.8 |

scaling and a translation (“Trans. + Iso. Scaling (4D)”), axis-aligned scalings and a translation (“Trans + Aniso. Scaling (6D)”), a linear transformation (“Linear (9D)”), and finally an affine transformation (“Affine (12D)”). We observe that using alignment networks allows significant clustering improvement in terms of accuracy and reconstruction quality. Moreover, restricting the output of the alignment networks leads to a lower performance: even for centered and rotation-aligned data such as ModelNet, allowing complex alignments benefits both clustering and reconstruction.

Linear families. We then evaluate models with affine alignment but different linear basis (“ours, full”). We compare the results between one-dimensional ($D = 1$) and five-dimensional ($D = 5$) linear families as well as between basis vectors learned in the pointwise and implicit parametrization (see Section 4.3.3). Increasing the dimension of the shape families improves the reconstruction error but slightly decreases the clustering accuracy with the implicit parametrization. This can be explained by the models becoming too expressive, resulting in point clouds from different classes being associated with the same model.

Baseline and supervised upper bound. As a baseline, we performed k-means clustering in feature space using the implementation of FoldingNet [353] proposed by [308]. The resulting accuracy is comparable to that of our best models’. However, FoldingNet relies on learning black-box deep deformations of a planar patch, and the resulting shape family and generation process are thus harder to interpret than ours.

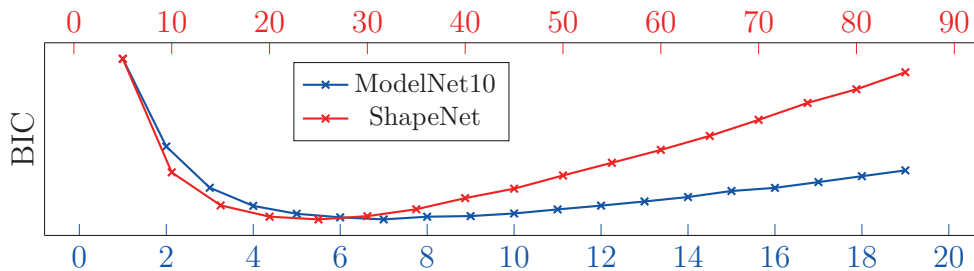


Figure 4.9: **Model Selection.** We can select the number of clusters K using the BIC. We obtain 7 clusters for ModelNet and 30 for ShapeNet, which is consistent with the shapes’ diversity.

We also trained our model in a supervised manner by associating a class to each model, and only training each model on point clouds from their class (“with supervision” lines, in light gray). As expected, this “oracle” setting performs better in terms of clustering accuracy, but with lower reconstruction quality. This can be explained by the presence of classes with high variability such as *chairs* which require several families to fully cover, and similar classes such as *desks* and *tables* which can be well reconstructed by a single family.

Non-aligned data. In Table 4.2, we report our approach’s performance when trained on ModelNet10 with random rotations. We observe that adding alignment networks to the model results in significantly better metrics compared to simple prototypes. Our full models with alignment are able to reach reconstruction qualities almost comparable to the equivalent models trained on aligned shapes. Similarly, the drop in clustering performance is reduced when adding the alignment networks and linear shape families. This outlines the capacity of our models to handle raw unaligned data. We present in Figure 4.6 illustrations of the prototypes learned in both setting.

Choice of K . The number of models can be automatically selected through usual model selection heuristics such as the Bayesian Information Criterion (BIC), as we show in Figure 4.9. Being entirely unsupervised, there is no restriction on how linear families relate to classes: complex classes can be represented by several models, and similar classes by a single family. However, as demonstrated in our clustering experiments, when the number of classes and models are the same, linear families and classes tend to be assigned on a one-to-one basis

Table 4.3: **10-shot Segmentation.** We report pointwise IoU for 9 classes and the average IoU over all 16 classes of ShapeNetPart. See text for details.

| | airplane | bag | cap | car | chair | lamp | laptop | mug | table | avg |
|----------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Shared encoder | | | | | | | | | | |
| Gadella <i>et al.</i> 2020 [95] | — | — | — | — | — | — | — | — | — | 74.1 |
| Ours, full $D = 5$ (random) | 71.7 | 70.6 | 84.0 | 62.1 | 78.8 | 68.7 | 93.1 | 87.5 | 70.6 | 72.5 |
| Ours, full $D = 5$ (prototype) | 79.4 | 73.0 | 81.8 | 72.1 | 83.6 | 76.1 | 94.7 | 89.8 | 76.2 | 77.4 |
| One encoder per class | | | | | | | | | | |
| Wang <i>et al.</i> 2020 [332] | 67.3 | 74.4 | 86.3 | — | 83.4 | 68.7 | 93.8 | 90.9 | 74.2 | — |
| Groueix <i>et al.</i> 2019 [116] | 67.1 | — | — | 61.4 | 78.9 | 65.8 | — | — | 66.1 | — |
| Ours, full $D = 5$ (random) | 72.2 | 66.0 | 75.5 | 63.0 | 79.1 | 68.9 | 93.1 | 84.2 | 69.4 | — |
| Ours, full $D = 5$ (prototype) | 80.0 | 79.7 | 76.1 | 72.0 | 83.6 | 77.1 | 94.9 | 91.1 | 75.9 | — |

4.4.3 Application to Few/Low-Shot Segmentation

Our linear shape models can perform semantic segmentation by transferring point labels from the model’s prototype to the reconstructed point cloud. More precisely, given an input point cloud x , we identify the model k with the lowest reconstruction error. We then compute $\tilde{x} = \mathcal{R}_{\text{full}}^k(x)$, the point cloud reconstructed by this model. We transfer the point annotation from the prototype c^k to \tilde{x} . Finally, each point of x is assigned the label of the closest point of \tilde{x} . This strategy is especially meaningful in a few-shot setting, since only the prototypes need to be annotated.

Few-shot segmentation. In this setting, where we only use a few annotations for each class and train our model with only the reconstruction loss as described earlier, we consider two methods to annotate the prototypes:

- *Random.* We randomly pick one sample from the train set for each model and propagate its labels to their nearest points of the aligned prototype.
- *Prototype.* We align all samples from the train set for each model’s prototype and label each point with majority voting. This second setting is meant to emulate the *manual* annotation of the 10 prototypes. While this is not directly comparable to other approaches, it outlines the crucial advantage given by our approach, which identifies a small set of prototype shapes that can be annotated instead of using random samples. Some prototypes annotated in this manner can be seen in Figure 4.10.

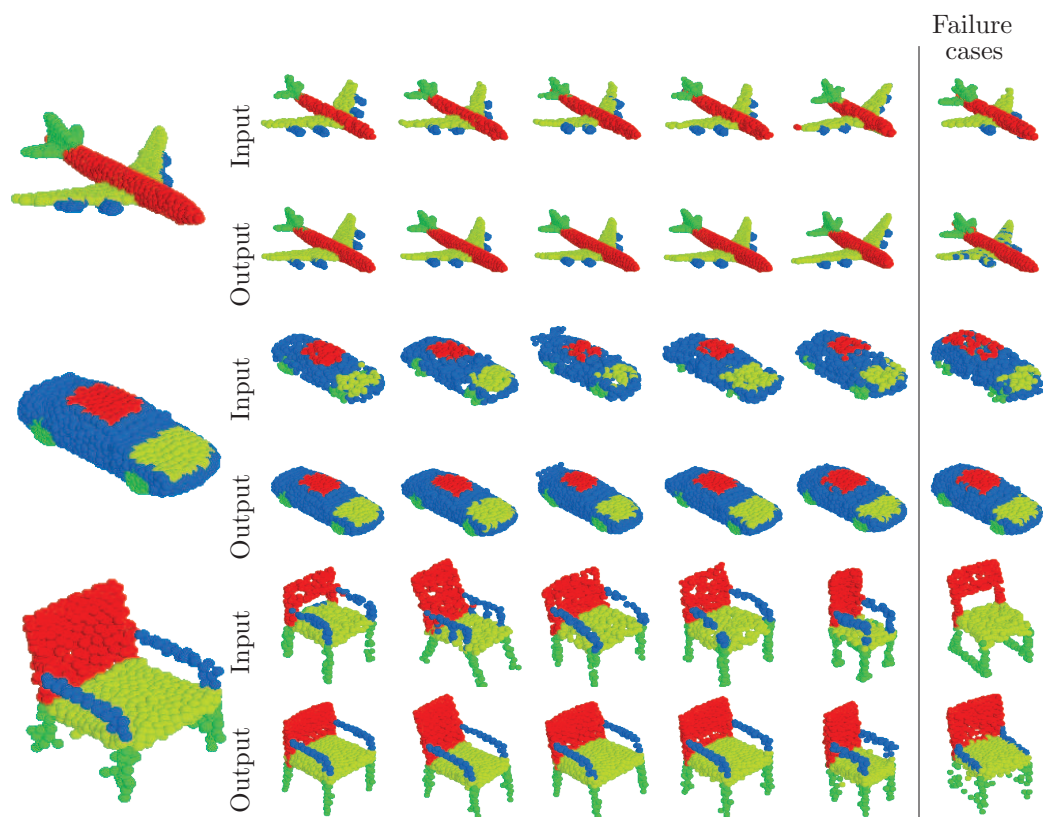


Figure 4.10: **Reconstruction Results.** Examples of samples annotated pointwise with our semantic-segmentation method ($D = 5$). We visually selected failure cases where semantic regions were wrongly predicted. Prototypes are represented on the left column, the “input” lines display input samples with ground truth annotations and the “output” lines our reconstruction with pixel labels propagated from the prototype.

We use the densely annotated ShapeNetPart [280] to evaluate the segmentation performance of our few-shot segmentation scheme. We report in Table 4.3 the performance of our 10-shot segmentation scheme for nine classes of ShapeNetCore, and the average performance over all 16 classes. As mentioned in Section 4.3.4, all the alignment and projection networks share a common PointNet [257] encoder which acts as a global feature extractor. To compare with previous works that use either a shared model or a different model per class, we present results using either a single encoder for all classes or one encoder per class. Using only 10 samples from the dataset to annotate our prototypes, we observe that the annotation from random samples performs on par or better than *state-of-the-art* approaches. Annotating prototypes (using all training samples) significantly outperforms all methods. This shows that our approach can be used to precisely and densely annotate large shape datasets with minimal human intervention. We also observe some failure cases

Table 4.4: **Low-Shot Supervised Segmentation Results on ShapeNet-Part.** We report the IoU averaged over all classes.

| Training data | SONet [173] | 3D-PointCapsNet [375] | Ours, full $D = 5$ |
|---------------|-------------|-----------------------|--------------------|
| 1% | 64 | 67 | 68 |
| 5% | 69 | 70 | 72 |

shown in the last column of Figure 4.10: since our model can only move points and not add or subtract them, shapes with optional parts, such as the arms of chairs, may be mislabeled.

Low-shot semantic segmentation. Our model can also be trained in a low-shot setting, and can learn to perform semantic segmentation from a small number of annotated examples. We first initialize a set number of prototypes per class with random examples from the training set. This allows us to associate each prototype’s point with a *part* semantic label. We then perform our standard training scheme, but with an altered Chamfer distance, which can only match points with the same part label from the true and reconstructed point clouds. At inference time, we can associate a part label to each point of the input shape by taking the points’ closest neighbor in their reconstructed shape. This setting is supervised in the sense that we use the point labels explicitly during training. As presented in Table 4.4, our model trained on only 1 and 5% of the annotated shapes yields an improvement of +1 and +2 average IoU points respectively, compared to 3D-Capsule [375]. In contrast to this more complex model, our linear shape models remain viewable and interpretable.

4.5 Conclusion

We presented a new take on linear shape models with deep learning, representing large un-annotated collections of 3D shapes. Our alignment-aware model produces concise, expressive and interpretable overviews of unaligned point clouds collections. We show that our method leads to *state-of-the-art* results for few-shot segmentation.

Acknowledgements. This work was supported in part by ANR project Ready3D ANR-19-CE23-0007 and HPC resources from GENCI-IDRIS (Grant 2020-AD011012096). We thank François Darmon and Yang Xiao for inspiring discussions and valuable feedback.

Chapter 5

Learnable Earth Parser: Discovering 3D Prototypes in Aerial Scans

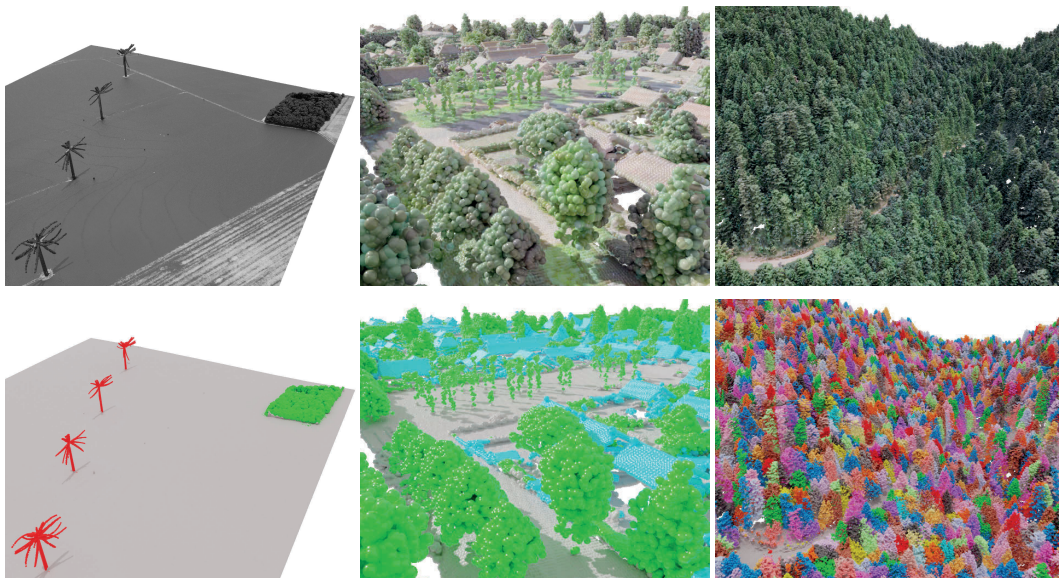


Figure 5.1: **Learnable Earth Parser.** Our unsupervised method takes large aerial 3D scans as input and model them with a small set of learned prototypes. Our approach is trained without annotation and produces legible decomposition of complex scenes, which can be used for semantic and instance segmentation.

Abstract

We propose an unsupervised method for parsing large 3D scans of real-world scenes into interpretable parts. Our goal is to provide a practical tool for analyzing 3D scenes with unique characteristics in the context of aerial surveying and mapping, without relying on application-specific user annotations. Our approach is based on a probabilistic reconstruction model that decomposes an input 3D point cloud into a small set of learned prototypical shapes. Our model provides an interpretable reconstruction of complex scenes and leads to relevant instance and semantic segmentations. To demonstrate the usefulness of our results, we introduce a novel dataset of seven diverse aerial LiDAR scans. We show that our method outperforms *state-of-the-art* unsupervised methods in terms of decomposition accuracy while remaining visually interpretable. Our method offers significant advantage over existing approaches, as it does not require any manual annotations, making it a practical and efficient tool for 3D scene analysis. Our code and dataset are available at <https://romainloiseau.fr/learnable-earth-parser>.

This chapter’s work was initially presented in:

- [Romain Loiseau](#), Elliot Vincent, Mathieu Aubry, Loic Landrieu, “Learnable Earth Parser: Discovering 3D Prototypes in Aerial Scans”, arXiv, 2023.

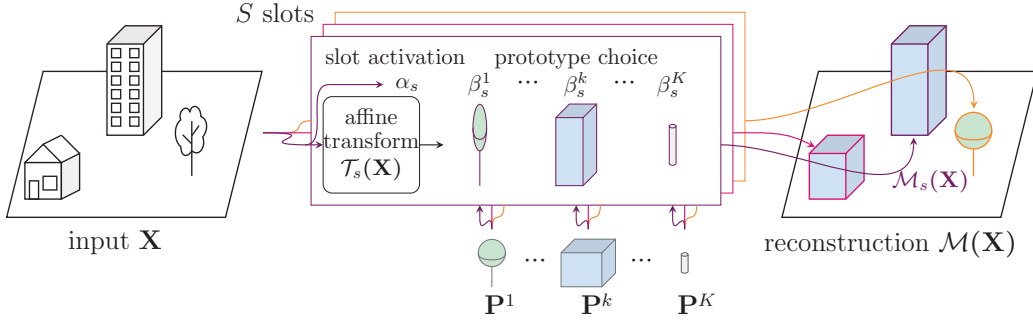


Figure 5.2: **Method Overview.** Our model approximates an input point cloud \mathbf{X} with S slot models. Each slot maps \mathbf{X} to an affine 3D deformation $\mathcal{T}_s(\mathbf{X})$, a slot activation probability α_s , and the joint probabilities $\beta_s^1, \dots, \beta_s^K$ of the slot being activated and choosing one of the K prototype point clouds $\mathbf{P}^1, \dots, \mathbf{P}^K$. The output $\mathcal{M}_s(\mathbf{X})$ of an activated slot s is obtained by applying the transformation $\mathcal{T}_s(\mathbf{X})$ to its most likely prototype. Non-activated slots do not contribute to the output.

5.1 Introduction

Modern aerial 3D scanning technologies open up unprecedented opportunities for environmental monitoring and economic intelligence. However, their practical use remains challenging due to the complexity of real-world scenes, the diversity of usage scenarios, and the difficulty of annotation. Therefore, our aim is to develop an approach that could help perform tasks as diverse as counting trees in a forest, identifying the various components of a factory, measuring the surface of greenhouses or monitor urban growth, all in an unsupervised fashion.

To do so, we address two important limitations of existing 3D deep learning methods. First, they are often primarily designed, trained, and tested on synthetic [347, 48, 280, 216] or highly curated data [9, 186, 157], which fail to capture the endless variability of the real world. Moreover, they often assume that annotations are available for tasks of interest. Second, even unsupervised approaches [4, 370] often rely on learning abstract feature representations, making them difficult to interpret [369]. Although some work has attempted to decompose 3D shapes into meaningful components without supervision [317, 78, 252, 195], they were all designed on simple synthetic shapes and none generalizes to real data.

To overcome these limitations, we present the Learnable Earth Parser, an unsupervised deep learning method designed to decompose large-scale 3D point

clouds into interpretable parts. Our model learns a small set of 3D prototypical shapes that are selected, positioned, rotated, and resized to reconstruct an input point cloud. We introduce a novel probabilistic formulation that enables the design of a reconstruction loss for training jointly the selection of prototypes with the rest of the model.

To evaluate the effectiveness of our approach, we created a new open-access dataset consisting of 7 aerial LiDAR scans, covering 7.7km² and containing 98 million 3D points with annotations in diverse urban and natural environments. Our results demonstrate that the Learnable Earth Parser learns decompositions superior to traditional and deep learning baselines, leading to convincing performance for semantic and instance segmentation, as shown in Figure 5.1.

We believe that our contributions provide researchers and practitioners with new tools and resources to tackle the challenges of real-world 3D data.

5.2 Related Work

Our proposed unsupervised method uses point-cloud reconstruction to learn to decompose large aerial point clouds and is evaluated on a novel and diverse dataset of 3D scans. In the following, we briefly present related work for primitive-based point cloud decomposition and automatic decomposition of LiDAR data.

Aerial LiDAR datasets. We propose a dataset that spans 7.7km², which is of similar scale than prominent datasets [240, 355, 98, 383] of varying sizes from 1 to 10 km² [325, 294]. In contrast to these existing datasets, our Earth Parser Dataset covers a variety of urban, natural, and rural scenes, making it more representative of the diversity of possible usage scenarios.

LiDAR scan decomposition. Automatically parsing large LiDAR scans poses unique challenges due to the size and diversity of the acquisitions [349]. Previous approaches used simple shape primitives, such as lines [120, 70], planes [236, 100, 118], or volumes [177]. These may not be expressive enough to capture the complexity of scanned objects. Other approaches are designed for specific object classes, such as trees [219] or buildings [163, 137], but remain limited in their ability to represent a wide range of shapes. In contrast,

our Learnable Earth Parser overcomes these limitations by learning ad-hoc prototypes for each new scene, ensuring both expressivity and adaptability.

LiDAR data are often treated as digital elevation models, *i.e.* images with pixel elevations [127, 193, 119]. Thus, our work is related to image-based primitive prediction [268, 126, 156, 251] and unsupervised multi-object image segmentation [192, 296, 231, 360]. However, 3D point clouds have higher precision and can better represent multi-layered structures such as forest areas.

Being able to decompose real-world scenes with unknown characteristics and jointly learning primitive shapes and parsing would constitute a significant step towards more realistic unsupervised primitive discovery.

5.3 Method

Our goal is to learn to break down a point cloud into simpler and more easily understandable components. To achieve this, we propose an *analysis-by-synthesis* approach where we train a highly-constrained model \mathcal{M} to approximate a point cloud as a combination of simple 3D shapes.

5.3.1 Probabilistic Scene Reconstruction Model

In this section, we first define our model, which selects up to S shapes among K prototypes and deform each to best approximate an input point cloud \mathbf{X} , as illustrated in Figure 5.2. We then explain how we can model the selection as a probabilistic process, which will enable us to define meaningful training losses. This can be seen as an extension of the model of Paschalidou *et al.* [252] to multiple primitives.

Scene reconstruction model. Our full model \mathcal{M} is the combination of S reconstruction models \mathcal{M}_s , which we refer to as *slots* in analogy to the Slot Attention approach [192]. Each slot contributes to the final reconstruction only if it is activated. Slot activation is determined by a binary variable a_s : \mathcal{M}_s is activated if and only if $a_s = 1$. The output of $\mathcal{M}(\mathbf{X})$ is the combination of the reconstructions from all activated slots:

$$\mathcal{M}(\mathbf{X}) = \bigcup_{\substack{s=1 \dots S \\ a_s=1}} \mathcal{M}_s(\mathbf{X}) . \quad (5.1)$$

Each slot model outputs a point cloud which we define as the deformation of one of the K learnable prototype 3D point clouds $\mathbf{P}^1, \dots, \mathbf{P}^K$ shared across all slots. We associate to each slot s a network \mathcal{T}_s which maps \mathbf{X} to an affine transformation in 3D space $\mathcal{T}_s(\mathbf{X})$. The output $\mathcal{M}_s(\mathbf{X})$ of slot s is determined by a variable $b_s \in \{1, \dots, K\}$. If $b_s = k$, then the output of $\mathcal{M}_s(\mathbf{X})$ is \mathbf{Y}_s^k , the result of applying the transformation $\mathcal{T}_s(\mathbf{X})$ to the prototype \mathbf{P}^k :

$$\mathbf{Y}_s^k = \mathcal{T}_s(\mathbf{X})[\mathbf{P}^k] . \quad (5.2)$$

Please note that \mathbf{Y}_s^k is a function of \mathbf{X} . However, to keep our notations simple, we omit this dependence.

