3D Hybrid Urban Scene Semantic Mapping From Multi-modal Data

Mohamed Boussaha

LaSTIG – ACTE,
Institut National de l'Information
Géographique et Forestière (IGN)

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy delivered by:

Université Paris-Est – MSTIC (ED532)

Specialty: Geographical Information Sciences and technologies

3D Hybrid Urban Scene Semantic Mapping from Multi-modal Data

Mohamed BOUSSAHA

Defended *June 5, 2020* at
IGN, Saint-Mandé, France

Jury members

David Filliat	ENSTA Paris	Reviewer
Florent Lafarge	Titane, INRIA	Reviewer
Vincent Lepetit	Imagine, ENPC ParisTech	Examiner
Véronique Berge-Cherfaoui	Heudiasyc, UTC	Examiner
Jean-Emmanuel Deschaud	Mines ParisTech	Examiner
Bruno Vallet	Acte-LASTIG, IGN	Advisor
Patrick Rives	Lagadig, INRIA	Co-adviser
Loïc Landrieu	Strudel-LASTIG, IGN	Supervisor

This thesis is dedicated to my beloved family.

Acknowledgements

No one can make it from naive excitement to completed dissertation without a constant assistance, both technical and spiritual. I consider myself lucky to have the support of a committee with a wide range of skills, tastes, perspectives and experience.

First and foremost, I would like to express my sincere gratitude to both my advisor and co-advisor Bruno and Patrick for not only giving me the chance to join this P.h.D program but also for providing me with the necessary technical and scientific help all along the way. A special thanks goes as well to my supervisor Loïc for his unequivocal support.

I am indebted to the ANR (Agence Nationale de la Recherche), the public french organization which funded my P.h.D. Without its support the pLaTINUM project could not have reached its goal. My deepest gratitude is also for the french mapping agency IGN for considering me as one of its staff members till the end.

I would also like to extend my thanks to the pLaTINUM project members : Pascal Vasseur, Cédric Demonceaux, Valérie Gouet-Brunet, Sébastien Kramm, Désiré Sidibé, Eduardo Fernandez-Moral, Imeen Ben Salah and Nathan Piasco, from whom I have learned a great deal about the recent advances in the field of computer vision and robotics during the regular project meetings.

There are lots of others who stepped in at one critical moment or another. Amine for his exceptionally valuable suggestions, Quoc for helping me annotating a part of my dataset, Oussama, Stéphane and Nathan for offering insightful and funny discussions as well as endless encouragements.

Last but not least, None of this could have happened without the support of my family: my parents and brothers who encouraged me from the beginning to take this path, my wife who endured this long process with me, offering always unconditional support.

Abstract

With the democratization of collaborative navigation applications and autonomous robots, mobile mapping has received an increasing attention in recent years, both in academic and industrial circles. The digitization of the environment not only provides detailed and extensive knowledge enabling end-users to anticipate and plan their journeys, but also guarantees the availability of reliable up-to-date information in critical scenarios, for instance, when sensors of an autonomous car fail to perceive the surroundings. Mobile mapping raises, however, many challenges in terms of robustness, accuracy and scalability. Processing mapping data requires methods capable of handling massive data, with centimetric accuracy while coping with the acquisition specificities such as variability in levels of detail, occlusions, and strong variations in lighting conditions.

In the context of the French ANR project pLaTINUM, this thesis focuses on the development of a global geolocalised map of urban environments comprised of 3D representations based on geometric, photometric and semantic information. Firstly, a comparative investigation of suitable geometric representation options yields to the reconstruction of a large scale, high definition map with a textured 3D mesh. This representation is the result of a multi-modal fusion of oriented images and geo-referenced LiDAR scans acquired by a terrestrial mobile mapping platform. Subsequently, we propose to infer high level semantics to the reconstructed map by exploiting the complementarity between the two acquisition modalities: photometry and geometry. Throughout the rich literature regarding this subject, we have identified a need of an annotated multi-modal urban dataset comprising a large scale textured mesh. This has led us to produce our own dataset composed of 3D point clouds, 2D geolocalized panoramic and perspective images, depth and reflectance maps, and a 3D textured mesh with the corresponding ground truth annotations for each modality.

Secondly, we assume that the global map is represented by means of 3D point cloud structured by an adjacency graph. We introduce a novel supervised over-segmentation approach. This method operates in two steps: (i) local descriptors of 3D points are computed via deep metric learning, (ii) the point cloud is partitioned into uniform clusters called superpoints. The descriptors are learned such that they present high contrast at the interface between objects, thereby encouraging the partition to follow their natural contours. Our experiments on indoor and outdoor scenes show the clear superiority of our approach over state-of-the-art point cloud partitioning methods. We further illustrate how our method can be combined with a superpoint-based classification algorithm to enhance the performance of semantic segmentation of 3D point clouds, also improving the state-of-the-art in this field.

Finally, we extend this approach to textured meshes. Triangles, structured this time by the dual graph of the mesh, are partitioned into homogeneous groups called superfacets. Much like point clouds, local descriptors of the textured mesh are learned so that the boundaries of the objects exhibit high contrast. These descriptors are the result of merging descriptors learned from the convolution of the mesh edges on the one hand, and the texture descriptors extracted from the 2D image domain on the other. The experi-

ments conducted on our own multi-modal dataset, show the superiority of our approach compared to state-of-the-art methods for the task of 3D mesh over-segmentation.