Probabilistic modeling. We make our reconstruction model probabilistic by modeling a and b as random variables following (multi-)Bernoulli distributions. We call α_s the probability the slot s is activated and β_s^k the probability that it is activated and selects the prototype k , leading to a reconstruction \mathbf{Y}_s^k :

$$p(a_s = 1) = \alpha_s \quad (5.3)$$

$$p(a_s = 1, b_s = k) = \beta_s^k . \quad (5.4)$$

For each slot, we predict the vector $(1 - \alpha_s, \beta_s^1, \dots, \beta_s^K)$ with a neural network taking the point cloud \mathbf{X} as input and finishing with a softmax layer. Again, we don't write the dependency of the α_s and β_s on \mathbf{X} explicitly to simplify the notations.

The full model $\mathcal{M}(\mathbf{X})$ and the slots models $\mathcal{M}_s(\mathbf{X})$ can now be seen as random variables, producing different potential reconstructions with probabilities given by α and β . During inference, we consider only slots with $\alpha_s > 0.5$ and select the prototype with highest β_s^k . However, during training, we compute all reconstructions \mathbf{Y}_s^k .

5.3.2 Training Losses

Given a large 3D scene we train our model by sampling square patches \mathbf{X} from the scene. For each batch of patches, we minimize a loss composed of a reconstruction loss \mathcal{L}_{rec} and several regularization terms \mathcal{L}_{reg} implementing

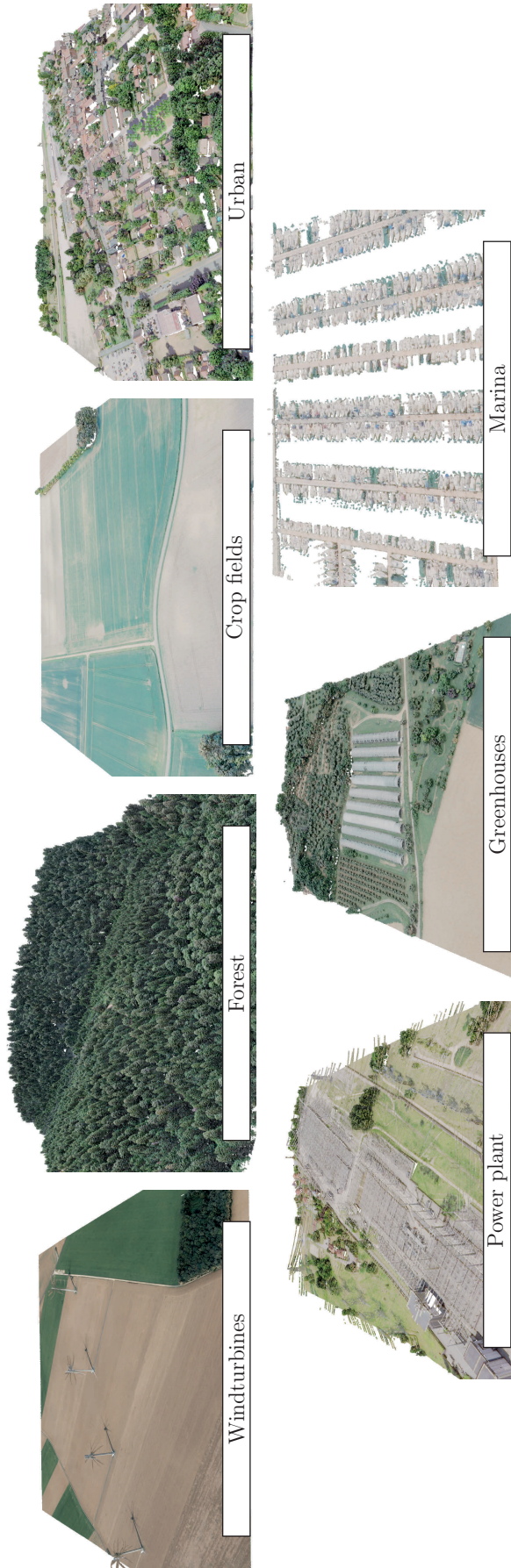


Figure 5.3: **Earth Parser Dataset.** Our proposed dataset contains 7 scenes representing various urban and natural environments acquired by aerial LiDAR. The illustration of the power plant and the greenhouses display the complete scenes, while other ones display a subset of each scene covering between 25 and 50% of the total area.

different priors:

$$\mathcal{L}(\mathcal{M}) = \mathbb{E}_{\mathbf{X}} [\mathcal{L}_{\text{rec}}(\mathcal{M}, \mathbf{X})] + \mathcal{L}_{\text{reg}}(\mathcal{M}) . \quad (5.5)$$

Reconstruction loss. We define the reconstruction loss \mathcal{L}_{rec} as the sum of two losses:

$$\mathcal{L}_{\text{rec}}(\mathcal{M}, \mathbf{X}) = \mathcal{L}_{\text{acc}}(\mathcal{M}, \mathbf{X}) + \mathcal{L}_{\text{cov}}(\mathcal{M}, \mathbf{X}) . \quad (5.6)$$

\mathcal{L}_{acc} encourages likely reconstructions of $\mathcal{M}(\mathbf{X})$ to accurately approximate \mathbf{X} , and \mathcal{L}_{cov} ensures coverage, i.e., that each points of \mathbf{X} is well-reconstructed by at least one activated models. We define each term using the asymmetric Chamfer distance d between two point clouds \mathbf{X} and \mathbf{Y} defined as:

$$d(\mathbf{X}, \mathbf{Y}) = \frac{1}{|\mathbf{X}|} \sum_{x \in \mathbf{X}} \min_{y \in \mathbf{Y}} \|x - y\|_2^2 , \quad (5.7)$$

with $|\cdot|$ the number of points in a point cloud. We write $d(x, \mathbf{Y})$ the distance between the point x and its closest point in \mathbf{Y} , considering x as a point cloud containing a single point.

We define \mathcal{L}_{acc} as the average over all slots s of the expected distance between $\mathcal{M}_s(\mathbf{X})$ and \mathbf{X} :

$$\mathcal{L}_{\text{acc}}(\mathcal{M}, \mathbf{X}) = \frac{1}{S} \sum_{s=1}^S \mathbb{E}_{a,b} [d(\mathcal{M}_s(\mathbf{X}), \mathbf{X})] \quad (5.8)$$

$$= \frac{1}{S} \sum_{s=1}^S \sum_{k=1}^K \beta_s^k d(\mathbf{Y}_s^k, \mathbf{X}) . \quad (5.9)$$

Conversely, we define \mathcal{L}_{cov} as the average over all points x of \mathbf{X} of the expected distance between x and its closest point in the reconstruction:

$$\mathcal{L}_{\text{cov}}(\mathcal{M}, \mathbf{X}) = \frac{1}{|\mathbf{X}|} \sum_{x \in \mathbf{X}} \mathbb{E}_{a,b} \left[\min_{s|a_s=1} d(x, \mathcal{M}_s(\mathbf{X})) \right] . \quad (5.10)$$

Following the ideas of Paschalidou *et al.* [252], we first define $\Delta(x, s)$ as the expected distance between x and $\mathcal{M}_s(\mathbf{X})$ conditionally to the slot s being

activated:

$$\Delta(x, s) = \mathbb{E}_{b_s | a_s=1} [d(x, \mathcal{M}_s(\mathbf{X}))] \quad (5.11)$$

$$= \frac{1}{\alpha_s} \sum_{k=1}^K \beta_s^k d(x, \mathbf{Y}_s^k) . \quad (5.12)$$

Next, we compute for each point x a permutation σ_x of $[1, S]$ such that $\Delta(x, \sigma_x(s))$ is non-decreasing, i.e.:

$$\Delta(x, \sigma_x(1)) \leq \dots \leq \Delta(x, \sigma_x(S)) . \quad (5.13)$$

If s is the closest activated slot to x , then all the slots closer to x must be deactivated. This observation leads us to rewrite \mathcal{L}_{cov} as follows:

$$\mathcal{L}_{\text{cov}}(\mathcal{M}, \mathbf{X}) = \frac{1}{|\mathbf{X}|} \sum_{x \in \mathbf{X}} \sum_{s=1}^S \Delta(x, s) \alpha_s \prod_{r < \sigma_x(s)} (1 - \alpha_{\sigma_x^{-1}(r)}) , \quad (5.14)$$

with σ^{-1} the inverse permutation of σ .

Regularization losses. We define several regularization losses implementing our priors on the model output and preventing degenerate local minima:

- To encourage the slot activation to be sparse, we use the following loss penalizing slot activation:

$$\mathcal{L}_{\text{act}}(\mathcal{M}) = \sum_{s=1}^S \mathbb{E}_{\mathbf{X}} [\alpha_s] . \quad (5.15)$$

- To avoid slots that are never used, we use the following loss, which we compute batch-wise:

$$\mathcal{L}_{\text{slot}}(\mathcal{M}) = - \sum_{s=1}^S \min \left(\frac{\mathbb{E}_{\mathbf{X}} [\alpha_s]}{\sum_{t=1}^S \mathbb{E}_{\mathbf{X}} [\alpha_t]}, \epsilon_S \right) , \quad (5.16)$$

with ϵ_S a hyperparameter that can be interpreted as the smallest acceptable relative use frequency for a slot.

- To avoid unused prototypes, we use the following loss:

$$\mathcal{L}_{\text{proto}}(\mathcal{M}) = - \sum_{k=1}^K \min \left(\frac{\mathbb{E}_{\mathbf{X}} [\sum_{s=1}^S \beta_s^k]}{\mathbb{E}_{\mathbf{X}} [\sum_{s=1}^S \alpha_s]}, \epsilon_K \right) , \quad (5.17)$$

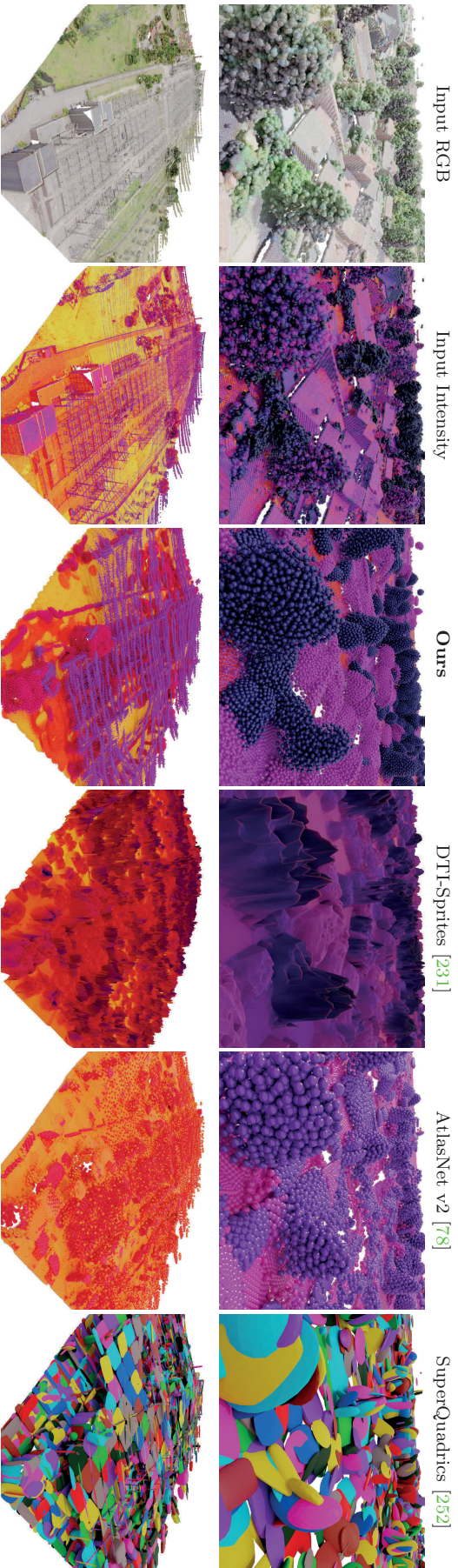


Figure 5.4: **Reconstruction Quality.** We show two partial scenes with their RGB and intensity values, as well as their reconstruction by our method and competing models. We use the prototypes’ intensity to color the points or pixels. As SuperQuadratics does not model the intensity, we use a random colour scheme.

with ϵ_K a hyperparameter that can be interpreted as the smallest acceptable relative use frequency for a prototype.

The full regularization loss is a weighted sum of the three losses described above:

$$\mathcal{L}_{\text{reg}} = \lambda_{\text{act}}\mathcal{L}_{\text{act}} + \lambda_{\text{slot}}\mathcal{L}_{\text{slot}} + \lambda_{\text{proto}}\mathcal{L}_{\text{proto}} , \quad (5.18)$$

where we use $\lambda_{\text{act}} = 10^{-4}$, $\lambda_{\text{slot}} = \lambda_{\text{proto}} = 0.1$ and $\epsilon_S = \epsilon_K = 0.1$ in all our experiments on the Earth Parser Dataset.

Final loss. We train our model to minimize the following loss:

$$\mathcal{L}(\mathbf{X}, \mathcal{M}) = \mathcal{L}_{\text{rec}}(\mathbf{X}, \mathcal{M}) + \mathcal{L}_{\text{reg}}(\mathcal{M}) . \quad (5.19)$$

5.3.3 Training and implementation details

Model configuration. We process the input point cloud \mathbf{X} using a similar architecture as in [252]. We first voxelize it into a $64 \times 64 \times 64$ grid and convert it to a vector using a sequence of 6 3D sparse convolutions [66] and 6 strided convolutions. The resulting representation is then transformed using one linear layer for each slot. These features are then decoded by simple 2-layer MLPs: one generates the distribution parameters α_s and β_s^k , and the other ones the parameters of the 3D transformations $\mathcal{T}_s(X)$. The transformations include an anisotropic scaling, a y -axis tilt of $\pm\pi/10$, a rotation around the z -axis, and a translation, in this specific order. We use $S = 64$ slots and $K = 6$ prototypes as default parameters.

The intensity of the return signal of each point is available in the LiDAR scans. We associate each prototype with a single learnable intensity parameter and perform the Chamfer distance (Equation 5.7) in 4 dimensions: spatial coordinates normalized to $[0, 1]^3$ and intensity to $[0, 0.1]$.

Curriculum learning. The network predicts simultaneously the slot’s probability distributions and their deformations. This results in many concurrent degrees of freedom and can make the training process unstable. Therefore, following Monnier *et al.* [231] and Loiseau *et al.* [195], we implement a multi-stage curriculum learning strategy. We first initialize the prototypes as point

Table 5.1: **Earth Parser Dataset.** Our proposed dataset is composed of 7 diverse scenes.

| Name | Surface in km ² | # points ×10 ⁶ | annotation ratio in % | num. of classes |
|--------------|-------------------------------|------------------------------|--------------------------|--------------------|
| Crop fields | 1.1 | 19.7 | 77.4 | 2 |
| Forest | 1.1 | 46.7 | 97.8 | 2 |
| Greenhouses | 0.1 | 1.3 | 95.6 | 3 |
| Marina | 0.1 | 0.5 | 92.7 | 2 |
| Power plant | 0.2 | 8.6 | 78.4 | 4 |
| Urban | 1.1 | 15.7 | 95.9 | 3 |
| Windturbines | 4.2 | 5.6 | — | — |
| Total | 7.7 | 98.3 | 89.6 | — |

clouds uniformly sampled from a random cuboid and gradually unfreeze the model parameters in the following order: (i) translation, rotation, tilt, slot activation, and choice of prototype; (ii) intensities of the prototypes, when available; (iii) scales of the prototypes; (iv) shapes of the prototypes (positions of their 3D points); (v) anisotropic scalings of the prototypes. As shown in the result section (Section 5.4), each step of this curriculum scheme improves the performances.

Prototypes selection. We automatically select the number of prototypes for a complete scene using a simple greedy algorithm. We measure the increase of reconstruction loss when preventing the model from selecting each prototype individually. We remove the prototype with the lowest increase if it is lower than 5%, and iterate.

Detailed architecture. Our model takes a point cloud \mathbf{X} and computes a voxelization in a grid of size $64 \times 64 \times 64$. As shown in Figure 5.5, our model is composed of (i) a point encoder $\mathcal{E}_{\text{point}}$, (ii) a scene encoder $\mathcal{E}_{\text{scene}}$, (iii) S slot feature extractors \mathcal{D}_s and (iv) five shared slot parameters generators: $\mathcal{D}_{\text{proba}}$, $\mathcal{D}_{\text{scale}}$, $\mathcal{D}_{\text{rot-y}}$, $\mathcal{D}_{\text{rot-z}}$, $\mathcal{D}_{\text{translate}}$. We provide details on these networks below.

- **Point encoder.** Each input point of \mathbf{X} is associated with a 10-dimensional descriptor: (1-3) normalized position in the tile in $[-1, 1]^3$, (4-6) *rgb* color, (7) normalized LiDAR reflectance, and (8-10) its offset relative to the center of

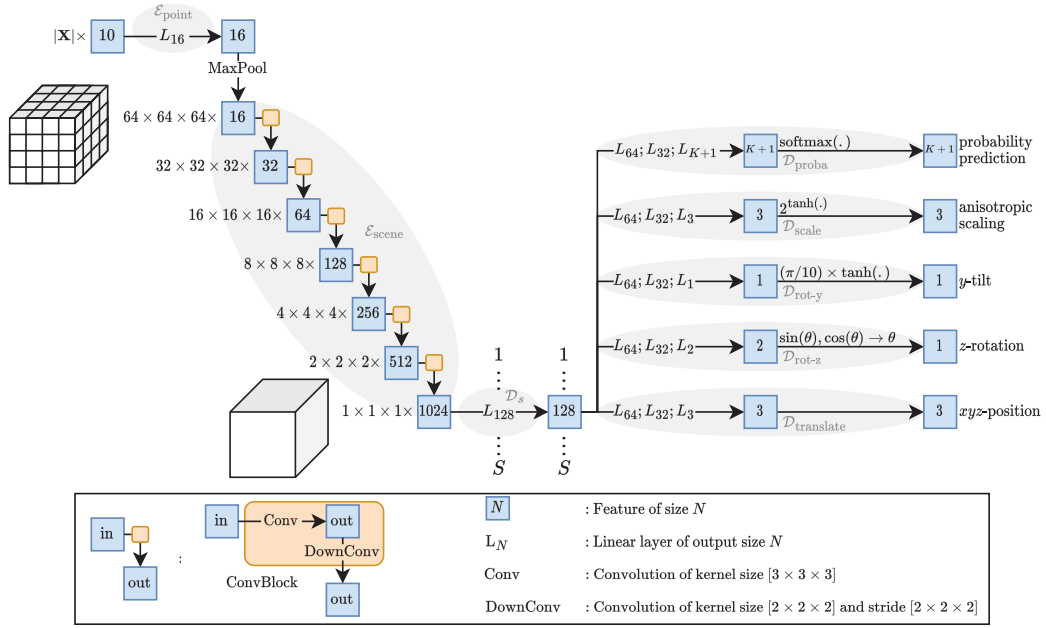


Figure 5.5: **Learnable Earth Parser Detailed Architecture.** Details of the architecture showing all layers in $\mathcal{E}_{\text{point}}$, $\mathcal{E}_{\text{scene}}$, \mathcal{D}_s , $\mathcal{D}_{\text{proba}}$, $\mathcal{D}_{\text{scale}}$, $\mathcal{D}_{\text{rot-y}}$, $\mathcal{D}_{\text{rot-z}}$ and $\mathcal{D}_{\text{translate}}$. We use LayerNorm [17] and LeakyRelu after all hidden layers.

its assigned voxel. The point encoder $\mathcal{E}_{\text{point}}$ is a linear layer that maps these descriptors to a 16-dimensional point feature.

- **Scene encoder.** We compute voxel features by max-pooling the features of the points associated to each voxel. The scene encoder $\mathcal{E}_{\text{scene}}$ then maps these voxel features to a single scene feature, a vector of size 1024, by using a sequence of 6 3D sparse convolutions [66] with kernel size $[3, 3, 3]$ and 6 strided convolutions with kernel size $[2, 2, 2]$ and stride $[2, 2, 2]$.

- **Slot feature extractor.** Each slot s takes as input the scene feature produced by $\mathcal{E}_{\text{scene}}$ and maps it to a slot feature of size 128 with a dedicated linear layer \mathcal{D}_s .

- **Slot parameters generators.** Five 3-layers MLPs are shared by all slots to map their slot features to the associated parameters of the reconstruction model.

- $\mathcal{D}_{\text{proba}}$ outputs the slot activation and prototype choice probability α_s et β_s^k .
- $\mathcal{D}_{\text{scale}}$ outputs three scales in $[-1/2, 2]$, corresponding to scaling the prototypes in each canonical directions.
- $\mathcal{D}_{\text{rot-y}}$ outputs a rotation in $[-\pi/10, \pi/10]$ to be applied around the y axis.

- $\mathcal{D}_{\text{rot-z}}$ outputs a 2D point on the unit circle which is then mapped to a rotation in $[-\pi, \pi]$ to be applied around the z axis.
- $\mathcal{D}_{\text{translate}}$ outputs a 3D translation vector in \mathbb{R}^3 .

These parameters are used to determine the activation of the slot, choose a prototype, then apply a sequence of transformations in the following order: scaling, y -rotation, z -rotation, and translation.

Implementation details. Our model is trained separately for each scene by randomly sampling square patches. During training, the patches are subsampled to a maximum 10 000 points. We use the efficient CUDA implementation of the Chamfer distance by PyTorch3D [263] which significantly speeds up training. We use the ADAM optimizer [155] with a learning rate of 10^{-4} and default parameters, except for the prototypes’ intensities, scales and points’ positions which we learn without weight decay. We define an “epoch” as 512 batches of 64 patches, and each stage of the curriculum is trained until convergence. Due to the arbitrary square shape of our samples \mathbf{X} , some objects can appear only partly in a patch. We don’t want the network to learn prototypes specifically to fit such object parts, as it is an artifact of our sampling procedure. Instead, we propose to ignore the points of the reconstruction \mathbf{Y}_s^k that falls beyond the normalized $[-1, 1]$ extent of the patches. This allows the network to predict full objects without being penalized in terms of accuracy. To do so, we modify Equation 5.9 as follows:

$$\mathcal{L}_{\text{acc}}(\mathcal{M}, \mathbf{X}) = \frac{1}{S} \sum_{s=1}^S \sum_{k=1}^K \beta_s^k d(\tilde{\mathbf{Y}}_s^k, \mathbf{X}) , \quad (5.20)$$

where $\tilde{\mathbf{Y}}_s^k$ is the subset of points of \mathbf{Y}_s^k that falls within the horizontal extent of their patch $[-1, 1]^2 \times \mathbb{R}$. To prevent the slots from predicting shapes outside of the patch extent, we regularize our model by the square Euclidean distance between the output of $\mathcal{D}_{\text{translate}}$ and the set $[-1, 1]^2 \times \mathbb{R}$ for each slot.

5.4 Results

In this section, we assess quantitatively and qualitatively the ability of our method to parse complex 3D aerial data. In Section 5.4.1, we first give an

overview of our proposed dataset of aerial LiDAR scans. In Section 5.4.2, we then discuss our evaluation metrics and baselines. Finally, we present quantitative (Section 5.4.3) and qualitative (Section 5.4.4) analysis of our results.

5.4.1 Earth Parser Dataset

We introduce a new dataset to train and evaluate parsing methods on large, uncurated aerial LiDAR scans. We use data from the French Mapping Agency associated to the LiDAR-HD [142] project. Each scan is composed of several airborne LiDAR acquisitions taken at different angles and fused, leading to a minimum resolution of 20 points/m². The points are associated with their laser reflectance (intensity), and colored based on asynchronous aerial photography. The majority of 3D points are annotated with a coarse semantic label, such as ground, building, or vegetation.

We selected 7 scenes, covering over 7.7km² and a total of 98 million 3D points, with diverse content and complexity, such as dense habitations, forests, or complex industrial facilities. The characteristics of the scenes are detailed in Table 5.1 and each is visualized in Figure 5.3.

Classes names. As show in Table 5.2, each scene of the Earth Parser Dataset is annotated with different classes among “ground”, “vegetation”, “building”, “boats”, “bridge”, “electric lines”, and “windturbine”.

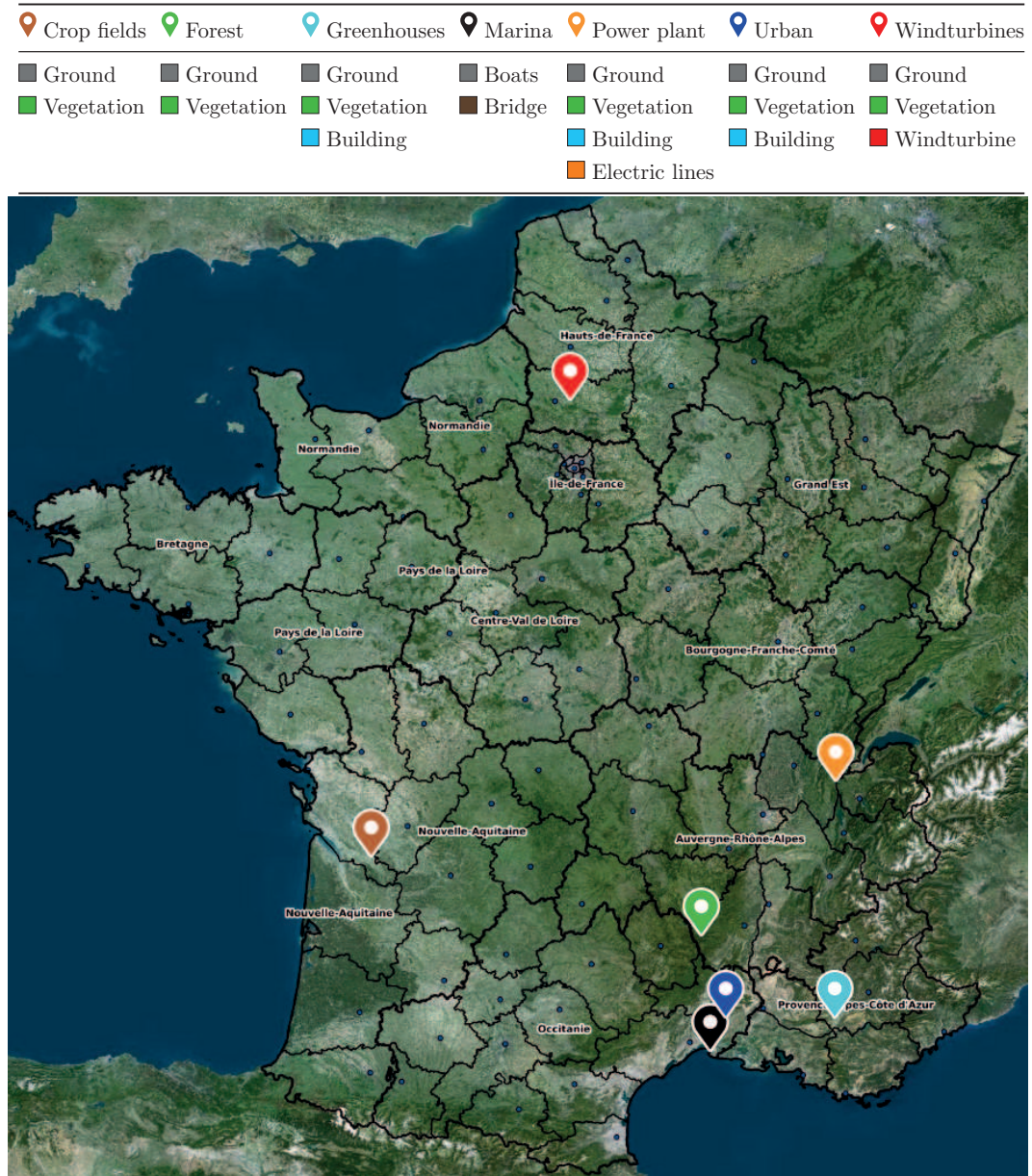
Localization. We report the localization of the scenes of Earth Parser Dataset in Table 5.2. Our dataset has been acquired in various environments distributed on the French territory.

5.4.2 Evaluation Metrics and Baselines

We quantitatively evaluated our performances for reconstruction and semantic segmentation and compared with several unsupervised scene decomposition approaches.

Evaluation metrics. The goal of our model is to summarize a point cloud using few prototypes and the quality of the reconstruction is of course critical.

Table 5.2: **Earth Parser Dataset Classes and Localisation.** We show the class names and color codes for the seven scenes of our dataset. Unlabeled points are represented in black ■. The Earth Parser Dataset was acquired at different locations in France, spanning a wide variety of environments.



We measure it using the symmetric Chamfer distance (“Cham.” in the Tables) between the input and the output of our model, only taking the points’ positions into account (not the intensity).

If the points of our prototype point clouds are associated to a semantic class, we can propagate the labels from the reconstruction to the input cloud and perform semantic segmentation. We then evaluate the quality of the prediction with the class-averaged Intersection-over-Union (mIoU) metric. In a practical scenario, an operator can manually annotate the points of the 3D prototypes, which can then be used to label the entirety of \mathbf{X} with minimal effort. To perform automatic evaluation, we assign instead to each prototype’s point the most frequent class in the reconstructions of the closest point in the input (see Figure 5.8).

Baselines. We adapted several unsupervised approaches for scene reconstruction and/or semantic segmentation tasks to provide baselines for our approach:

- **k-means.** We cluster the points of the input with the k-means algorithm [205] using as many clusters as we use prototypes in our method. We obtained the best results by using a combination of the point’s intensity and elevation as features for clustering. We then assign to each centroid its most frequent class, leading to a semantic segmentation. This method does not reconstruct the input, but gives us a simple baseline score for semantic segmentation.
- **SuperQuadrics revisited.** We use the method of Paschalidou *et al.* [252] to learn to approximate the scenes with an adaptive number of superquadrics [20]. It provides a baseline for reconstruction and a qualitative comparison for instance segmentation.
- **DTI-Sprites.** We use the point cloud to construct a digital elevation model, *i.e.* a 2.5D image where each pixel has an elevation and intensity value. We then adapt the unsupervised image decomposition approach of Monnier *et al.* [231] to break down this image into a set of 2.5D *sprites*. We evaluate the quality of the 2.5D reconstruction by sampling 25 point per pixel and using its elevation and the semantic segmentation in a way similar to our method.
- **AtlasNet v2.** This extension [78] of AtlasNet [115] uses a fixed number of learnable prototype point clouds to reconstruct the input. It can be evaluated for both reconstruction and semantic segmentation in a way similar to ours.

We extend it to handle intensity in a manner akin to our approach, which improves its segmentation results.

Similar to our method, we train all baselines except k-means by sampling square patches in each scene. Figure 5.4 shows the output of the reconstruction methods.

5.4.3 Quantitative Results

Earth Parser Dataset. Quantitative reconstruction and semantic segmentation results are provided in Table 5.3. Despite being highly constrained, our model yields the best reconstruction in 6 out of 7 scenes. Moreover, our model significantly outperforms all the other evaluated methods in terms of semantic segmentation across all scenes.

Despite its simplicity, the k-means baseline provides semantic segmentation performance on par with our best reconstruction baselines. DTI-Sprites [231] leads to lower reconstruction quality, which is expected as it models a 3D point cloud in 2.5D and this hurts its semantic segmentation performances. Atlas-Net v2 [78] provides good reconstructions but segmentation fails for scenes such as crop fields or urban areas due to its inability to adapt the prototypes used depending on the considered patches. On the contrary, SuperQuadrics [252] can adjust the number of prototypes used and their shapes to some degree, but it is not adapted for semantic segmentation since it uses a single type of prototype. In contrast, our approach learns a small set of prototypical shapes and our probabilistic slot selection approach can handle inputs with a varying number of objects.

Ablation study. The results of our ablation study are reported in Table 5.4. First, we observe that the prototype selection post-processing step has limited impact on the quality of the prediction and reconstructions, but it allows us to significantly decrease the number of prototypes and adapt it to the scene. Second, we evaluate the impact of reducing the expressivity of our model. We successively remove: (i) the anisotropic scaling of \mathcal{T} , and the possibility of learning the prototype’s (ii) points position, (iii) scale, and (iv) intensity. As expected, each deformation we remove decreases the quality of the 3D reconstruction and segmentation. Learning such model with a curriculum scheme

Table 5.3: **Results on the Earth Parser Dataset.** We report the quality of the reconstruction (Cham.) and semantic segmentation (mIoU) on each of the scenes of our Earth Parser Dataset. While our method does not always provide the most faithful reconstructions, it leads to the most accurate point classification.

| | Rec. | Crop fields | | Forest | | Greenhouses | | Marina | | Power plant | | Urban | | Wind turbines | |
|-----------------------|----------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|---------------|------|
| | | Cham. | mIoU | Cham. | mIoU | Cham. | mIoU | Cham. | mIoU | Cham. | mIoU | Cham. | mIoU | Cham. | mIoU |
| k-means (i,z) [205] | ✗ | — | 93.8 | — | 71.5 | — | 39.3 | — | 41.4 | — | 42.8 | — | 56.5 | — | — |
| SuperQuadratics [252] | 3D ✗ | 0.86 | — | 1.04 | — | 0.60 | — | 0.93 | — | 0.58 | — | 0.40 | — | 13.5 | — |
| DTI-Sprites [231] | 2.5D+i ✓ | 6.10 | 83.2 | 14.59 | 40.2 | 5.36 | 42.0 | 6.16 | 41.4 | 5.36 | 29.0 | 2.99 | 47.3 | 36.19 | — |
| AtlasNet v2 [78] | 3D+i ✓ | 1.07 | 43.1 | 1.58 | 71.4 | 0.56 | 49.1 | 0.73 | 42.1 | 0.45 | 41.6 | 0.63 | 48.8 | 8.80 | — |
| Ours | 3D+i ✓ | 0.72 | 96.9 | 0.88 | 83.7 | 0.40 | 91.3 | 0.82 | 78.7 | 0.44 | 52.2 | 0.29 | 83.2 | 6.65 | — |






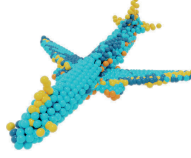


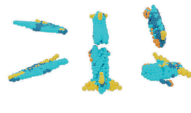

Table 5.4: **Ablation Study.** We evaluate the effect of our prototype selection post-processing, our model’s degrees of freedom, and our different regularization losses. See text for details.

| | | Urban | | Marina | |
|------------------------|------------------------------------|-------------|-------------|-------------|-------------|
| | | Cham. | mIoU | Cham. | mIoU |
| Learnable Earth Parser | | 0.29 | 83.2 | 0.82 | 78.7 |
| ↳ w/o post-processing | | 0.28 | 83.7 | 0.96 | 78.3 |
| expressivity | ↳ w/o aniso-scale | 0.33 | 82.4 | 1.04 | 67.2 |
| | ↳ w/o prototypes | 0.36 | 68.3 | 1.07 | 42.8 |
| | ↳ w/o scales | 0.55 | 58.9 | 1.33 | 40.8 |
| | ↳ w/o intensities | 0.55 | 58.7 | 1.09 | 40.8 |
| | ↳ w/o \mathcal{L}_{act} | 0.17 | 54.1 | 0.80 | 56.9 |
| losses | ↳ w/o $\mathcal{L}_{\text{slot}}$ | 0.25 | 77.8 | 0.81 | 43.7 |
| | ↳ w/o $\mathcal{L}_{\text{proto}}$ | 0.28 | 57.2 | 0.97 | 40.7 |

enables us to demonstrate strong reconstruction and semantic performances, without falling into bad minima. Third, we study the impact of removing the regularization losses introduced in Section 5.3.2. As expected, removing any of these losses clearly decreases the semantic segmentation performance, but improves the reconstruction quality for the losses related to the slots’ activation.

ShapeNet. We also evaluate the performance of our model on the 2690 planes from ShapeNet-Part [280], whose points are annotated as wing, engine, tail, or body. Since those point clouds are aligned, we experimented with and without random rotations around the z -axis during training and evaluation. We report the performance of our approach, AtlasNet v2 [78] and SuperQuadrics revisited [252], and illustrations of a reconstructed shape from the dataset in Table 5.5. While AtlasNet v2 provides a better reconstruction and segmentation in the aligned dataset, our model can better handle rotations. SuperQuadrics reconstructions make sense qualitatively, but are much worse in terms of accuracy, and do not enable semantic segmentation.

Table 5.5: **Synthetic Shapes**. We train our method on all planes from ShapeNet-Part [280], with and without random rotations. We show the reconstruction of an input plane and the prototypes learned on the dataset without rotations.

| | AtlasNet v2 [78] | SuperQuadrics [252] | Ours |
|-----------------------|---|--|---|
| aligned | Recons.  |  |  |
| | Protos.  | — |  |
| | Cham. 0.89 | 3.07 | 0.95 |
| | mIoU 72.8 | — | 67.8 |
| with random rotations | Recons.  |  |  |
| | Protos.  | — |  |
| | Cham. 1.46 | 2.91 | 1.34 |
| | mIoU 34.5 | — | 68.6 |

5.4.4 Qualitative Results

Reconstruction. Figure 5.4 shows the reconstructions of our method and baselines for two scenes. DTI-Sprites only provides a 2.5D reconstitution and struggles to capture objects like wires or poles accurately. AtlasNet v2 suffers from its inability to adapt its prototype usage to the input. SuperQuadrics' primitives are too coarse for reconstructing the diversity of objects present in a complex scan and are of a single type. In contrast, our Learnable Earth Parser models the intensity and can adapt to diverse shapes.

Instance segmentation. We can perform instance segmentation simply by considering each slot as a different instance. This is particularly interesting for parsing natural woodlands, a key endeavor for forest management [234] and

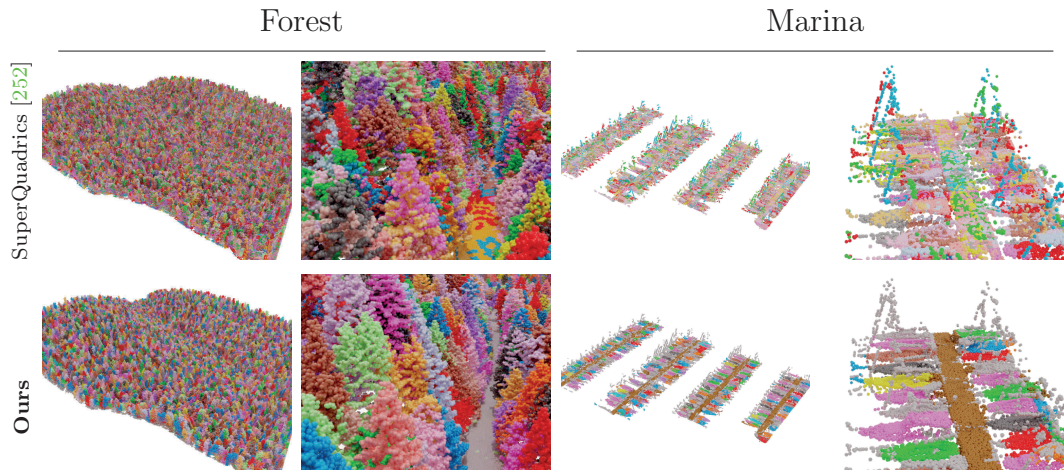


Figure 5.6: **Instance Segmentation.** We can identify the reconstruction of each slot as a separate instance, allowing us to perform instance segmentation of complex data such as dense natural forests or boats in a marina. For this visualization, we considered the points associated to “trees” or “boat hull” prototypes and color each of their instance randomly. We represent the instances predicted with our algorithm and by SuperQuadric [252]. We see that SuperQuadrics’ reconstruction struggles modeling complex objects with only one instance. Moreover, our method make it easier to differentiate between different object types such as “trees” or “boat hull”, while all superquadric are generated in the same way.

biomass estimation [88]. While this task has a long history of handcrafted approaches [327], current deep learning approaches are mostly limited to artificial or low-density forests [301]. As shown qualitatively in Figure 5.6, our Learnable Earth Parser can learn without any supervision to separate individual trees in dense forests or boats in a marina from aerial LiDAR scans. We show a comparison of the instance segmentation produced by SuperQuadrics [252] and our Learnable Earth Parser. Since SuperQuadrics [252] uses a restricted family of 3D shapes to reconstruct an input scene, it has worst qualitative performances for instance segmentation when compared to our model, which learns scene-specific prototypes and can provide semantic information.

Semantic segmentation. Associating to each point of the input the semantic class of the closest reconstructed point, we can perform semantic segmentation. As shown in Figure 5.7, our model can predict with minimal effort the semantic segmentation of a urban district or a power plant.

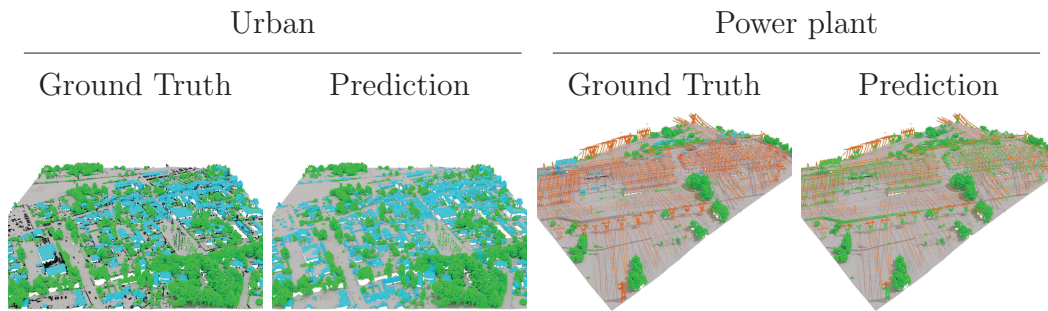


Figure 5.7: **Semantic Segmentation.** Our Learnable Earth Parser can perform semantic segmentation of large real-world scene based on prototypes annotation. Black points in the ground truth are non-annotated points.

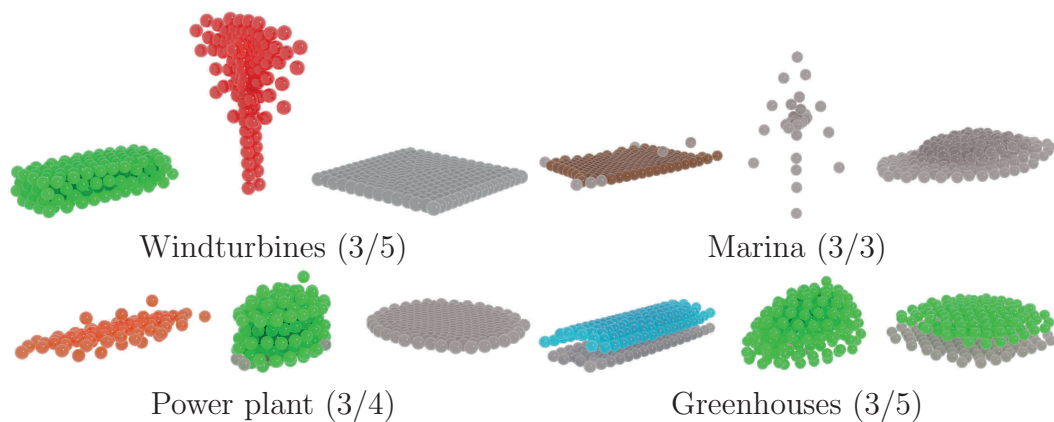


Figure 5.8: **Learned Prototypes.** Selected learned prototypes on different scenes. We show three prototypes among those selected by our post-processing selection.

Interpretable prototypes. In Figure 5.8, we show prototypes learned on our Earth Parser Dataset with colors showing the associated semantic label for each point. These prototypes give at a glance insights on the content of these real-world scenes. Our model is able to learn a wide variety of shapes, such as boats’ masts, wind turbines or greenhouses. Our selection strategy is also able to adapt the number of prototypes to the complexity of the overall scene.

Earth Parser Dataset results. We show in Figure 5.9 the ground truth semantic segmentation, our predicted semantic segmentation, our reconstruction and our learned prototypes. They showcase the quality, interpretability, and diversity of use cases of our model on this dataset of aerial LiDAR scans.

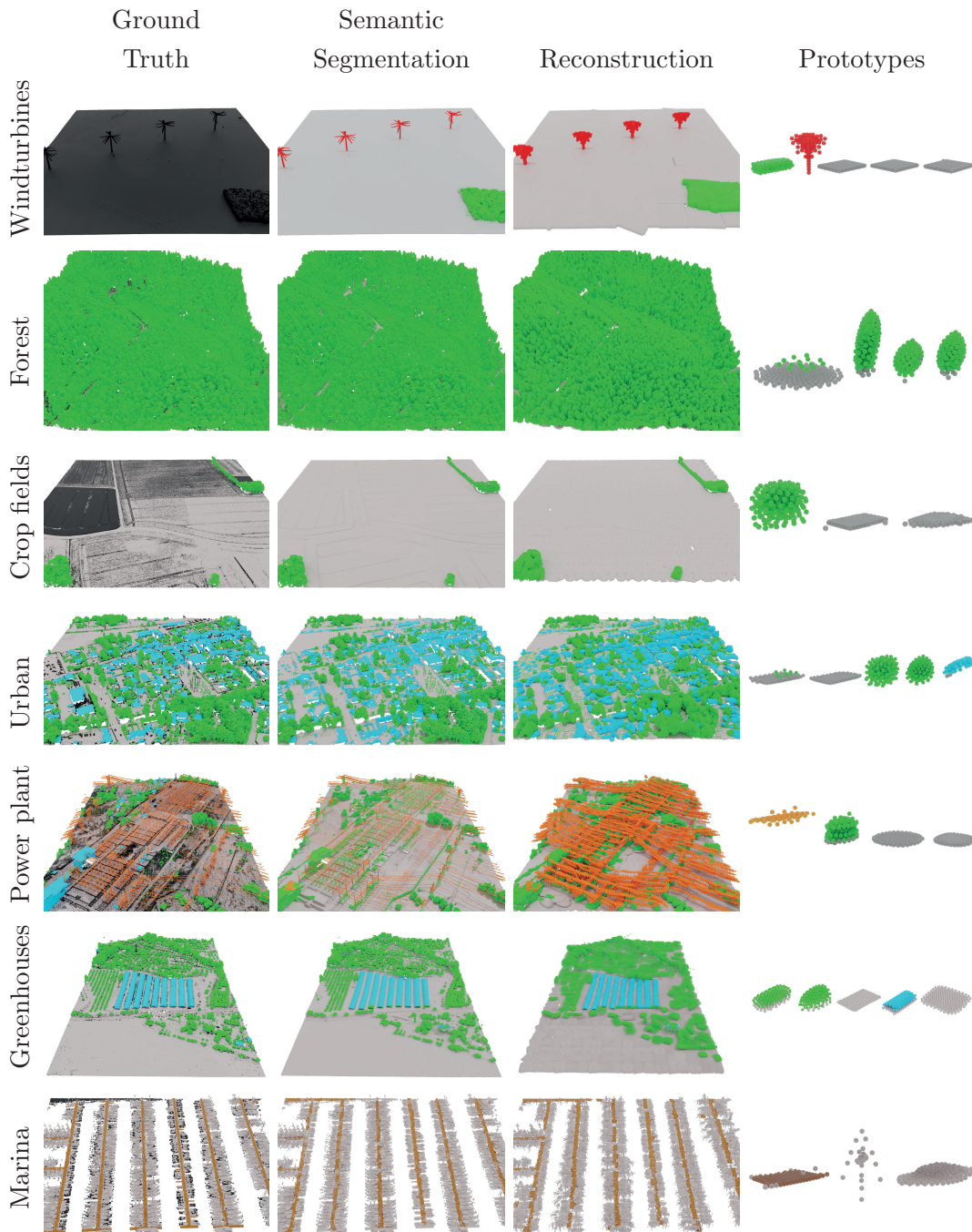


Figure 5.9: **Qualitative Results.** For all scenes of the Earth Parser Dataset, we show the ground truth labels, the semantic segmentation, reconstruction, and prototypes learned by our Learnable Earth Parser.

5.5 Conclusion

We introduced a novel unsupervised method for parsing complex real-world aerial scans into simple parts using a small set of learned prototypical shapes. We demonstrate the quality and interpretability of our results on a novel dataset of aerial LiDAR scans. To the best of our knowledge, we are the first to demonstrate the possibility of performing deep unsupervised 3D shape analysis on such a challenging real-world dataset. We believe that our results open new perspectives for computer-assisted environment monitoring and economic intelligence.

Acknowledgements. This work was supported in part by ANR project READY3D ANR-19-CE23-0007 and was granted access to the HPC resources of IDRIS under the allocation 2022-AD011012096R2 made by GENCI. The scenes of Earth Parser Dataset were acquired and annotated by the LiDAR-HD [142] project. We thank Zenodo for hosting the dataset. We thank Zeynep Sonat Baltaci, Nicolas Dufour, Antoine Guédon, Helen Mair Rawsthorne, Tom Monnier, Damien Robert, Mathis Petrovich and Yannis Siglidis for inspiring discussions and valuable feedback.

Chapter 6

Conclusion

In this chapter, we summarize our contributions and outline some future research directions.

6.1 Contributions

The key takeaways from this thesis are twofold: (i) the acquisition geometry of 3D sensors can be leveraged to accelerate data processing and improve efficiency, and (ii) the use of models operating in input space leads to powerful and interpretable unsupervised methods. By developing methods which take into account the sensor’s specificity and that do not need to be trained on gigantic amounts of annotated data, we worked toward more efficient and interpretable models for real-world 3D data analysis. In this thesis, we make three main contributions:

Online Segmentation of LiDAR Sequences. In Chapter 3, we propose a novel approach that takes advantage of the acquisition geometry of LiDAR sensors. Our real-time semantic segmentation method is based on an efficient spatio-temporal transformer and outperforms existing methods in terms of latency and compacity while preserving *state-of-the-art* segmentation performances. We also introduce an open-source annotated dataset specifically designed to assess and design real-time segmentation methods. We hope that this work will encourage future work to take advantage of the specificities of LiDAR data to reach higher efficiency.

Exploring Shape Collections. In Chapter 4, we present a new take on linear shape models with deep learning. We present an interpretable unsupervised representation learning framework for large unstructured 3D shape collections. Our alignment-aware model allows for concise, expressive, and interpretable overviews of unaligned point-cloud collections.

Discovering Prototypes in Aerial Scans. In Chapter 5, we extend the 3D unsupervised learning framework of Chapter 4 to parse complex aerial scans into simple parts using a small set of learned shapes. We also present an open-source dataset acquired by aerial LiDAR in various environments, such as forests, urban areas, and marinas. We demonstrate the possibility of performing deep unsupervised 3D shape analysis on challenging real-world data. We believe that our results open new perspectives for computer-assisted environment monitoring and economic intelligence, such as improved biomass or urban expansion monitoring.

6.2 Perspectives

In this section, we discuss exciting research directions to extend the results of this thesis. First, we propose approaches to improve the efficiency of deep learning methods for real-world 3D data. Second, we consider possible extensions of the deep transformation-invariant clustering framework to other data and applications.

6.2.1 Improving the Efficiency of 3D Data Representations

As the amount of computing resources needed to train and deploy *state-of-the-art* deep learning models continues to grow, we believe that working toward more computationally efficient models is crucial. Here, we discuss ideas to improve the efficiency of 3D deep learning methods through better data representations.

In Chapter 3, we showed that by considering the structure of the 3D sensor, we could leverage transformer models to obtain better performances and computational efficiency. Additionally, operating on coarser 3D partitions than

commonly used regular grids, while maintaining high semantic consistency, could also benefit computational efficiency. Indeed, such coarser partitions can be used to compress the data and exploit spatial redundancy, making subsequent processes less demanding. To this aim, Landrieu *et al.* [167] propose to over-segment 3D point clouds into superpoints with significant improvements compared to *state-of-the-art*. Using a transformer on top of a precomputed geometric partition has also shown great promise when applied to large-scale real-world 3D data [266]. Thus, learning end-to-end a way to perform coarse partitioning of the 3D space alongside a dedicated transformer could enable more efficient and accurate models. Such models have already shown promising results on instance segmentation [140] but are not end-to-end. Finally, adapting attention mechanisms to learned 3D partitioning could enhance both methods’ performances and efficiency.

6.2.2 Improving Transformation-Invariant Clustering

Some classical approaches have explored representing data collections with a few deformable prototypes [91, 92, 90, 93]. More recently, the deep transformation-invariant clustering framework introduced by Monnier *et al.* [230] enabled to learn deformations alongside the prototypes, making it suitable for large image collections. In this thesis, we extended this framework to 3D shapes [195] and real-world 3D scenes [197]. This framework could be extended further to parse scenes with a multi-scale structure, or to other data type entirely.

New modalities. In this thesis, we considered 3D objects and scenes, but the deep transformation-invariant framework can also be applied to other modalities such as sounds or images for example. We now propose a generic formulation of the deep transformation-invariant framework. For an arbitrary modality space Ω , the method requires:

- (i) a set X of N samples $x \in \Omega$,
- (ii) a differentiable distance $d : \Omega \times \Omega \mapsto \mathbf{R}$ between elements of Ω , and
- (iii) a family of parametric differentiable deformations $\mathcal{T}_\theta : \Omega \mapsto \Omega$ operating on the elements of Ω , with θ the parameters of the transformations.

Deep transformation-invariant clustering approaches rely on a reconstruction loss to perform clustering. The generic clustering objective function \mathcal{L} can be written as:

$$\mathcal{L}(X, P, \Theta) = \sum_{x \in X} \min_{k=1}^K d(x, \mathcal{T}_{\Theta(x)}(P_k)) , \quad (6.1)$$

with $P = \{P_1, \dots, P_K\} \in \Omega^K$ a set of K learnable prototypes, and Θ a neural network which takes as input an element of Ω and which outputs the parameters θ of the transformation \mathcal{T}_θ to apply to prototypes to reconstruct this given input. Different variations of this loss can be designed when using a variable number of prototypes to reconstruct an input [231, 197], when generating the prototypes [293], or when using supervision [329].

While the work of Monnier *et al.* [230] has been designed for images, we extended it to three-dimensional point clouds at object [195] and scene level [197]. Other work developed the framework for other modalities such as character analysis and recognition in text lines [293] and clustering and classification of satellite image time series [329]. During the thesis, we also had the opportunity to adapt this framework to audio data, leading to the following publications:

- Romain Loiseau, Baptiste Bouvier, Yann Teytaut, Elliot Vincent, Mathieu Aubry, Loic Landrieu, “A Model You Can Hear: Audio Identification With Playable Prototypes”, ISMIR, 2022.
- Romain Loiseau, Baptiste Bouvier, Yann Teytaut, Elliot Vincent, Mathieu Aubry, Loic Landrieu, “A Model You Can Hear: Audio Classification With Playable Prototypes”, CVPR Sight and Sound Workshop, 2022.

In this work¹ [196], we were motivated by the fact that current sound-processing methods typically rely on abstract and high-dimensional representations that are difficult to interpret. Inspired by approaches developed for image and 3D data, we propose an audio identification model based on learnable prototypes in spectral space. We design data-specific audio transformation networks to model gain, pitch, and high and low-frequency filters, see Figure 6.1. The learned audio prototypes can be used to cluster and classify input audio samples from large collections of sounds. Our model can be trained with or without supervision and reaches *state-of-the-art* instrument and speaker identification

¹Code available at <https://romainloiseau.fr/a-model-you-can-hear>.

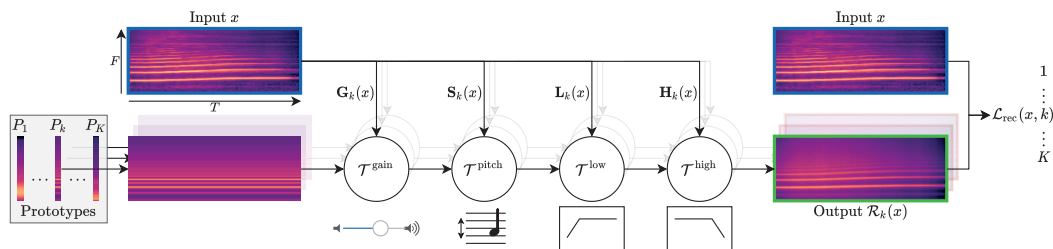


Figure 6.1: **A Model You Can Hear Overview.** Given an **input sound**, we predict for each prototype a *gain*, a *pitch* shift, as well as *low* and *high frequency filters* at each timestamp to generate the **output**. Prototypes and transformations are learned jointly using a reconstruction loss that can be supervised or not.

results on the SOL [19, 47] and Librispeech [246] datasets, respectively. Our model also remains easily interpretable, as our prototypes can be directly listened to and manipulated in amplitude and pitch, and capture the timber of individual instruments and voices.

The deep transformation-invariant clustering framework has been proven useful for analyzing unstructured data collections. It is also very generic, as shown by the variety of modalities it can handle. For example, medical data such as spine CT scans, and historical data where patterns can be identified, would be particularly suited for such methods.

Multi-scale clustering. The Deep Transformation Invariant clustering approach could also benefit multi-scale scene analysis. For example, our parsing method has shown promising results for real-world 3D scene decomposition. However, our 3D prototypes are all with the same 3D scale. Learning with such prototypes leads to scene decomposition at a fixed level of detail. Using prototypes with different scales could enable a more fine-grained scene representation and capture both large structures and small structures with more details. However, doing so in an end-to-end framework raises several challenges: (i) learned prototypes of the smaller scale may overfit the input scene, leading the model not to use larger ones, and (ii) the computational complexity would increase, as the number of potentially activated slots needed to reconstruct a scene would grow with the number of potentially discovered structures.

Bibliography

- [1] Solmaz Abbasi and Farshad Tajeripour. “Detection of brain tumor in 3D MRI images using local binary patterns and histogram orientation gradient”. In: *Neurocomputing* (2017).
- [2] Ahmad Fikri Abdullah, Alias Rahman, and Zoran Vojinovic. “LiDAR filtering algorithms for urban flood application: Review on current algorithms and filters test”. In: *Laserscanning* (2009).
- [3] Yair Adato et al. “Toward a theory of shape from specular flow”. In: *ICCV* (2007).
- [4] Antonio Alliegro, Davide Boscaini, and Tatiana Tommasi. “Joint supervised and self-supervised learning for 3D real world challenges”. In: *ICPR* (2021).
- [5] Pierre Alliez. “Recent advances in compression of 3D meshes”. In: *European Signal Processing Conference* (2005).
- [6] Dragomir Anguelov et al. “Scape: shape completion and animation of people”. In: *SIGGRAPH* (2005).
- [7] Lasse F Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. “Carbontracker: Tracking and predicting the carbon footprint of training deep learning models”. In: *arXiv* (2020).
- [8] Apple Inc. “Medium range optical systems for remote sensing receivers”. WO2018/039249. Dec. 14, 2018.
- [9] Iro Armeni et al. “3D semantic parsing of large-scale indoor spaces”. In: *CVPR* (2016).
- [10] Anurag Arnab et al. “ViViT: A Video Vision Transformer”. In: *ICCV* (2021).
- [11] David Arthur and Sergei Vassilvitskii. *k-means++: The advantages of careful seeding*. Tech. rep. Stanford, 2006.
- [12] Souhaib Attaiki and Maks Ovsjanikov. “Understanding and improving features learned in deep functional maps”. In: *CVPR* (2023).
- [13] Matan Atzmon, Haggai Maron, and Yaron Lipman. “Point convolutional neural networks by extension operators”. In: *ACM Transactions on Graphics* (2018).

- [14] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. “The wave kernel signature: A quantum mechanical approach to shape analysis”. In: *ICCV Workshop* (2011).
- [15] Mehmet Aygün, Zorah Lähner, and Daniel Cremers. “Unsupervised dense shape correspondence using heat kernels”. In: *3DV* (2020).
- [16] Mehmet Aygun et al. “4D Panoptic LiDAR Segmentation”. In: *CVPR* (2021).
- [17] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. “Layer normalization”. In: *arXiv* (2016).
- [18] Xiao Bai et al. “Explainable deep learning for efficient and robust pattern recognition: A survey of recent developments”. In: *Pattern Recognition* (2021).
- [19] Guillaume Ballet et al. “Studio online 3.0: An internet ”killer application” for remote access to IRCAM sounds and processing tools”. In: *Journées d’Informatique Musicale* (1999).
- [20] Alan H Barr. “Superquadrics and angle-preserving transformations”. In: *Computer Graphics and Applications* (1981).
- [21] Jordan Steven Bates et al. “Estimating Canopy Density Parameters Time-Series for Winter Wheat Using UAS Mounted LiDAR”. In: *Remote Sensing* (2021).
- [22] Herbert Bay et al. “Speeded-up robust features (SURF)”. In: *CVIU* (2008).
- [23] J. Behley et al. “SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences”. In: *ICCV* (2019).
- [24] Serge Belongie, Jitendra Malik, and Jan Puzicha. “Shape context: A new descriptor for shape matching and object recognition”. In: *NeurIPS* (2000).
- [25] Pál Benkő, Ralph R Martin, and Tamás Várady. “Algorithms for reverse engineering boundary representation models”. In: *Computer-Aided Design* (2001).
- [26] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. “The Lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks”. In: *CVPR* (2018).
- [27] Prarthana Bhattacharyya, Chengjie Huang, and Krzysztof Czarnecki. “SA-Det3D: Self-attention based context-aware 3D object detection”. In: *ICCV Workshop* (2021).
- [28] Thomas Binford. “Visual perception by computer”. In: *Proc. IEEE Conf. on Systems and Control* (1975).
- [29] Michael J Black et al. “BEDLAM: A Synthetic Dataset of Bodies Exhibiting Detailed Lifelike Animated Motion”. In: *CVPR* (2023).

- [30] Volker Blanz and Thomas Vetter. “A morphable model for the synthesis of 3D faces”. In: *Computer Graphics and Interactive Techniques* (1999).
- [31] R Blomley et al. “Shape distribution features for point cloud analysis—a geometric histogram approach on multiple scales”. In: *ISPRS* (2014).
- [32] Federica Bogo et al. “FAUST: Dataset and evaluation for 3D mesh registration”. In: *CVPR* (2014).
- [33] Federica Bogo et al. “Dynamic FAUST: Registering human bodies in motion”. In: *CVPR* (2017).
- [34] Marcello Bolognesi et al. “Accuracy of cultural heritage 3D models by RPAS and terrestrial photogrammetry”. In: *ISPRS* (2014).
- [35] Davide Boscaini et al. “Learning shape correspondence with anisotropic convolutional neural networks”. In: *NeurIPS* (2016).
- [36] Leon Bottou and Yoshua Bengio. “Convergence properties of the k-means algorithms”. In: *NeurIPS* (1995).
- [37] Alexandre Boulch. “ConvPoint: Continuous convolutions for point cloud processing”. In: *Computers & Graphics* (2020).
- [38] Alexandre Boulch et al. “SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks”. In: *Computers & Graphics* (2018).
- [39] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. “Recovering non-rigid 3D shape from image streams”. In: *CVPR* (2000).
- [40] Miguel C Brito et al. “Photovoltaic potential in a Lisbon suburb using LiDAR data”. In: *Solar Energy* (2012).
- [41] Michael M Bronstein and Iasonas Kokkinos. “Scale-invariant heat kernel signatures for non-rigid shape recognition”. In: *CVPR* (2010).
- [42] Michael M Bronstein et al. “Geometric deep learning: going beyond euclidean data”. In: *Signal Processing Magazine* (2017).
- [43] Holger Caesar et al. “nusenes: A multimodal dataset for autonomous driving”. In: *CVPR* (2020).
- [44] Nicolas Carion et al. “End-to-end object detection with transformers”. In: *ECCV* (2020).
- [45] Donna Carless et al. “Mapping landscape-scale peatland degradation using airborne lidar and multispectral data”. In: *Landscape Ecology* ().
- [46] LE Carvalho and Aldo von Wangenheim. “3D object recognition and classification: a systematic literature review”. In: *Pattern Analysis and Applications* (2019).
- [47] Carmine Emanuele Cella et al. “OrchideaSOL: a dataset of extended instrumental techniques for computer-aided orchestration”. In: *ICMC* (2020).

- [48] Angel X. Chang et al. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.
- [49] Angel Chang et al. “Matterport3d: Learning from rgb-d data in indoor environments”. In: *3DV* (2017).
- [50] Thomas Chaton et al. “Torch-Points3D: A Modular Multi-Task Framework for Reproducible Deep Learning on 3D Point Clouds”. In: *3DV* (2020).
- [51] Nesrine Chehata, Li Guo, and Clément Mallet. “Airborne lidar feature selection for urban classification using random forests”. In: *Laserscanning* (2009).
- [52] Chao Chen et al. “Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis”. In: *CVPR* (2019).
- [53] YH Chen and CY Liu. “Quadric surface extraction using genetic algorithms”. In: *Computer-Aided Design* (1999).
- [54] Yunlu Chen et al. “3D Equivariant Graph Implicit Functions”. In: *ECCV* (2022).
- [55] Zhaiyu Chen et al. “Reconstructing compact building models from point clouds using deep implicit fields”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* (2022).
- [56] Zhiqin Chen and Hao Zhang. “Learning implicit fields for generative shape modeling”. In: *CVPR* (2019).
- [57] Ran Cheng et al. “AF2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network”. In: *CVPR* (2021).
- [58] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. “Implicit functions in feature space for 3D shape reconstruction and completion”. In: *CVPR* (2020).
- [59] Christopher Choy, JunYoung Gwak, and Silvio Savarese. “4d spatio-temporal convnets: Minkowski convolutional neural networks”. In: *CVPR* (2019).
- [60] Bumseok Chun and J-M Guldmann. “Spatial statistical analysis and simulation of the urban heat island in high-density central cities”. In: *Landscape and Urban Planning* (2014).
- [61] Özgün Çiçek et al. “3D U-Net: learning dense volumetric segmentation from sparse annotation”. In: *Medical Image Computing and Computer-Assisted Intervention* (2016).
- [62] Davide Coccomini et al. “Combining EfficientNet and vision transformers for video deepfake detection”. In: *ICIAP* (2021).

- [63] Papers with code. *3D Semantic Segmentation on SemanticKITTI*. URL: <https://paperswithcode.com/sota/3d-semantic-segmentation-on-semantickitti> (visited on 05/05/2023).
- [64] Jasmine Collins et al. “Abo: Dataset and benchmarks for real-world 3d object understanding”. In: *CVPR* (2022).
- [65] European Commission. *Monitoring Agricultural ResourceS (MARS)*. URL: https://joint-research-centre.ec.europa.eu/monitoring-agricultural-resources-mars_en (visited on 04/17/2023).
- [66] Spconv Contributors. *Spconv: Spatially Sparse Convolution Library*. <https://github.com/traveller59/spconv>. 2022.
- [67] Tim F Cootes and Christopher J Taylor. “Statistical models of appearance for medical image analysis and computer vision”. In: *Medical Imaging : Image Processing* (2001).
- [68] Marius Cordts et al. “The cityscapes dataset for semantic urban scene understanding”. In: *CVPR* (2016).
- [69] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. “SalsaNext: Fast, Uncertainty-aware Semantic Segmentation of LiDAR Point Clouds for Autonomous Driving”. In: *Advances in Visual Computing, ISVC* (2020).
- [70] Yang Cui et al. “Automatic 3-D reconstruction of indoor environment with mobile laser scanning point clouds”. In: *Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2019).
- [71] Stéphane d’Ascoli et al. “Convit: Improving vision transformers with soft convolutional inductive biases”. In: *ICML* (2021).
- [72] Mohamed Dahmane and Jean Meunier. “Emotion recognition using dynamic grid-based HoG features”. In: *International Conference on Automatic Face & Gesture Recognition (FG)* (2011).
- [73] Angela Dai et al. “ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes”. In: *CVPR* (2017).
- [74] Yuchao Dai, Hongdong Li, and Mingyi He. “A simple prior-free method for non-rigid structure-from-motion factorization”. In: *IJCV* (2014).
- [75] Boris Delaunay et al. “Sur la sphere vide”. In: *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* (1934).
- [76] Jérôme Demantké et al. “Dimensionality based scale selection in 3D lidar point clouds”. In: *ISPRS* (2012).
- [77] Yu Deng, Jiaolong Yang, and Xin Tong. “Deformed implicit field: Modeling 3D shapes with learned dense correspondence”. In: *CVPR* (2021).
- [78] Theo Deprelle et al. “Learning elementary structures for 3D shape generation and matching”. In: *NeurIPS* (2019).
- [79] Jean-Emmanuel Deschaud et al. “Paris-CARLA-3D: A real and synthetic outdoor point cloud dataset for challenging tasks in 3D mapping”. In: *Remote Sensing* (2021).

- [80] Radosvet Desislavov, Fernando Martinez-Plumed, and José Hernández-Orallo. “Trends in AI inference energy consumption: Beyond the performance-vs-parameter laws of deep learning”. In: *Sustainable Computing: Informatics and Systems* (2023).
- [81] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. “Superscript: Self-supervised interest point detection and description”. In: *CVPR* (2018).
- [82] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *ICLR* (2021).
- [83] Alexey Dosovitskiy et al. “CARLA: An Open Urban Driving Simulator”. In: *Conference on Robot Learning* (2017).
- [84] Herbert Edelsbrunner. *Geometry and topology for mesh generation*. Cambridge University Press, 2001.
- [85] Hehe Fan, Yi Yang, and Mohan Kankanhalli. “Point 4D Transformer Networks for Spatio-Temporal Modeling in Point Cloud Videos”. In: *CVPR* (2021).
- [86] Tsung-Pao Fang and Les A Piegl. “Delaunay triangulation in three dimensions”. In: *Computer Graphics and Applications* (1995).
- [87] Clara Fernandez-Labrador et al. “Unsupervised Learning of Category-Specific Symmetric 3D Keypoints from Point Sets”. In: *ECCV* (2020).
- [88] António Ferraz et al. “Airborne lidar estimation of aboveground forest biomass in the absence of field inventory”. In: *Remote Sensing* (2016).
- [89] 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).
- [90] Andrew Fitzgibbon and Andrew Zisserman. “On affine invariant clustering and automatic cast listing in movies”. In: *ECCV* (2002).
- [91] Brendan J Frey and Nebojsa Jojic. “Estimating Mixture Models of Images and Inferring Spatial Transformations Using the EM Algorithm”. In: *CVPR* (1999).
- [92] Brendan J Frey and Nebojsa Jojic. “Fast, Large-Scale Transformation-Invariant Clustering”. In: *NeurIPS* (2002).
- [93] Brendan J Frey and Nebojsa Jojic. “Transformation-Invariant Clustering Using the EM Algorithm”. In: *Transactions on Pattern Analysis and Machine Intelligence* (2003).
- [94] Huan Fu et al. “3d-future: 3d furniture shape with texture”. In: *IJCV* (2021).
- [95] Matheus Gadelha et al. “Label-Efficient Learning on Point Clouds using Approximate Convex Decompositions”. In: *ECCV* (2020).
- [96] Ran Gal et al. “iWIRES: An analyze-and-edit approach to shape manipulation”. In: *SIGGRAPH* (2009).

- [97] Ge Gao et al. “Saliency-guided adaptive seeding for supervoxel segmentation”. In: *IROS* (2017).
- [98] Weixiao Gao et al. “SUM: A benchmark dataset of semantic urban meshes”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* (2021).
- [99] Alberto Garcia-Garcia et al. “Pointnet: A 3d convolutional neural network for real-time object class recognition”. In: *IJCNN* (2016).
- [100] Xuming Ge et al. “A multi-primitive-based hierarchical optimal approach for semantic labeling of ALS point clouds”. In: *Remote Sensing* (2019).
- [101] Andreas Geiger et al. “Vision meets Robotics: The KITTI Dataset”. In: *International Journal of Robotics Research* (2013).
- [102] Kyle Genova et al. “Learning shape templates with structured implicit functions”. In: *CVPR* (2019).
- [103] Kyle Genova et al. “Local deep implicit functions for 3D shape”. In: *CVPR* (2020).
- [104] Jakob Geyer et al. “A2D2: Audi autonomous driving dataset”. In: *arXiv* (2020).
- [105] Charlie Giattino et al. “Artificial Intelligence”. In: *Our World in Data* (2022). <https://ourworldindata.org/artificial-intelligence>.
- [106] Justin Gilmer et al. “Neural message passing for quantum chemistry”. In: *ICML* (2017).
- [107] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. “Mesh r-cnn”. In: *ICCV* (2019).
- [108] Patrick Gonzalez et al. “Forest carbon densities and uncertainties from Lidar, QuickBird, and field measurements in California”. In: *Remote Sensing of Environment* ().
- [109] Google Trends. *Research interests for “deep learning” and “what is ai?”* URL: <https://trends.google.fr/trends/explore?date=all&q=deep%5C%20learning,what%5C%20is%5C%20ai> (visited on 05/05/2023).
- [110] Tolga Görüm. “Landslide recognition and mapping in a mixed forest environment from airborne LiDAR data”. In: *Engineering Geology* (2019).
- [111] Paulo FU Gotardo, Olga Regina Pereira Bellon, and Luciano Silva. “Range image segmentation by surface extraction using an improved robust estimator”. In: *CVPR* (2003).
- [112] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. “3d semantic segmentation with submanifold sparse convolutional networks”. In: *CVPR* (2018).
- [113] Benjamin Graham and Laurens van der Maaten. “Submanifold Sparse Convolutional Networks”. In: *arXiv* (2017).

- [114] Sorin Grigorescu et al. “A survey of deep learning techniques for autonomous driving”. In: *Journal of Field Robotics* (2020).
- [115] Thibault Groueix et al. “A papier-mâché approach to learning 3D surface generation”. In: *CVPR* (2018).
- [116] Thibault Groueix et al. “Unsupervised cycle-consistent deformation for shape matching”. In: *Computer Graphics Forum* (2019).
- [117] Stéphane Guinard, Loic Landrieu, and Bruno Vallet. “Weakly supervised segmentation-aided classification of urban scenes from 3D LiDAR point clouds”. In: *ISPRS Workshop* (2017).
- [118] Stéphane Guinard et al. “Piecewise-planar approximation of large 3D data as graph-structured optimization”. In: *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* (2019).
- [119] Florent Guiotte et al. “Semantic segmentation of lidar points clouds: rasterization beyond digital elevation models”. In: *IEEE Geoscience and Remote Sensing Letters* (2020).
- [120] Bo Guo et al. “An improved method for power-line reconstruction from point cloud data”. In: *Remote sensing* (2016).
- [121] Jianyuan Guo et al. “CMT: Convolutional neural networks meet vision transformers”. In: *CVPR* (2022).
- [122] Meng-Hao Guo et al. “PCT: Point cloud transformer”. In: *Computational Visual Media* (2021).
- [123] Yulan Guo et al. “Performance evaluation of 3D local feature descriptors”. In: *ACCV* (2015).
- [124] Yulan Guo et al. “A comprehensive performance evaluation of 3D local feature descriptors”. In: *IJCV* (2016).
- [125] Yulan Guo et al. “Deep learning for 3D point clouds: A survey”. In: *Transactions on Pattern Analysis and Machine Intelligence* (2020).
- [126] Abhinav Gupta, Alexei Efros, and Martial Hebert. “Blocks world revisited: Image understanding using qualitative geometry and mechanics”. In: *ECCV* (2010).
- [127] Norbert Haala and Claus Brenner. “Extraction of buildings and trees in urban environments”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* (1999).
- [128] Timo Hackel et al. “Semantic3d. net: A new large-scale point cloud classification benchmark”. In: *ISPRS* (2017).
- [129] Oshri Halimi et al. “Unsupervised learning of dense shape correspondence”. In: *CVPR* (2019).
- [130] Rana Hanocka et al. “Meshcnn: a network with an edge”. In: *ACM Transactions on Graphics* (2019).
- [131] Kaveh Hassani and Mike Haley. “Unsupervised multi-task feature learning on point clouds”. In: *CVPR* (2020).

- [132] Pedro Hermosilla, Tobias Ritschel, and Timo Ropinski. “Total Denoising: Unsupervised learning of 3D point cloud cleaning”. In: *CVPR* (2019).
- [133] Vu Hoang Hiep et al. “Towards high-resolution large-scale multi-view stereo”. In: *CVPR* (2009).
- [134] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. “Transforming auto-encoders”. In: *International Conference on Artificial Neural Networks* (2011).
- [135] Fangzhou Hong et al. “LiDAR-Based Panoptic Segmentation via Dynamic Shifting Network”. In: *CVPR* (2021).
- [136] Radu Horaud et al. “An overview of depth cameras and range scanners based on time-of-flight technologies”. In: *Machine vision and applications* (2016).
- [137] Jin Huang et al. “City3D: Large-Scale Building Reconstruction from Airborne LiDAR Point Clouds”. In: *Remote Sensing* (2022).
- [138] Lila Huang et al. “OctSqueeze: Octree-Structured Entropy Model for LiDAR Compression”. In: *CVPR* (2020).
- [139] Xinyu Huang et al. “The Apolloscape dataset for autonomous driving”. In: *CVPR Workshop* (2018).
- [140] Le Hui et al. “Learning Superpoint Graph Cut for 3D Instance Segmentation”. In: *NeurIPS* (2022).
- [141] Braden Hurl, Krzysztof Czarnecki, and Steven Waslander. “Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception”. In: *Intelligent Vehicles symposium (IV)* (2019).
- [142] IGN. *LiDAR-HD: a 3D mapping of France’s soil and subsoil*. URL: <https://geoservices.ign.fr/lidarhd>.
- [143] Katsushi Ikeuchi. “Determining surface orientations of specular surfaces by using the photometric stereo method”. In: *Transactions on Pattern Analysis and Machine Intelligence* (1981).
- [144] Velodyne LiDAR Inc. *HDL-64E User’s Manual*. Velodyne LiDAR Inc. 345 Digital Drive, Morgan Hill, CA 95037, 2008.
- [145] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *ICML* (2015).
- [146] Catalin Ionescu et al. “Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments”. In: *Transactions on Pattern Analysis and Machine Intelligence* (2013).
- [147] Michel Jaboyedoff et al. “Use of LIDAR in landslide investigations: a review”. In: *Natural hazards* (2012).
- [148] Chiyu Jiang et al. “ShapeFlow: Learnable Deformations Among 3D Shapes”. In: *NeurIPS* (2020).

- [149] Peng Jiang et al. “Rellis-3D dataset: Data, benchmarks and analysis”. In: *ICRA* (2021).
- [150] Andrew E Johnson. “Spin-images: a representation for 3-D surface matching”. PhD thesis. Carnegie Mellon University, 1997.
- [151] Andrew E Johnson and Martial Hebert. “Using spin images for efficient object recognition in cluttered 3D scenes”. In: *Transactions on Pattern Analysis and Machine Intelligence* (1999).
- [152] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. “A survey of simple geometric primitives detection methods for captured 3D data”. In: *Computer Graphics Forum* (2019).
- [153] Ekaterina Kalinicheva et al. “Multi-Layer Modeling of Dense Vegetation From Aerial LiDAR Scans”. In: *CVPR Earth Vision Workshop* (2022).
- [154] A. Katharopoulos et al. “Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention”. In: *ICML* (2020).
- [155] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *ICLR* (2014).
- [156] Florian Kluger et al. “Cuboids revisited: Learning robust 3D shape fitting to single rgb images”. In: *CVPR* (2021).
- [157] Arno Knapitsch et al. “Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction”. In: *ACM Transactions on Graphics* (2017).
- [158] Sebastian Koch et al. “ABC: A big CAD model dataset for geometric deep learning”. In: *CVPR* (2019).
- [159] Iasonas Kokkinos et al. “Intrinsic shape context descriptors for deformable shapes”. In: *CVPR* (2012).
- [160] Sören König and Stefan Gumhold. “Consistent Propagation of Normal Orientations in Point Clouds.” In: *VMV* (2009), pp. 83–92.
- [161] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *NeurIPS* (2012).
- [162] Florent Lafarge and Clément Mallet. “Creating large-scale city models from 3D-point clouds: a robust approach with hybrid representation”. In: *IJCV* (2012).
- [163] Florent Lafarge et al. “Automatic 3D building reconstruction from DEMs: an application to PLEIADES simulations”. In: *ISPRS* (2006).
- [164] Xin Lai et al. “Stratified transformer for 3D point cloud segmentation”. In: *CVPR* (2022).
- [165] Ireneusz Laks et al. “Possibilities of using low quality digital elevation models of floodplains in hydraulic numerical models”. In: *Water* (2017).
- [166] Jean-François Lalonde et al. “Scale selection for classification of point-sampled 3D surfaces”. In: *International Conference on 3D Digital Imaging and Modeling* (2005).

- [167] Loic Landrieu and Mohamed Boussaha. “Point cloud oversegmentation with graph-structured deep metric learning”. In: *CVPR* (2019).
- [168] Loic Landrieu and Martin Simonovsky. “Large-scale point cloud semantic segmentation with superpoint graphs”. In: *CVPR* (2018).
- [169] Benjamin Langmann, Klaus Hartmann, and Otmar Loffeld. “Depth Camera Technology Comparison and Performance Evaluation.” In: *ICPRAM* (2012).
- [170] Pierre Lemaire et al. “Fully automatic 3D facial expression recognition using differential mean curvature maps and histograms of oriented gradients”. In: *International Conference and Workshops on Automatic Face and Gesture Recognition (FG)* (2013).
- [171] Aleš Leonardis, Alok Gupta, and Ruzena Bajcsy. “Segmentation of range images as the search for geometric parametric models”. In: *IJCV* (1995).
- [172] Ales Leonardis, Ales Jaklic, and Franc Solina. “Superquadrics for segmenting and modeling range data”. In: *Transactions on Pattern Analysis and Machine Intelligence* (1997).
- [173] Jiaxin Li, Ben M Chen, and Gim Hee Lee. “So-net: Self-organizing network for point cloud analysis”. In: *CVPR* (2018).
- [174] Jun Li et al. “Grass: Generative recursive autoencoders for shape structures”. In: *Transactions on Graphics* (2017).
- [175] Lingxiao Li et al. “Supervised fitting of geometric primitives to 3D point clouds”. In: *CVPR* (2019).
- [176] Li Li et al. “Point2Roof: End-to-end 3D building roof modeling from airborne LiDAR point clouds”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* (2022).
- [177] Minglei Li, Peter Wonka, and Liangliang Nan. “Manhattan-world urban reconstruction from point clouds”. In: *ECCV* (2016).
- [178] Nanxi Li et al. “A Progress Review on Solid-State LiDAR and Nanophotonics-Based LiDAR Sensors”. In: *Laser & Photonics Reviews* (2022).
- [179] Xianzhi Li et al. “A rotation-invariant framework for deep point cloud analysis”. In: *Transactions on Visualization and Computer Graphics* (2021).
- [180] Xinju Li and Igor Guskov. “3D object recognition from range images using pyramid matching”. In: *ICCV* (2007).
- [181] Xinke Li et al. “Primitive3D: 3D Object Dataset Synthesis from Randomly Assembled Primitives”. In: *CVPR* (2022).
- [182] Yangyan Li et al. “Globfit: Consistently fitting primitives by discovering global relations”. In: *SIGGRAPH* (2011).
- [183] Yangyan Li et al. “Pointcnn: Convolution on x-transformed points”. In: *NeurIPS* (2018).

- [184] You Li and Javier Ibanez-Guzman. “Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems”. In: *IEEE Signal Processing Magazine* (2020).
- [185] Zhidong Liang et al. “RangeIoUDet: Range Image Based Real-Time 3D Object Detector Optimized by Intersection Over Union”. In: *CVPR* (2021).
- [186] Yiyi Liao, Jun Xie, and Andreas Geiger. “KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2D and 3D”. In: *Transactions on Pattern Analysis and Machine Intelligence* (2022).
- [187] Kevin Lin, Lijuan Wang, and Zicheng Liu. “End-to-end human pose and mesh reconstruction with transformers”. In: *CVPR* (2021).
- [188] Yangbin Lin et al. “Toward better boundary preserved supervoxel segmentation for 3D point clouds”. In: *ISPRS* (2018).
- [189] Or Litany et al. “Deep functional maps: Structured prediction for dense shape correspondence”. In: *ICCV* (2017).
- [190] Weixiao Liu et al. “Robust and accurate superquadric recovery: a probabilistic approach”. In: *CVPR* (2022).
- [191] Ze Liu et al. “Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows”. In: *ICCV* (2021).
- [192] Francesco Locatello et al. “Object-centric learning with slot attention”. In: *NeurIPS* (2020).
- [193] Suresh K Lodha et al. “Aerial LiDAR data classification using support vector machines (SVM)”. In: *International Symposium on 3D Data Processing, Visualization, and Transmission* (2006).
- [194] Romain Loiseau, Mathieu Aubry, and Loic Landrieu. “Online Segmentation of LiDAR Sequences: Dataset and Algorithm”. In: *ECCV* (2022).
- [195] Romain Loiseau et al. “Representing Shape Collections with Alignment-Aware Linear Models”. In: *3DV* (2021).
- [196] Romain Loiseau et al. “A Model You Can Hear: Audio Identification with Playable Prototypes”. In: *ISMIR* (2022).
- [197] Romain Loiseau et al. “Learnable Earth Parser: Discovering 3D Prototypes in Aerial Scans”. In: *arXiv* (2023).
- [198] William E Lorensen and Harvey E Cline. “Marching cubes: A high resolution 3D surface construction algorithm”. In: *SIGGRAPH* (1987).
- [199] David G Lowe. “Object recognition from local scale-invariant features”. In: *ICCV* (1999).
- [200] David G Lowe. “Local feature view clustering for 3D object recognition”. In: *CVPR* (2001).
- [201] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. “Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model”. In: *arXiv* (2022).

- [202] Niko Lukač et al. “Buildings roofs photovoltaic potential assessment based on LiDAR (Light Detection And Ranging) data”. In: *Energy* (2014).
- [203] Zixin Luo et al. “Geodesc: Learning local descriptors by integrating geometry constraints”. In: *ECCV* (2018).
- [204] Zixin Luo et al. “Contextdesc: Local descriptor augmentation with cross-modality context”. In: *CVPR* (2019).
- [205] J MacQueen. “Classification and analysis of multivariate observations”. In: *5th Berkeley Symp. Math. Statist. Probability* (1967).
- [206] Naureen Mahmood et al. “AMASS: Archive of motion capture as surface shapes”. In: *ICCV* (2019).
- [207] Franco Manessi, Alessandro Rozza, and Mario Manzo. “Dynamic graph convolutional networks”. In: *Pattern Recognition* (2020).
- [208] Sivabalan Manivasagam et al. “Lidarsim: Realistic lidar simulation by leveraging the real world”. In: *CVPR* (2020).
- [209] Jiageng Mao et al. “One Million Scenes for Autonomous Driving: ONCE Dataset”. In: *NeurIPS Datasets and Benchmarks Track* (2021).
- [210] Jiageng Mao et al. “Voxel Transformer for 3D Object Detection”. In: *ICCV* (2021).
- [211] Yanis Marchand, Bruno Vallet, and Laurent Caraffa. “Evaluating Surface Mesh Reconstruction of Open Scenes”. In: *ISPRS* (2021).
- [212] Zoltan Csaba Marton, Radu Bogdan Rusu, and Michael Beetz. “On fast surface reconstruction methods for large and noisy point clouds”. In: *ICRA* (2009).
- [213] Jonathan Masci et al. “Geodesic convolutional neural networks on riemannian manifolds”. In: (2015).
- [214] Daniel Maturana and Sebastian Scherer. “Voxnet: A 3D convolutional neural network for real-time object recognition”. In: *IROS* (2015).
- [215] Pavel Maur. “Delaunay triangulation in 3d”. In: *Technical Report, Departmen. of Computer Science and Engineering* (2002).
- [216] John McCormac et al. “SceneNet RGB-D: Can 5M Synthetic Images Beat Generic ImageNet Pre-training on Indoor Segmentation?” In: *ICCV* (2017).
- [217] Donald McKenzie and L Crystal. “Modeling Understory Vegetation and Its Response to Fire”. In: *Models for Planning Wildlife Conservation in Large Landscapes* (2011).
- [218] Donald Meagher. “Geometric modeling using octree encoding”. In: *CGIP* (1982).
- [219] Jie Mei et al. “3D tree modeling from incomplete point clouds via optimization and L 1-MST”. In: *International Journal of Geographical Information Science* (2017).

- [220] Mohamed El-Mekawy, Anders Östman, and Ihab Hijazi. “A unified building model for 3D urban GIS”. In: *ISPRS International Journal of Geo-Information* ().
- [221] Facundo Mémoli and Guillermo Sapiro. “A theoretical and computational framework for isometry invariant recognition of point cloud data”. In: *Foundations of Computational Mathematics* (2005).
- [222] Lars Mescheder et al. “Occupancy networks: Learning 3D reconstruction in function space”. In: *CVPR* (2019).
- [223] Gal Metzer et al. “Orienting point clouds with dipole propagation”. In: *ACM Transactions on Graphics (TOG)* (2021).
- [224] Francesco Milano et al. “Primal-dual mesh convolutional neural networks”. In: *NeurIPS* (2020).
- [225] Parham A Mirzaei. “Recent challenges in modeling of urban heat island”. In: *Sustainable Cities and Society* (2015).
- [226] Anastasiia Mishchuk et al. “Working hard to know your neighbor’s margins: Local descriptor learning loss”. In: *NeurIPS* (2017).
- [227] Ishan Misra, Rohit Girdhar, and Armand Joulin. “An end-to-end transformer model for 3D object detection”. In: *ICCV* (2021).
- [228] Niloy J Mitra and An Nguyen. “Estimating surface normals in noisy point cloud data”. In: *Computational Geometry* (2003).
- [229] Fabrice Monnier, Bruno Vallet, and Bahman Soheilian. “Trees detection from laser point clouds acquired in dense urban areas by a mobile mapping system”. In: *ISPRS* (2012).
- [230] Tom Monnier, Thibault Groueix, and Mathieu Aubry. “Deep Transformation-Invariant Clustering”. In: *NeurIPS* (2020).
- [231] Tom Monnier et al. “Unsupervised layered image decomposition into object prototypes”. In: *ICCV* (2021).
- [232] Federico Monti et al. “Geometric deep learning on graphs and manifolds using mixture model cnns”. In: *CVPR* (2017).
- [233] Pierre Moreels and Pietro Perona. “Evaluation of features detectors and descriptors based on 3D objects”. In: *IJCV* (2007).
- [234] Felix Morsdorf et al. “LIDAR-based geometric reconstruction of boreal type forest stands at single tree level for forest and wildland fire management”. In: *Remote Sensing of Environment* (2004).
- [235] Nur Atirah Muhadi et al. “The use of LiDAR-derived DEM in flood applications: A review”. In: *Remote Sensing* (2020).
- [236] Liangliang Nan and Peter Wonka. “Polyfit: Polygonal surface reconstruction from point clouds”. In: *ICCV* (2017).
- [237] Javier Naranjo-Alcazar et al. “Open set audio classification using autoencoders trained on few data”. In: *Sensors* (2020).

- [238] Netherlands. *Current Elevation File Netherlands*. URL: <https://www.ahn.nl/> (visited on 05/10/2023).
- [239] Gerhard Neuhold et al. “The mapillary vistas dataset for semantic understanding of street scenes”. In: *CVPR* (2017).
- [240] Joachim Niemeyer, Franz Rottensteiner, and Uwe Soergel. “Contextual classification of lidar data and building object detection in urban areas”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* (2014).
- [241] Pierre Novikoff. “De l’interaction perception-navigation en robotique mobile”. PhD thesis. Paris 6, 1991.
- [242] Maks Ovsjanikov et al. “Functional maps: a flexible representation of maps between shapes”. In: *ACM Transactions on Graphics (ToG)* (2012).
- [243] Onur Özyeşil et al. “A survey of structure from motion.” In: *Acta Numerica* (2017).
- [244] G Dias Pais et al. “3DRegNet: A deep neural network for 3D point registration”. In: *CVPR* (2020).
- [245] Xuran Pan et al. “3D object detection with pointformer”. In: *CVPR* (2021).
- [246] Vassil Panayotov et al. “Librispeech: an asr corpus based on public domain audio books”. In: *ICASSP* (2015).
- [247] Yatian Pang et al. “Masked autoencoders for point cloud self-supervised learning”. In: *ECCV* (2022).
- [248] Nicolas Paparoditis et al. “Stereopolis II: A multi-purpose and multi-sensor 3D mobile mapping system for street visualisation and 3D metrology”. In: *Revue française de photogrammétrie et de télédétection* (2012).
- [249] Jeremie Papon et al. “Voxel cloud connectivity segmentation-supervoxels for point clouds”. In: *CVPR* (2013).
- [250] Jeong Joon Park et al. “Deepsdf: Learning continuous signed distance functions for shape representation”. In: *CVPR* (2019).
- [251] Despoina Paschalidou, Luc Van Gool, and Andreas Geiger. “Learning unsupervised hierarchical part decomposition of 3D objects from a single rgb image”. In: *CVPR* (2020).
- [252] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. “Superquadrics revisited: Learning 3D shape parsing beyond cuboids”. In: *CVPR* (2019).
- [253] Despoina Paschalidou et al. “Neural parts: Learning expressive 3D shape abstractions with invertible neural networks”. In: *CVPR* (2021).
- [254] Mark Pauly, Richard Keiser, and Markus Gross. “Multi-scale feature extraction on point-sampled surfaces”. In: *Computer Graphics Forum* (2003).

- [255] Adrien Poulenard and Maks Ovsjanikov. “Multi-directional geodesic neural networks via equivariant convolution”. In: *ACM Transactions on Graphics* (2018).
- [256] Charles Ruizhongtai Qi et al. “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *NeurIPS* (2017).
- [257] Charles R Qi et al. “Pointnet: Deep learning on point sets for 3D classification and segmentation”. In: *CVPR* (2017).
- [258] Xiaojuan Qi et al. “3d graph neural networks for rgbd semantic segmentation”. In: *ICCV* (2017).
- [259] Shenhan Qian et al. “UNIF: United Neural Implicit Functions for Clothed Human Reconstruction and Animation”. In: *ECCV* (2022).
- [260] Zheng Qin et al. “Geometric transformer for fast and robust point cloud registration”. In: *CVPR* (2022).
- [261] J Pena Queralta et al. “FPGA-based architecture for a low-cost 3D lidar design and implementation from multiple rotating 2D lidars with ROS”. In: *IEEE sensors* (2019).
- [262] Michaël Ramamonjisoa, Sinisa Stekovic, and Vincent Lepetit. “MonteBoxFinder: Detecting and Filtering Primitives to Fit a Noisy Point Cloud”. In: *ECCV* (2022).
- [263] Nikhila Ravi et al. “Accelerating 3D Deep Learning with PyTorch3D”. In: *SIGGRAPH* (2020).
- [264] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. “Octnet: Learning deep 3D representations at high resolutions”. In: *CVPR* (2017).
- [265] Gilberto Rivera et al. “LiDAR applications in precision agriculture for cultivating crops: A review of recent advances”. In: *Computers and Electronics in Agriculture* (2023).
- [266] Damien Robert, Hugo Raquet, and Loic Landrieu. “Efficient 3D Semantic Segmentation with Superpoint Transformer”. In: *arXiv* (2023).
- [267] Damien Robert, Bruno Vallet, and Loic Landrieu. “Learning Multi-View Aggregation In the Wild for Large-Scale 3D Semantic Segmentation”. In: *CVPR* (2022).
- [268] Lawrence G. Roberts. “Machine perception of three-dimensional solids”. PhD thesis. Massachusetts Institute of Technology, 1963.
- [269] Fanny Roche et al. “Autoencoders for music sound modeling: a comparison of linear, shallow, deep, recurrent and variational models”. In: *Sound and Music Computing* (2018).
- [270] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *MICCAI* (2015).
- [271] Stefan Roth and Michael J Black. “Specular flow and the recovery of surface structure”. In: (2006).

- [272] Xavier Roynard, Jean-Emmanuel Deschaud, and François Goulette. “Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification”. In: *The International Journal of Robotics Research* (2018).
- [273] Santiago Royo and Maria Ballesta-Garcia. “An overview of LiDAR imaging systems for autonomous vehicles”. In: *Applied Sciences* (2019).
- [274] David Rozenberszki, Or Litany, and Angela Dai. “Language-grounded indoor 3D semantic segmentation in the wild”. In: *ECCV* (2022).
- [275] Raif M Rustamov et al. “Laplace-Beltrami eigenfunctions for deformation invariant shape representation”. In: *Symposium on geometry processing* (2007).
- [276] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. “Dynamic routing between capsules”. In: *NeurIPS* (2017).
- [277] Aditya Sanghi. “Info3d: Representation learning on 3D objects using mutual information maximization and contrastive learning”. In: *ECCV* (2020).
- [278] Aswin C Sankaranarayanan et al. “Specular surface reconstruction from sparse reflection correspondences”. In: *CVPR* (2010).
- [279] Jonathan Sauder and Bjarne Sievers. “Self-supervised deep learning on point clouds by reconstructing space”. In: *NeurIPS* (2019).
- [280] Manolis Savva, Angel X. Chang, and Pat Hanrahan. “Semantically-Enriched 3D Models for Common-sense Knowledge”. In: *CVPR Workshop* (2015).
- [281] Maximilian Scherer, Michael Walter, and Tobias Schreck. *Histograms of oriented gradients for 3D object retrieval*. Václav Skala-UNION Agency, 2010.
- [282] Nico Schertler, Bogdan Savchynskyy, and Stefan Gumhold. “Towards globally optimal normal orientations for large point clouds”. In: *Computer Graphics Forum* (2017).
- [283] Ruwen Schnabel, Patrick Degener, and Reinhard Klein. “Completion and reconstruction with primitive shapes”. In: *Computer Graphics Forum* (2009).
- [284] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. “Efficient RANSAC for point-cloud shape detection”. In: *Computer Graphics Forum* (2007).
- [285] John Secord and Avidesh Zakhor. “Tree detection in urban regions using aerial lidar and image data”. In: *Geoscience and Remote Sensing Letters* (2007).
- [286] Jaime Sevilla et al. “Compute trends across three eras of machine learning”. In: *IJCNN* (2022).
- [287] Abhishek Sharma and Maks Ovsjanikov. “Weakly supervised deep functional maps for shape matching”. In: *NeurIPS* (2020).

- [288] Gopal Sharma et al. “Csgnet: Neural shape parser for constructive solid geometry”. In: *CVPR* (2018).
- [289] Gopal Sharma et al. “Parsenet: A parametric surface fitting network for 3D point clouds”. In: *ECCV* (2020).
- [290] Xi Shen et al. “Ransac-flow: generic two-stage image alignment”. In: *ECCV* (2020).
- [291] Hoi Sheung and Charlie CL Wang. “Robust mesh reconstruction from unoriented noisy points”. In: *SIAM/ACM Joint Conference on Geometric and Physical Modeling* (2009).
- [292] Weijing Shi and Raj Rajkumar. “Point-gnn: Graph neural network for 3D object detection in a point cloud”. In: *CVPR* (2020).
- [293] Ioannis Siglidis et al. “The Learnable Typewriter: A Generative Approach to Text Line Analysis”. In: *arXiv* (2023).
- [294] Nina M Singer and Vijayan K Asari. “DALES objects: A large scale benchmark dataset for instance segmentation in aerial LiDAR”. In: *IEEE Access* (2021).
- [295] Lawrence Sirovich and Michael Kirby. “Low-dimensional procedure for the characterization of human faces”. In: *Journal of the Optical Society of America* (1987).
- [296] Dmitriy Smirnov et al. “Marionette: Self-supervised sprite learning”. In: *NeurIPS* (2021).
- [297] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. “Sun rgb-d: A rgb-d scene understanding benchmark suite”. In: *CVPR* (2015).
- [298] Shuran Song et al. “Semantic scene completion from a single depth image”. In: *CVPR* (2017).
- [299] Robin Strudel et al. “Segmenter: Transformer for Semantic Segmentation”. In: *ICCV* (2021).
- [300] Hang Su et al. “Splatnet: Sparse lattice networks for point cloud processing”. In: *CVPR* (2018).
- [301] Chenxin Sun et al. “Individual tree crown segmentation and crown width extraction from a heightmap derived from aerial laser scanning data using a deep learning framework”. In: *Frontiers in plant science* (2022).
- [302] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. “A concise and provably informative multi-scale signature based on heat diffusion”. In: *Computer graphics forum* (2009).
- [303] Pei Sun et al. “Scalability in perception for autonomous driving: Waymo open dataset”. In: *CVPR* (2020).
- [304] Pei Sun et al. “RSN: Range Sparse Net for Efficient, Accurate LiDAR 3D Object Detection”. In: *CVPR* (2021).

- [305] Minhyuk Sung et al. “Data-driven structural priors for shape completion”. In: *ACM Transactions on Graphics* (2015).
- [306] Weikai Tan et al. “Toronto-3D: A large-scale mobile LiDAR dataset for semantic segmentation of urban roadways”. In: *CVPR Workshop* (2020).
- [307] Haotian Tang et al. “Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution”. In: *ECCV* (2020).
- [308] An Tao. *Unsupervised Point Cloud Reconstruction for Classific Feature Learning*. 2020. URL: <https://github.com/AnTao97/UnsupervisedPointCloudReconstruction>.
- [309] Maxim Tatarchenko et al. “Tangent convolutions for dense prediction in 3d”. In: *CVPR* (2018).
- [310] Maxim Tatarchenko et al. “What do single-view 3D reconstruction networks learn?” In: *CVPR* (2019).
- [311] Hugues Thomas et al. “KPConv: Flexible and Deformable Convolution for Point Clouds”. In: *ICCV* (2019).
- [312] Yurun Tian, Bin Fan, and Fuchao Wu. “L2-net: Deep learning of discriminative patch descriptor in euclidean space”. In: *CVPR* (2017).
- [313] Noé Tits et al. “Visualization and interpretation of latent spaces for controlling expressive speech synthesis through audio analysis”. In: *Conference of the International Speech Communication Association* (2019).
- [314] Garvita Tiwari et al. “Neural-gif: Neural generalized implicit functions for animating people in clothing”. In: *ICCV* (2021).
- [315] Carlo Tomasi and Takeo Kanade. “Shape and motion from image streams under orthography: a factorization method”. In: *IJCV* (1992).
- [316] Lorenzo Torresani, Aaron Hertzmann, and Chris Bregler. “Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors”. In: *Transactions on Pattern Analysis and Machine Intelligence* (2008).
- [317] Shubham Tulsiani et al. “Learning shape abstractions by assembling volumetric primitives”. In: *CVPR* (2017).
- [318] Greg Turk and James F O’Brien. “Shape transformation using variational implicit functions”. In: *SIGGRAPH* (2005).
- [319] Matthew A Turk and Alex P Pentland. “Eigenfaces for recognition”. In: *Journal of cognitive neuroscience* (1991).
- [320] Matthew A Turk and Alex P Pentland. “Face recognition using eigenfaces”. In: *CVPR* (1991).
- [321] Konstantinos Tzevanidis et al. “From multiple views to textured 3D meshes: a gpu-powered approach”. In: *ECCV Trends and Topics in Computer Vision Workshop* (2012).

- [322] Mikaela Angelina Uy et al. “Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data”. In: *ICCV* (2019).
- [323] Valeo Schalter und Sensoren GmbH. “LiDAR device for a vehicle and method for increasing the detection range of a corresponding LiDAR device”. WO2020/187677. Apr. 10, 2020.
- [324] Bruno Vallet et al. “TerraMobilita/iQmulus urban point cloud analysis benchmark”. In: *Computers & Graphics* (2015).
- [325] Nina Varney, Vijayan K Asari, and Quinn Graehling. “DALES: A Large-scale Aerial LiDAR Data Set for Semantic Segmentation”. In: *CVPR Workshop* (2020).
- [326] Ashish Vaswani et al. “Attention is all you need”. In: *NeurIPS* (2017).
- [327] Jari Vauhkonen et al. “Comparative testing of single-tree detection algorithms under different types of forest”. In: *Forestry* (2012).
- [328] Nitika Verma, Edmond Boyer, and Jakob Verbeek. “Feastnet: Feature-steered graph convolutions for 3D shape analysis”. In: *CVPR* (2018).
- [329] Elliot Vincent, Jean Ponce, and Mathieu Aubry. “Pixel-wise Agricultural Image Time Series Classification: Comparisons and a Deformable Prototype-based Approach”. In: *arXiv* (2023).
- [330] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. “Fast transformers with clustered attention”. In: *NeurIPS* (2020).
- [331] Hanchen Wang et al. “Unsupervised point cloud pre-training via occlusion completion”. In: *ICCV* (2021).
- [332] Lingjing Wang, Xiang Li, and Yi Fang. “Few-Shot Learning of Part-Specific Probability Space for 3D Shape Segmentation”. In: *CVPR* (2020).
- [333] Luqi Wang et al. “Fire risk assessment for building operation and maintenance based on BIM technology”. In: *Building and Environment* (2021).
- [334] Ruisheng Wang, Jiju Peethambaran, and Dong Chen. “Lidar point clouds to 3-D urban models : A review”. In: *Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2018).
- [335] Yue Wang et al. “Dynamic graph cnn for learning on point clouds”. In: *ACM Transactions on Graphics* (2019).
- [336] Yu Wang et al. “Linear subspace design for real-time shape deformation”. In: *Transactions on Graphics* (2015).
- [337] GK Wedajo. “LiDAR DEM Data for flood mapping and assessment; opportunities and challenges: A Review”. In: *J. Remote Sens. Gis* (2017).
- [338] Martin Weinmann, Boris Jutzi, and Clément Mallet. “Feature relevance assessment for the semantic interpretation of 3D point cloud data”. In: *ISPRS Laser Scanning Workshop* (Nov. 2013).

- [339] Martin Weinmann et al. “Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* (2015).
- [340] Ben G Weinstein et al. “Individual tree-crown detection in RGB imagery using semi-supervised deep learning neural networks”. In: *Remote Sensing* (2019).
- [341] Xin Wen et al. “Point2SpatialCapsule: Aggregating features and spatial relationships of local regions on point clouds using spatial-aware capsules”. In: *IEEE Transactions on Image Processing* (2020).
- [342] Karen F West et al. “Context-driven automated target detection in 3D data”. In: *Automatic Target Recognition* (2004).
- [343] Svante Wold, Kim Esbensen, and Paul Geladi. “Principal component analysis”. In: *Chemometrics and intelligent laboratory systems* (1987).
- [344] Kan Wu et al. “Rethinking and improving relative position encoding for vision transformer”. In: *ICCV* (2021).
- [345] Tong Wu et al. “Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation”. In: *CVPR* (2023).
- [346] Yuwei Wu et al. “Primitive-Based Shape Abstraction via Nonparametric Bayesian Inference”. In: *ECCV* (2022).
- [347] Zhirong Wu et al. “3D ShapeNets: A deep representation for volumetric shapes”. In: *CVPR* (2015).
- [348] Michael A Wulder et al. “Lidar sampling for large-area forest characterization: A review”. In: *Remote Sensing of Environment* (2012).
- [349] Shaobo Xia et al. “Geometric primitives in LiDAR point clouds: A review”. In: *Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2020).
- [350] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. “Sun3d: A database of big spaces reconstructed using sfm and object labels”. In: *ICCV* (2013).
- [351] Saining Xie et al. “Pointcontrast: Unsupervised pre-training for 3D point cloud understanding”. In: *ECCV* (2020).
- [352] Siming Yan et al. “Implicit autoencoder for point cloud self-supervised representation learning”. In: *arXiv* (2022).
- [353] Yaoqing Yang et al. “Foldingnet: Point cloud auto-encoder via deep grid deformation”. In: *CVPR* (2018).
- [354] Zetong Yang et al. “A unified query-based paradigm for point cloud understanding”. In: *CVPR* (2022).
- [355] Zhen Ye et al. “LASDU: A large-scale aerial lidar dataset for semantic labeling in dense urban areas”. In: *ISPRS International Journal of Geo-Information* (2020).

- [356] Li Yi et al. “Syncspecnn: Synchronized spectral cnn for 3d shape segmentation”. In: *CVPR* (2017).
- [357] Yang You et al. “Pointwise rotation-invariant network with adaptive sampling and 3D spherical voxel convolution”. In: *Artificial Intelligence* (2020).
- [358] Alex Yu et al. “Plenotrees for real-time rendering of neural radiance fields”. In: *ICCV* (2021).
- [359] Bailang Yu et al. “Automated derivation of urban building density information using airborne LiDAR data and object-based method”. In: *Landscape and Urban Planning* (2010).
- [360] Hong-Xing Yu, Leonidas J. Guibas, and Jiajun Wu. “Unsupervised Discovery of Object Radiance Fields”. In: *ICLR* (2022).
- [361] Xumin Yu et al. “Point-bert: Pre-training 3D point cloud transformers with masked point modeling”. In: *CVPR* (2022).
- [362] Ekim Yurtsever et al. “A survey of autonomous driving: Common practices and emerging technologies”. In: *IEEE access* (2020).
- [363] Andrei Zaharescu et al. “Surface feature detection and description with applications to mesh matching”. In: *CVPR* (2009).
- [364] Manzil Zaheer et al. “Deep sets”. In: *NeurIPS* (2017).
- [365] Georgios Zamanakos et al. “A comprehensive survey of LIDAR-based 3D object detection methods with deep learning for autonomous driving”. In: *Computers & Graphics* (2021).
- [366] Dimitris Zarpalas, Georgios Kordelas, and Petros Daras. “Recognizing 3D objects in cluttered scenes using projection images”. In: *ICIP* (2011).
- [367] Ling Zhang and Zhigang Zhu. “Unsupervised feature learning for point cloud understanding by contrasting and clustering using graph convolutional neural networks”. In: *3DV* (2019).
- [368] Yang Zhang et al. “PolarNet: An Improved Grid Representation for Online LiDAR Point Clouds Semantic Segmentation”. In: *CVPR* (2020).
- [369] Yu Zhang et al. “A survey on neural network interpretability”. In: *IEEE Transactions on Emerging Topics in Computational Intelligence* (2021).
- [370] Zaiwei Zhang et al. “Self-supervised pretraining of 3D features on any point-cloud”. In: *ICCV* (2021).
- [371] Zhiyuan Zhang et al. “Rotation invariant convolutions for 3D point clouds deep learning”. In: *3DV* (2019).
- [372] Chunhong Zhao, Qihao Weng, and Anna M Hersperger. “Characterizing the 3-D urban morphology transformation to understand urban-form dynamics: A case study of Austin, Texas, USA”. In: *Landscape and Urban Planning* (2020).
- [373] Hengshuang Zhao et al. “Point transformer”. In: *CVPR* (2021).

- [374] Na Zhao, Tat-Seng Chua, and Gim Hee Lee. “Few-shot 3D point cloud semantic segmentation”. In: *CVPR* (2021).
- [375] Yongheng Zhao et al. “3D point capsule networks”. In: *CVPR* (2019).
- [376] Yongheng Zhao et al. “Quaternion equivariant capsule networks for 3D point clouds”. In: *ECCV* (2020).
- [377] Yongheng Zhao et al. “3DPointCaps++: Learning 3D Representations with Capsule Networks”. In: *IJCV* (2022).
- [378] Zerong Zheng et al. “Deep implicit templates for 3D shape representation”. In: *CVPR* (2021).
- [379] Guozheng Zhi et al. “Urban flood risk assessment and analysis with a 3D visualization method coupling the PP-PSO algorithm and building data”. In: *Journal of Environmental Management* (2020).
- [380] Zixiang Zhou, Yang Zhang, and Hassan Foroosh. “Panoptic-PolarNet: Proposal-free LiDAR Point Cloud Panoptic Segmentation”. In: *CVPR* (2021).
- [381] Xinge Zhu et al. “Cylindrical and asymmetrical 3D convolution networks for LiDAR segmentation”. In: *CVPR* (2021).
- [382] Pablo Zinemanas et al. “An Interpretable Deep Learning Model for Automatic Sound Classification”. In: *Electronics* (2021).
- [383] SM Zolanvari et al. “DublinCity: Annotated LiDAR point cloud and its applications”. In: *BMVC* (2019).
- [384] Chuhan Zou et al. “3d-prnn: Generating shape primitives with recurrent neural networks”. In: *ICCV* (2017).

List of Figures

| | | |
|------|---|----|
| 1.1 | 3D Representations and Applications | 2 |
| 1.2 | 3D Data Usage for Public Policies | 4 |
| 1.3 | Global Trends in AI's Efficiency and Interest | 6 |
| 1.4 | Thesis Timeline | 10 |
| 1.5 | Thesis Outline | 13 |
| 2.1 | 3D Geometric Representations | 16 |
| 2.2 | 3D Deep Learning Paradigms | 23 |
| 2.3 | 3D Tasks | 26 |
| 2.4 | U-Net Architecture | 28 |
| 2.5 | 3D Datasets | 31 |
| 3.1 | Online LiDAR Segmentation | 35 |
| 3.2 | Coverage from HelixNet | 40 |
| 3.3 | Extracts from HelixNet | 41 |
| 3.4 | Localization of the Sequences | 42 |
| 3.5 | Sensor Acquisition Geometry | 43 |
| 3.6 | Helix4D Architecture | 45 |
| 3.7 | Spatio-Temporal Attention | 47 |
| 3.8 | Relative Positional Encoding Bins | 48 |
| 3.9 | Positional Encoding | 49 |
| 3.10 | Data Partitioning Schemes | 50 |
| 3.11 | Influence of Slice Size | 53 |
| 4.1 | Discovered Linear Models | 57 |
| 4.2 | Method Overview | 59 |
| 4.3 | Learned Prototypes and Comparisons | 65 |
| 4.4 | Influence of Degrees of Freedom D on ModelNet | 67 |
| 4.5 | Basis Vectors | 69 |
| 4.6 | Modeling ModelNet10 | 71 |
| 4.7 | Visualizing Reconstructions on ABC | 72 |
| 4.8 | Visualizing Reconstructions on ShapeNet | 73 |
| 4.9 | Model Selection | 76 |
| 4.10 | Reconstruction Results | 78 |
| 5.1 | Learnable Earth Parser | 83 |
| 5.2 | Method Overview | 85 |

| | | |
|------|--|-----|
| 5.3 | Earth Parser Dataset | 89 |
| 5.4 | Reconstruction Quality | 92 |
| 5.5 | Learnable Earth Parser Detailed Architecture | 95 |
| 5.6 | Instance Segmentation | 104 |
| 5.7 | Semantic Segmentation | 105 |
| 5.8 | Learned Prototypes | 105 |
| 5.9 | Qualitative Results | 106 |
| 6.1 | A Model You Can Hear Overview | 113 |
| A.1 | Aperçu de la thèse | 144 |
| A.2 | Modalités 3D | 145 |
| A.3 | Tâches 3D | 147 |
| A.4 | Jeux de données 3D | 148 |
| A.5 | HelixNet | 149 |
| A.6 | Helix4D | 150 |
| A.7 | Prototypes appris | 151 |
| A.8 | Architecture du modèle | 152 |
| A.9 | Résultats qualitatifs | 154 |
| A.10 | Learnable Earth Parser | 155 |
| A.11 | Earth Parser Dataset | 156 |

List of Tables

| | | |
|-----|--|-----|
| 3.1 | Embarked LiDAR Datasets with Semantic Point Annotations . | 39 |
| 3.2 | Semantic Segmentation Results | 51 |
| 3.3 | HelixNet Semantic Segmentation Scores | 52 |
| 3.4 | Ablation Study | 54 |
| 4.1 | Results on ModelNet10 | 74 |
| 4.2 | Non-Aligned Data | 75 |
| 4.3 | 10-shot Segmentation | 77 |
| 4.4 | Low-Shot Supervised Segmentation Results on ShapeNetPart . | 79 |
| 5.1 | Earth Parser Dataset | 94 |
| 5.2 | Earth Parser Dataset Classes and Localisation | 98 |
| 5.3 | Results on the Earth Parser Dataset | 101 |
| 5.4 | Ablation Study | 102 |
| 5.5 | Synthetic Shapes | 103 |
| A.1 | Resultats de segmentation sémantique | 150 |
| A.2 | Resultats sur ModelNet10 | 152 |
| A.3 | 10-shot segmentation | 153 |
| A.4 | Résultats sur le Earth Parser Dataset | 155 |

Chapter A

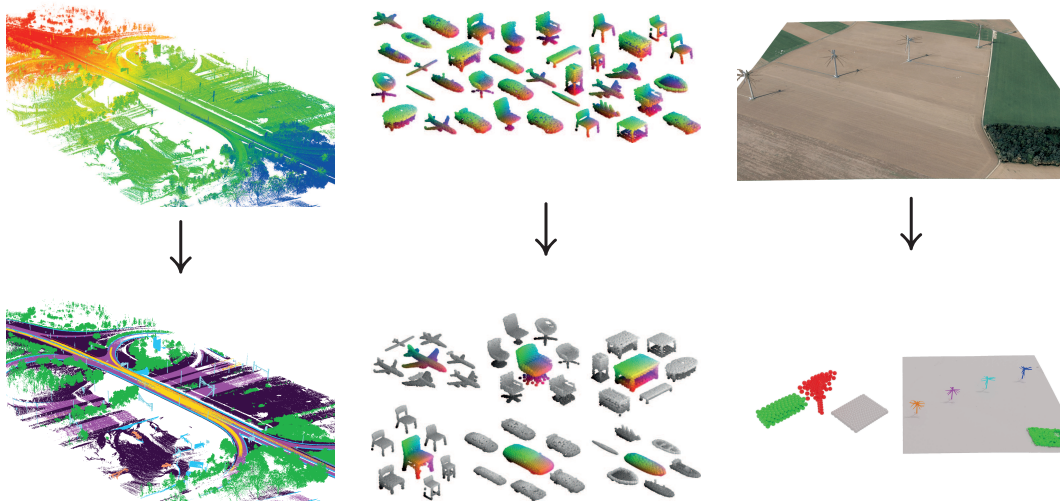
Résumé long en français

Analyse de Données 3D du Monde Réel : Efficacité et Interprétabilité

A.1 Introduction

Le traitement des données 3D a de nombreuses applications: dans l'industrie par exemple avec la conduite autonome, ou encore en appui aux politiques publiques pour la gestion du territoire. Dans cette thèse, nous nous intéressons à la conception de méthodes pour le traitement des données 3D, fonctionnant en temps réel, non supervisées ou interprétables. Ce chapitre détaille les objectifs, les motivations, les défis et les contributions de notre travail de recherche.

Notre principale motivation réside dans les défis et les opportunités du traitement des données 3D réelles. Nous pensons qu'il est crucial de concevoir des méthodes efficaces qui (i) répondent aux exigences d'applications telles que la conduite autonome, (ii) contribuent à réduire l'impact environnemental de l'intelligence artificielle en minimisant les besoins de calcul, et (iii) garantissent l'acceptabilité sociale des méthodes d'apprentissage via des modèles plus interprétables. L'impact potentiel des outils de traitement 3D sur l'action publique nous pousse à vouloir créer des méthodes qui peuvent être facilement adaptées aux besoins et applications spécifiques. De plus, nous sommes préoccupés par la diversité limitée des jeux de données du monde réel, que le domaine de la vision 3D utilise actuellement, et qui se concentrent souvent sur



(a) Chapitre 3. Segmentation sémantique des acquisitions LiDAR embarquées.

(b) Chapitre 4. Découverte de formes dans de grandes collections 3D.

(c) Chapitre 5. Analyse de données aériennes LiDAR non annotées du monde réel.

Figure A.1: **Aperçu de la thèse.** Dans cette thèse, nous abordons le traitement des données 3D, de la segmentation sémantique au regroupement et à la découverte de formes.

les zones urbaines denses. Ainsi, notre objectif est de contribuer à la création et au partage de jeux de données plus diversifiés qui représentent mieux les scénari du monde réel.

Le domaine de la vision par ordinateur 3D présente de nombreux défis à relever. Certains défis spécifiques ont retenu notre attention. Premièrement, la difficulté de traiter des scènes 3D complexes du monde réel présentant différents niveaux d'encombrement et d'occultation. Deuxièmement, la rareté des ensembles de données annotés permettant le développement des algorithmes. Et troisièmement, le défi de l'interprétabilité de l'apprentissage profond, qui est crucial pour comprendre et expliquer le fonctionnement interne des modèles complexes.

Cette thèse apporte une contribution significative au domaine de la vision 3D par ordinateur. Ce manuscrit présente trois projets principaux, dont deux ont été publiés dans des conférences et des ateliers internationaux, tandis que le troisième est en cours d'examen pour être publié dans une conférence internationale de vision par ordinateur. De plus, une extension de notre travail

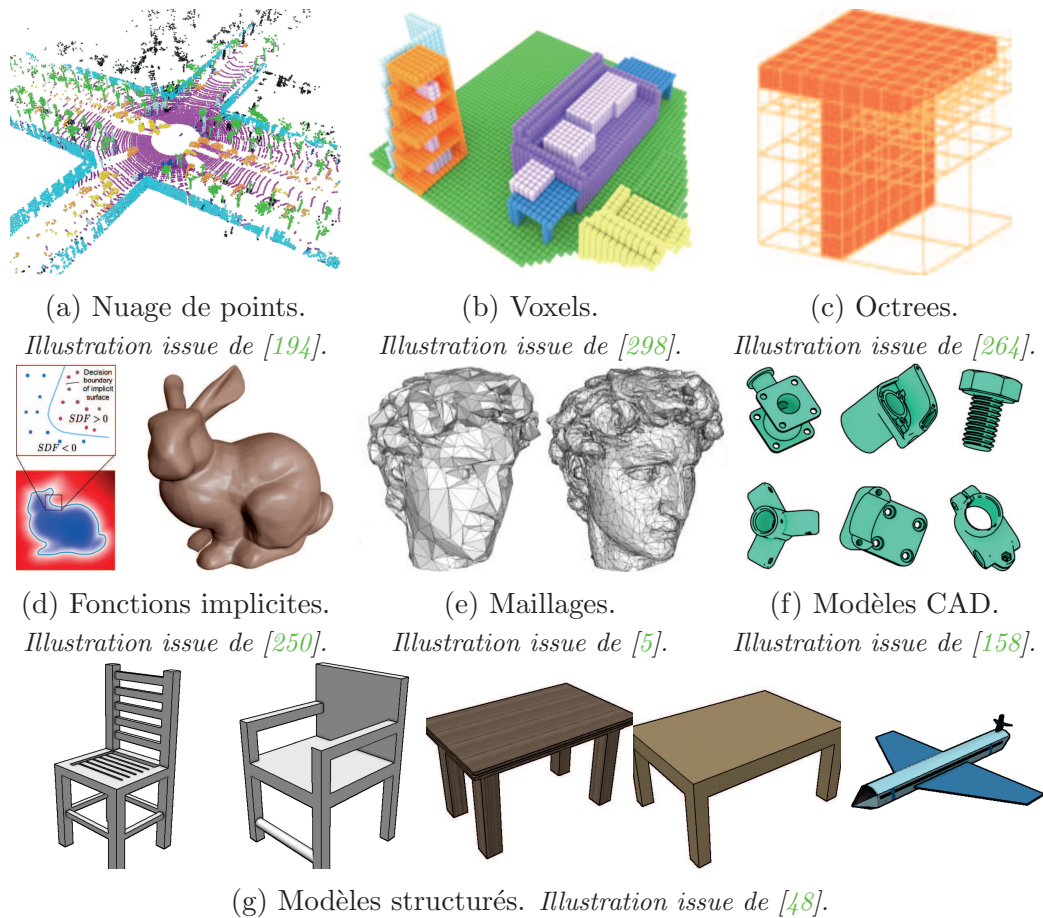


Figure A.2: **Modalités 3D.** Les données 3D peuvent être représentées de différentes manières en fonction de chaque objectif spécifique.

pour les données tridimensionnelles aux données audio a été présentée lors d'une conférence internationale d'analyse de données audio, et lors d'un atelier d'une conférence internationale de vision par ordinateur.

Comme vu en Figure A.1, cette thèse est structurée de la manière suivante :

Chapitre 2. État de l'art. Nous présentons les représentations, les tâches, les méthodes et les ensembles de données 3D les plus couramment utilisés dans le domaine.

Chapitre 3. Segmentation sémantique en temps réel de séquences LiDAR. Nous proposons une nouvelle approche, qui exploite la géométrie du capteur LiDAR pour concevoir une méthode de segmentation sémantique en temps réel surpassant les méthodes existantes en termes de latence, tout en préservant les performances de segmentation sémantique. Nous introduisons

également un ensemble de données annotées public, conçu pour évaluer et concevoir des méthodes de segmentation en temps réel.

Chapitre 4. Compréhension non supervisée de collections de formes 3D à grande échelle. Nous proposons d’apprendre des prototypes visibles pour comprendre des grandes collections de formes 3D, dans un contexte non supervisé. Nous présentons des résultats à l’état de l’art du regroupement de formes et de la segmentation sémantique avec très peu d’exemples annotés.

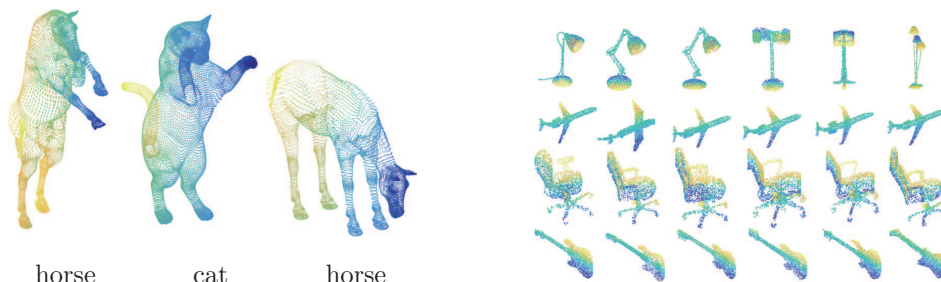
Chapitre 5. Prototypes appris en 3D pour des données du monde réel à grande échelle. Nous étendons le cadre d’apprentissage de prototypes 3D présenté dans le Chapitre 4 pour comprendre les acquisitions aériennes 3D du monde réel à grande échelle avec des prototypes appris. Nous introduisons également un ensemble de données LiDAR aériennes public acquis dans divers environnements, tels que des forêts, des zones urbaines et des ports de plaisance.

Chapitre 6. Conclusion. Dans ce chapitre, nous résumons nos contributions et discutons des principales limites de notre travail. Nous proposons également des pistes de recherche futures pour le traitement des données 3D.

A.2 État de l’art

Les avancées récentes dans la collecte et le traitement des données 3D ont été utilisées pour des applications telles que les véhicules autonomes, la gestion du territoire et l’analyse des données médicales. Ce chapitre présente les concepts clés du traitement des données 3D liés à notre travail. Nous commençons par discuter des représentations des données 3D, nous expliquons ensuite les tâches et les approches principales, et terminons par présenter les jeux de données 3D les plus utilisés.

Les principales façons de représenter et structurer les données 3D sont les suivantes : les nuages de points, les représentations basées sur les voxels, les fonctions implicites, les maillages et les représentations basées sur des primitives. Consultez la Figure A.2 pour des exemples.



(a) Classification de formes.

Illustration issue de [256].

(b) Regroupement de formes.

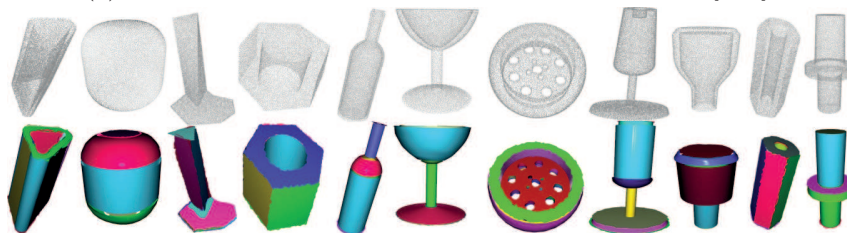
Illustration issue de [367].(c) Segmentation sémantique. *Illustration issue de [274].*(d) Découpage d'objets. *Illustration issue de [289].*

Figure A.3: **Tâches 3D.** Nous nous intéressons à plusieurs tâches de vision par ordinateur ayant pour objectif d'analyser des données 3D.

La caractérisation des formes est la première étape pour comprendre la structure des données 3D. Les descripteurs de forme doivent être discriminants et faciles à calculer. Avant l'avènement de l'apprentissage profond, les descripteurs 3D s'appuyaient principalement sur des caractéristiques géométriques [254, 342, 117, 281, 72, 170, 1], spectrales [275, 302, 41, 14] et globales.

Les réseaux neuronaux ont réussi à relever de nombreux défis en matière d'analyse de formes 3D. Les principales approches d'extraction de caractéristiques 3D peuvent être classifiées en fonction de la représentation des données 3D qu'elles utilisent : voxels [59], nuages de points [257], graphes [168], surfaces [42, 115], modèles structurés [174, 102], ou plus récemment des modèles

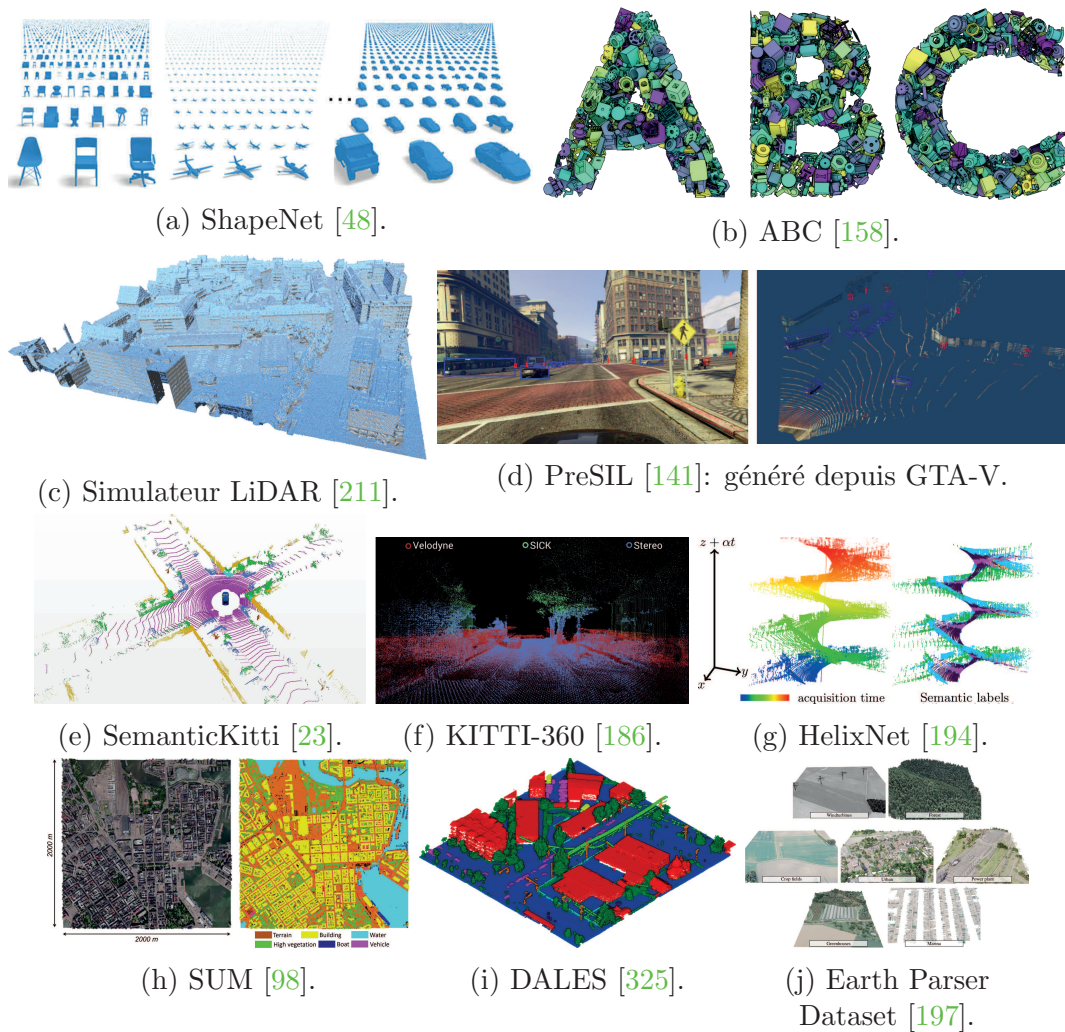


Figure A.4: **Jeux de données 3D**. Alors que les méthodes de traitement des données 3D sont de plus en plus performantes, leurs résultats et leur robustesse sont guidés en partie par l'augmentation croissante de la quantité de jeux de données 3D de bonne qualité.

volumétriques implicites [250, 56, 222]. L'apprentissage profond a été utilisé avec succès pour des tâches aussi diverses que la classification [207], la segmentation [311], la génération de formes [115, 310], la mise en correspondance [244], le débruitage [132] et la compression [138]. Nous détaillons les entrées et sorties de chaque tâche, expliquons les approches existantes et fournissons quelques exemples d'applications dans la Figure A.3.

Dans cette thèse, nous nous intéressons au traitement de nuages de points 3D, acquis à partir de capteurs 3D ou non. La littérature utilise beaucoup de jeux de données synthétiques et du monde réel différents pour évaluer

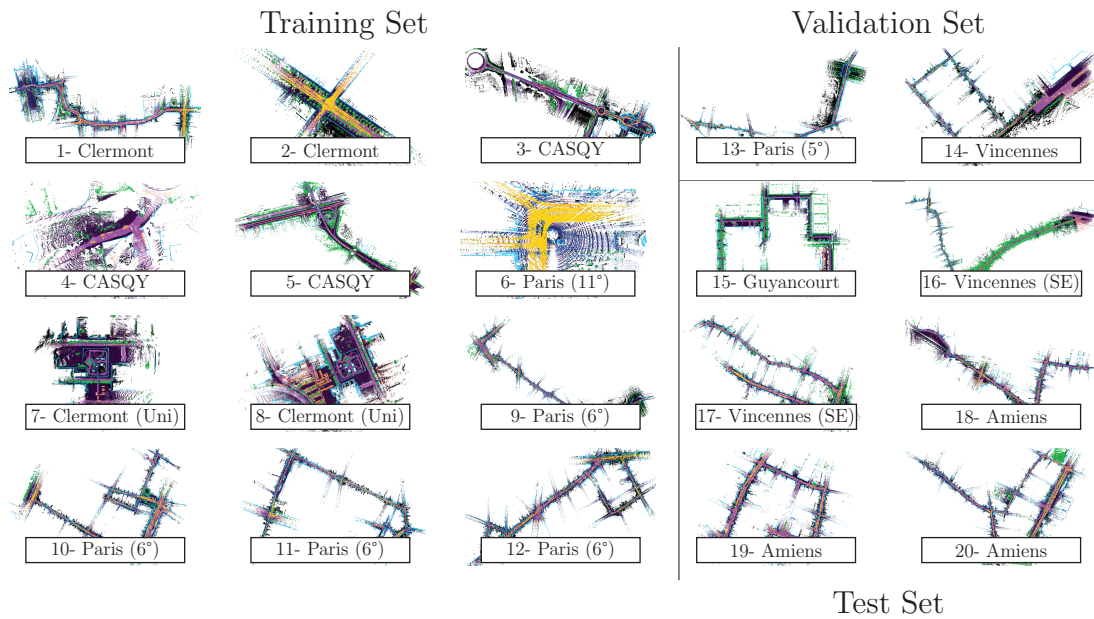


Figure A.5: **HelixNet**. Nous avons divisé les acquisitions en 12 séquences d’entraînement, 2 de validation et 6 de test. HelixNet contient des scènes dans différents environnements urbains provenant de capteurs statiques ou mobiles.

les méthodes. La Figure A.4 présente ces différents jeux de données. Nous présentons les principaux ensembles de données synthétiques centrés sur les objets couramment utilisés dans la communauté de la vision par ordinateur pour développer de nouvelles méthodes de traitement. Ensuite, nous décrivons des ensembles de données principalement utilisés dans la conduite autonome et collectés directement à partir de capteurs 3D. Enfin, nous présentons plusieurs ensembles de données LiDAR, qui sont acquis à partir de capteurs montés sur des plates-formes aériennes et offrent des défis uniques pour le traitement des nuages de points, en raison de leur grande échelle.

A.3 Segmentation automatique de flux LiDAR

Les capteurs LiDAR rotatifs montés sur les toits de véhicules sont largement utilisés par les véhicules autonomes. Cependant, la plupart des jeux de données sémantiques et des algorithmes, utilisés pour la segmentation des séquences LiDAR, utilisent des rotations complètes du capteur, entraînant une latence d’acquisition incompatible avec les applications en temps réel.

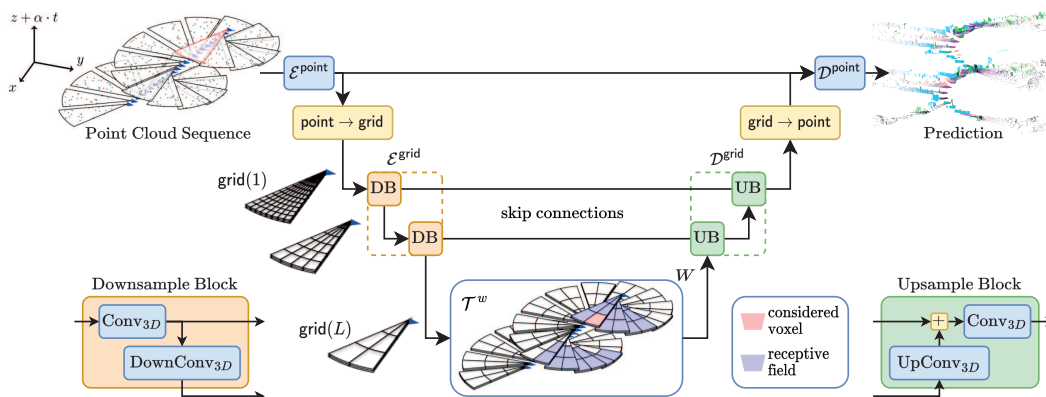


Figure A.6: **Helix4D**. Une séquence d’acquisition est divisée en tranches, dont les points sont encodés par $\mathcal{E}^{\text{point}}$ et agrégés dans une partition cylindrique fine. Un encodeur convolutionnel $\mathcal{E}^{\text{grid}}$ produit des descripteurs à des résolutions plus basses. Nous appliquons W mécanismes d’attention spatio-temporels \mathcal{T}^w consécutifs sur les voxels les plus larges, avec une attention s’étendant sur les tranches d’acquisition courantes et passées. Les descripteurs résultants sont échantillonnés à la résolution d’origine avec un décodeur convolutionnel $\mathcal{D}^{\text{grid}}$ en utilisant les descripteurs de l’encodeur aux résolutions intermédiaires. Enfin, les descripteurs sur la grille sont reprojétés sur les points, qui sont classés par $\mathcal{D}^{\text{point}}$.

Table A.1: **Resultats de segmentation sémantique**. Performance de Helix4D et des approches concurrentes sur HelixNet et sur l’ensemble de validation de SemanticKITTI*, avec en entrée 360 ou 72 degrés d’acquisition. Nous rapportons la moyenne de l’Intersection-over-Union (mIoU) et le temps d’inférence en ms. Les méthodes répondant à l’exigence de temps réel sont indiquées par ✓ et celles qui ne le sont pas par ✗. * SemanticKITTI est désigné par SK. Mesurer la latence sur cet ensemble de données nécessite de faire des approximations non-réalistes sur la position de la fibre.

| Méthode | Taille $\times 10^6$ | Rotation ● | | 104ms | $\frac{1}{5}$ rotation ► | | 21ms |
|---------------------|-------------------------|-------------|-------------|-------------|--------------------------|-------------|-------------|
| | | HelixNet | SK* | Inf. (ms) | HelixNet | SK* | Inf. (ms) |
| SalsaNeXt [69] | 6.7 | 69.4 | 55.8 | 23 ✓ | 68.2 | 55.6 | 10 ✓ |
| PolarNet [368] | 13.6 | 73.6 | 58.2 | 49 ✓ | 72.2 | 56.9 | 36 ✗ |
| Pan. PolarNet [380] | 13.7 | — | 64.5 | 50 ✓ | — | 60.3 | 44 ✗ |
| SPVNAS [307] | 10.8 | 73.4 | 64.7 | 73 ✓ | 69.9 | 57.8 | 44 ✗ |
| Cylinder3D [381] | 55.9 | 76.6 | 66.9 | 108 ✗ | 75.0 | 65.3 | 54 ✗ |
| Helix4D | 1.0 | 79.4 | 66.7 | 45 ✓ | 78.7 | 66.8 | 19 ✓ |

HelixNet. Pour résoudre ce problème, nous introduisons d’abord HelixNet (voir Figure A.5), un ensemble de données de 10 milliards de points annotés, contenant des horodatages et des informations de rotation de capteur nécessaires pour évaluer avec précision la capacité des algorithmes de segmen-

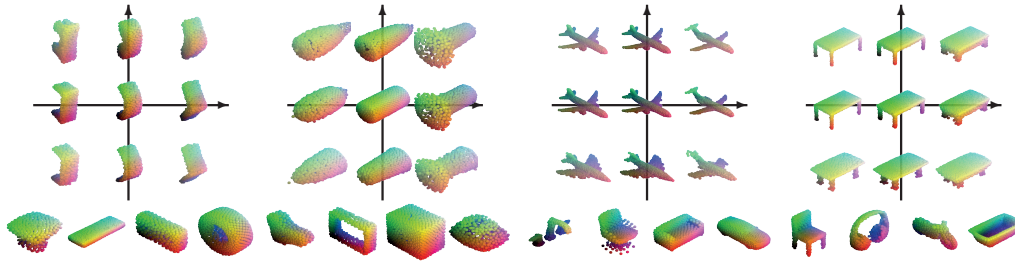


Figure A.7: **Prototypes appris.** Notre approche permet de découvrir sans supervision des modèles linéaires de formes à partir de grandes collections de formes. Nous montrons deux exemples de familles à deux dimensions et huit prototypes découverts pour ABC [158] (gauche) et ShapeNet [48] (droite).

tation à analyser les données en temps réel.

Helix4D. Deuxièmement, nous proposons Helix4D (voir Figure A.6), une architecture contenant un mécanisme d’attention spatio-temporel compact et efficace spécialement conçue pour les séquences LiDAR rotatives. Helix4D fonctionne sur des tranches d’acquisition correspondant à une fraction d’une rotation complète du capteur, réduisant ainsi considérablement la latence.

Comme montré en Table A.1, Helix4D atteint une précision comparable aux meilleurs algorithmes de segmentation sur HelixNet et SemanticKITTI, en divisant la latence par plus de 5 et la taille du modèle par plus de 50.

A.4 Exploration de collections de formes 3D

Dans ce chapitre, nous revisitons la représentation classique des nuages de points 3D avec des prototypes qui sont des modèles de formes linéaires. L’idée clé est d’utiliser l’apprentissage profond pour représenter une collection de formes 3D, sous forme de transformations affines de modèles de formes linéaires de faible dimension.

Modèles linéaires de forme. Chaque modèle linéaire est caractérisé par un prototype de forme, une base de forme de faible dimension, et deux réseaux neuronaux. Comme montré en Figure A.8, les réseaux prennent en entrée un nuage de points, et prédisent les coordonnées d’une forme dans la base linéaire, et la transformation affine qui approxime le mieux l’entrée. Les

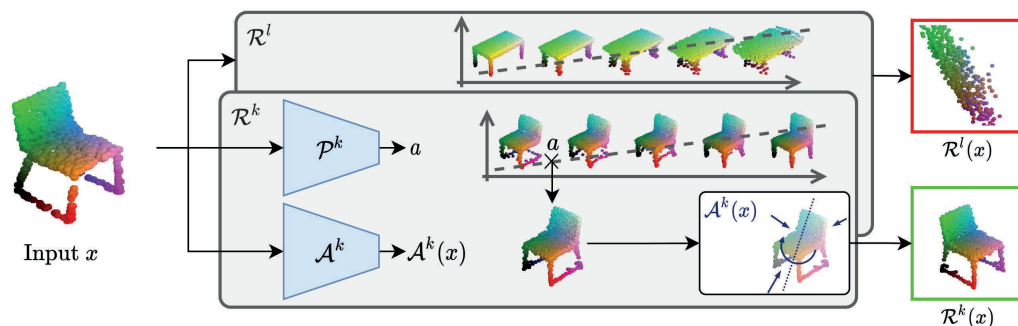


Figure A.8: **Architecture du modèle.** Étant donné un nuage de points d’entrée x , nous prédisons pour chaque modèle de forme \mathcal{R}^k l’élément qui reconstruit le mieux l’entrée : le réseau de projection \mathcal{P}^k produit les coordonnées a d’une forme dans la famille linéaire, et le réseau d’alignement \mathcal{A}^k prédit les paramètres d’une transformation affine $\mathcal{A}^k(x)$ qui est appliquée à la forme sélectionnée. Le nuage de points d’entrée est ensuite assigné au modèle de forme qui le reconstruit le mieux, ici mis en évidence en vert.

Table A.2: **Résultats sur ModelNet10.** Nous présentons les résultats avec 10 modèles de formes linéaires, d’abord pour différentes restrictions des réseaux d’alignement, puis pour différentes configurations de familles linéaires. Les étapes d’entraînement de notre modèle sont en **gras**. Nous rapportons le taux de classification correct en % (“Accuracy”) et la distance de Chamfer multipliée par 10^3 (“CD”). Les résultats sont en moyenne sur cinq exécutions.

| | | Accuracy | CD |
|----------------------|---------------------------------|-------------------|-------------------|
| Ours, proto | | 63.9 ± 1.5 | 20.0 ± 0.4 |
| ... with supervision | | 79.0 ± 0.2 | 23.5 ± 0.0 |
| Ours, align | Rigid transformation (6D) | 64.6 ± 5.2 | 16.2 ± 0.1 |
| | Trans. + Iso. Scaling (4D) | 71.5 ± 4.1 | 15.0 ± 0.1 |
| | Trans. + Aniso. Scaling (6D) | 74.1 ± 3.0 | 10.4 ± 0.1 |
| | Linear (9D) | 71.85 ± 4.7 | 11.1 ± 0.1 |
| | Affine (12D) | 75.9 ± 3.0 | 9.7 ± 0.0 |
| ... with supervision | | 88.9 ± 0.5 | 11.2 ± 0.0 |
| Ours, full | $D=1$ Pointwise parametrization | 74.3 ± 1.7 | 7.9 ± 0.0 |
| | Implicit parametrization | 77.5 ± 2.8 | 8.1 ± 0.0 |
| | ... with supervision | 89.7 ± 0.6 | 9.5 ± 0.0 |
| | $D=5$ Pointwise parametrization | 75.1 ± 1.7 | 5.7 ± 0.0 |
| | Implicit parametrization | 77.0 ± 3.4 | 5.9 ± 0.0 |
| ... with supervision | | 90.4 ± 1.0 | 7.8 ± 0.0 |
| FoldingNet [353] | | 76.3 ± 7.5 | 3.5 ± 0.0 |

Table A.3: **10-shot segmentation.** Nous rapportons l’IoU pour 9 classes et l’IoU moyen sur l’ensemble des 16 classes de ShapeNetPart. Notre modèle produit des résultats à l’état de l’art de la segmentation sémantique avec 10 exemples annotés à l’entraînement.

| | airplane | bag | cap | car | chair | lamp | laptop | mug | table | avg |
|----------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Shared encoder | | | | | | | | | | |
| Gadella <i>et al.</i> 2020 [95] | — | — | — | — | — | — | — | — | — | 74.1 |
| Ours, full $D = 5$ (random) | 71.7 | 70.6 | 84.0 | 62.1 | 78.8 | 68.7 | 93.1 | 87.5 | 70.6 | 72.5 |
| Ours, full $D = 5$ (prototype) | 79.4 | 73.0 | 81.8 | 72.1 | 83.6 | 76.1 | 94.7 | 89.8 | 76.2 | 77.4 |
| One encoder per class | | | | | | | | | | |
| Wang <i>et al.</i> 2020 [332] | 67.3 | 74.4 | 86.3 | — | 83.4 | 68.7 | 93.8 | 90.9 | 74.2 | — |
| Groueix <i>et al.</i> 2019 [116] | 67.1 | — | — | 61.4 | 78.9 | 65.8 | — | — | 66.1 | — |
| Ours, full $D = 5$ (random) | 72.2 | 66.0 | 75.5 | 63.0 | 79.1 | 68.9 | 93.1 | 84.2 | 69.4 | — |
| Ours, full $D = 5$ (prototype) | 80.0 | 79.7 | 76.1 | 72.0 | 83.6 | 77.1 | 94.9 | 91.1 | 75.9 | — |

modèles linéaires et les réseaux neuronaux sont appris de bout en bout en utilisant une seule fonction de coût de reconstruction.

Le principal avantage de notre approche est que, contrairement à de nombreuses approches récentes qui apprennent des représentations de formes complexes basées sur des caractéristiques abstraites, notre modèle est explicite et toutes les opérations se produisent dans l’espace 3D. Par conséquent, nos modèles de formes linéaires peuvent être facilement visualisés et annotés, et les cas d’échec peuvent être compris visuellement.

Bien que notre objectif principal soit d’introduire une représentation compacte et interprétable de collections de formes, nous montrons, en Table A.2 et Table A.3, qu’elle conduit également à des résultats à l’état de l’art du regroupement de collections de formes et de la segmentation sémantique avec très peu d’exemples annotés à l’entraînement.

A.5 Découverte de prototypes 3D dans des scans aériens

Dans cette partie, nous proposons une méthode non supervisée pour analyser de grandes acquisitions 3D de scènes du monde réel en parties interprétables. Notre objectif est de fournir un outil simple, pour analyser des scènes 3D avec

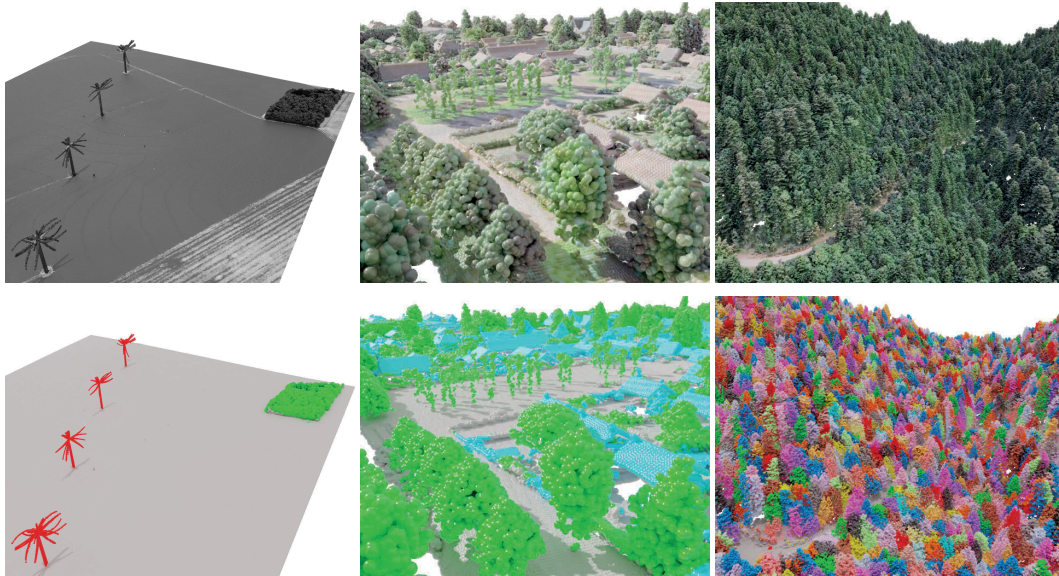


Figure A.9: **Résultats qualitatifs.** Notre méthode non supervisée prend en entrée de grandes numérisations 3D aériennes et les modèle avec un petit ensemble de prototypes appris. Notre approche est entraînée sans annotation et produit une décomposition lisible de scènes complexes, qui peut être utilisée pour la segmentation sémantique et par instance.

des caractéristiques uniques dans le contexte d’acquisitions aérienne et de la cartographie, sans compter sur des annotations spécifiques à l’application de l’utilisateur. Comme montré en Figure A.9, notre modèle produit sans aucune annotation, les segmentations sémantiques et segmentations d’instances de scènes complexes.

Learnable Earth Parser. Comme montré en Figure A.10, notre approche est basée sur un modèle de reconstruction probabiliste, qui décompose un nuage de points 3D en un petit nombre de prototypes appris. Notre modèle fournit une reconstruction interprétable de scènes complexes et produit des segmentations sémantiques et d’instances.

Earth Parser Dataset. Pour démontrer l’utilité de nos résultats, nous présentons un nouveau jeu de données de sept acquisitions par LiDAR aériens. Comme vu en Figure A.11, notre jeu de données est composé de scènes dans des environnements variés, montrant par exemple une centrale électrique, une forêt, ou un port de plaisance.

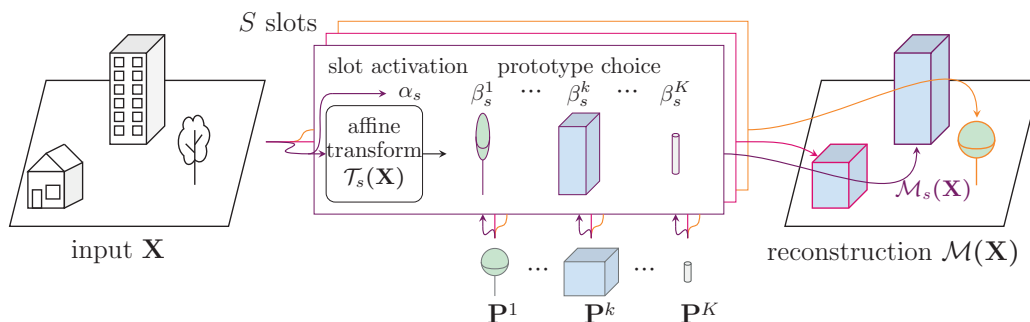


Figure A.10: **Learnable Earth Parser.** Notre modèle approxime un nuage de points d’entrée \mathbf{X} avec S sous-modèles. Chaque sous-modèle associe \mathbf{X} à une déformation affine 3D $\mathcal{T}_s(\mathbf{X})$, une probabilité d’activation α_s , et les probabilités conjointes $\beta_s^1, \dots, \beta_s^K$ du sous-modèle étant activé et choisissant l’un des K nuages de points prototypes $\mathbf{P}^1, \dots, \mathbf{P}^K$. La sortie $\mathcal{M}_s(\mathbf{X})$ d’un sous-modèle activé s est obtenue en appliquant la transformation $\mathcal{T}_s(\mathbf{X})$ à son prototype le plus probable. Les sous-modèles non activés ne contribuent pas à la sortie.

Table A.4: **Résultats sur le Earth Parser Dataset.** Nous rapportons la qualité de la reconstruction (Cham.) et de la segmentation sémantique (mIoU) pour chacune des scènes du Earth Parser Dataset. Bien que notre méthode ne fournisse pas toujours les reconstructions les plus fidèles, elle conduit à la classification de points la plus précise.

| | Rec. | Semantic | Crop fields | | Forest | | Greenhouses | | Marina | | Power plant | | Urban | | Windturbines |
|---------------------|--------|----------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|--------------|
| | | | Cham. | mIoU | Cham. | mIoU | Cham. | mIoU | Cham. | mIoU | Cham. | mIoU | Cham. | mIoU | Cham. |
| k-means (i,z) [205] | ✗ | ✓ | — | 93.8 | — | 71.5 | — | 39.3 | — | 41.4 | — | 42.8 | — | 56.5 | — |
| SuperQuadrics [252] | 3D | ✗ | 0.86 | — | 1.04 | — | 0.60 | — | 0.93 | — | 0.58 | — | 0.40 | — | 13.5 |
| DTI-Sprites [231] | 2.5D+i | ✓ | 6.10 | 83.2 | 14.59 | 40.2 | 5.36 | 42.0 | 6.16 | 41.4 | 5.36 | 29.0 | 2.99 | 47.3 | 36.19 |
| AtlasNet v2 [78] | 3D+i | ✓ | 1.07 | 43.1 | 1.58 | 71.4 | 0.56 | 49.1 | 0.73 | 42.1 | 0.45 | 41.6 | 0.63 | 48.8 | 8.80 |
| Ours | 3D+i | ✓ | 0.72 | 96.9 | 0.88 | 83.7 | 0.40 | 91.3 | 0.82 | 78.7 | 0.44 | 52.2 | 0.29 | 83.2 | 6.65 |

Comme montré en Figure A.4, nous surpassons les autres méthodes non supervisées en termes de précision de la décomposition, tout en restant visuellement interprétable. Notre méthode offre un avantage significatif par rapport aux approches existantes, car elle ne nécessite aucune annotation manuelle, ce qui en fait un outil pratique et efficace pour l’analyse de scènes 3D.

A.6 Conclusion

Nous avons introduit de nouveaux ensembles de données 3D du monde réel ainsi que des méthodes dédiées, et les avons utilisés pour faire avancer l’état de l’art pour diverses tâches.

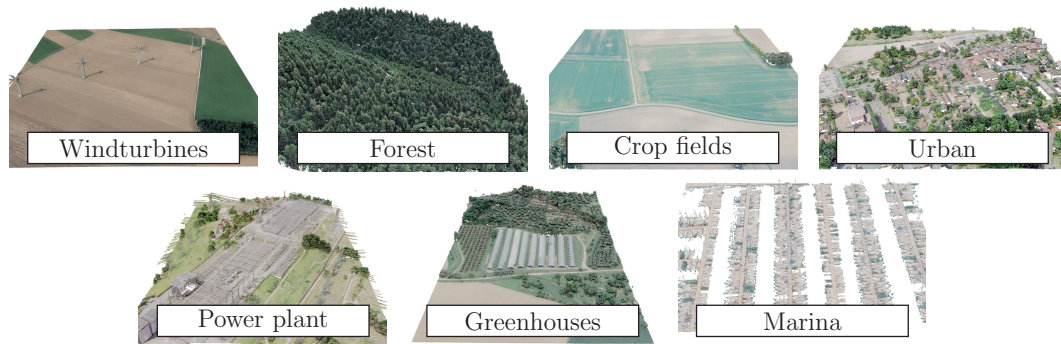


Figure A.11: **Earth Parser Dataset**. Notre jeu de données contient 7 scènes représentant divers environnements urbains et naturels, acquis par LiDAR aérien. Les illustrations de la centrale électrique et des serres montrent les scènes complètes, tandis que les autres montrent un sous-ensemble de chaque scène couvrant entre 25 et 50% de la superficie totale.

Dans le chapitre 3, nous proposons une nouvelle approche qui tire parti de la géométrie du capteur LiDAR. Notre méthode de segmentation sémantique en temps réel repose sur un mécanisme d’attention spatio-temporel efficace, et surpasse les méthodes existantes en termes de latence, tout en préservant les performances de segmentation de l’état de l’art. Nous introduisons également un ensemble de données annotées public spécifiquement conçu pour l’évaluation et la conception de méthodes de segmentation en temps réel. Nous espérons que ce travail ouvrira la voie à des méthodes exploitant les spécificités des données, pour concevoir des méthodes efficaces et précises.

Dans le chapitre 4, nous présentons une nouvelle approche des modèles linéaires de formes avec l’apprentissage profond. Notre modèle produit des aperçus concis, expressifs et interprétables de collections de nuages de points non alignés. Nous montrons que notre méthode conduit à des résultats à l’état de l’art de la segmentation avec peu d’exemples annotés. Nous espérons que ce travail permettra de meilleures représentations non supervisées de vastes collections de formes 3D non structurées.

Dans le chapitre 5, nous étendons le cadre d’apprentissage de prototypes 3D introduit dans le chapitre 4, pour découper des scans aériens complexes du monde réel en parties simples à l’aide d’un petit ensemble de formes prototypiques apprises. Nous introduisons également un ensemble de données LiDAR aérien public acquis dans différents environnements tels que les forêts, les zones

urbaines et les marinas. Nous montrons la possibilité d'effectuer une analyse de formes 3D profonde non supervisée sur un ensemble de données du monde réel. Nous pensons que nos résultats ouvrent de nouvelles perspectives pour la gestion de l'environnement assistée par ordinateur, et pour l'intelligence économique.

Les deux principales leçons tirées de cette thèse sont : (i) la géométrie du capteur 3D peut être exploitée pour mieux représenter les données, ce qui permet de gagner en efficacité ; et (ii) l'utilisation de connaissances préalables sur les données peut aider à concevoir des méthodes non supervisées puissantes et pertinentes. Les travaux précédents utilisent souvent des données 3D sans s'adapter aux spécificités du capteur, ou nécessitent des quantités massives de données annotées pour être entraînés. Au contraire, nous avons conçu des modèles efficaces spécifiquement adaptés aux données 3D. Comme le démontrent nos expérimentations, cela permet d'obtenir des modèles beaucoup plus efficaces sans perdre en précision.