Résumé

Avec la démocratisation des applications collaboratives d'assistance à la navigation et l'avènement de robots autonomes, la cartographie mobile suscite ces dernières années une attention croissante, tant dans les milieux académiques qu'industriels. La numérisation de l'environnement offre non seulement une connaissance fine et exhaustive permettant aux usagers d'anticiper et de planifier leurs déplacements, mais garantit aussi la disponibilité d'informations fiables à tout instant notamment en cas d'éventuelle défaillance des capteurs visuels d'un véhicule autonome. S'agissant d'un enjeu crucial pour une navigation fiable, la cartographie mobile soulève en revanche de nombreux défis en matière de robustesse, de précision et de passage à l'échelle. Cette problématique fait appel à des méthodes qui requièrent une capacité de traitement de données massives avec une précision centimétrique tout en gérant les spécificités de l'acquisition, incluant la variabilité du niveau de détails, des occultations et des fortes variations de luminosité.

Dans le cadre du projet ANR pLaTINUM, cette thèse porte sur le développement d'un référentiel global géolocalisé de l'environnement urbain constitué de représentations 3D basées sur des informations géométriques, photométriques et sémantiques. Dans un premier temps, une investigation approfondie de la représentation géométrique la plus adaptée à un tel référentiel, permet une reconstruction d'une carte haute définition à large échelle sous forme d'un maillage 3D texturé. Cette représentation est mise en place par fusion multimodale d'images orientées et de balayages LiDAR géo-référencés acquis depuis une plate-forme de cartographie mobile terrestre. Par la suite, nous proposons d'intégrer l'aspect sémantique au référentiel reconstruit, en exploitant la complémentarité entre les modalités d'acquisition photométriques et géométriques. À travers la riche littérature sur le sujet, nous identifions l'absence d'un jeu de données urbain multimodal annoté incluant un maillage texturé à large échelle. Nous levons ce verrou par la production de notre propre jeu de données composé de nuages de point 3D, d'images 2D perspectives et panoramiques géolocalisées, de cartes de profondeur et de reflectance ainsi qu'un maillage texturé avec les annotations correspondantes à chaque modalité.

Dans un second temps, nous considérons le référentiel comme un nuage de points 3D structuré par un graphe d'adjacence. Nous introduisons une nouvelle approche de sursegmentation par apprentissage supervisé. Cette méthode opère en deux temps: calcul de descripteurs locaux des points 3D par apprentissage profond de métrique, puis partition du nuage de points en zones uniformes, appelées superpoints. Les descripteurs sont appris de telle sorte qu'ils présentent de forts contrastes à l'interface entre objets, incitant la partition résultante à suive leurs contours naturels. Nos expériences sur des scènes d'intérieurs et d'extérieurs montrent la nette supériorité de notre approche sur les méthodes de partition de nuage de points de l'état de l'art, qui ne reposaient pas jusqu'à là sur l'apprentissage machine. Nous montrons également que notre méthode peut être combinée à un algorithme de classification de superpoints pour obtenir d'excellents résultats en terme de segmentation sémantique, améliorant aussi l'état de l'art sur cette problématique.

Enfin, nous étendons cette approche aux maillages texturés. Les triangles, structurés cette fois-ci par le graphe d'adjacence du maillage, sont partitionnés en groupes homogènes appelés superfacettes. À l'instar des nuages de points, des descripteurs locaux du maillage

texturé sont appris de façon à ce que les frontières d'objets sémantiquement distincts présentent un contraste élevé. Ces descripteurs sont le résultat d'une fusion des descripteurs appris sur le maillage par convolution des arêtes d'une part, et des descripteurs de texture d'autre part. Les expériences réalisées sur notre propre jeu de données multimodal illustrent la supériorité de notre approche par rapport aux méthodes de l'état de l'art pour la sur-segmentation d'un maillage.

Contents

List of Figures

List of Tables

Chapter

1 General introduction

Chapter content

Introduction

A few decades ago, the simple idea that driver-less cars exist or machines surpassing humans at common daily tasks, would have been met with incredulity. Nowadays, this utopia has become a reality. Machines are now able to beat humans at an entire raft of complex tasks ranging from notoriously hard strategy games [1], data science [1], writing pop songs [2], creating original art pieces [3] or even at spotting issues in legal contracts [4]. These achievements have created an unprecedented enthusiasm for applications based on artificial intelligence (AI).

Autonomous navigation of aerial and terrestrial robots (*e.g.* cars, drones, etc.), in particular, is not immune to this increasing AI-based trend. Currently, we are arguably witnessing the heyday of the self-driving revolution. Research and development corporations all over the world are racing towards developing autonomous agents capable of

[1] https://deepmind.com/research/case-studies/alphago-the-story-so-far
[2] http://www.bbc.com/culture/story/20180112-is-this-the-worlds-first-good-robot-album
[3] https://deepart.io/
[4] https://www.lawgeex.com/resources/aivslawyer/

providing reliable navigation services. This competition resulted in an arsenal of products deployed in our daily life such as Tesla, Lyft and Uber self-driving cars among others.

In order to be able to plan a path and navigate towards a specific target, an autonomous agent has to apprehend its surroundings as would humans do. In an unknown environment, humans rely on a *map* to reach a desired destination. In a similar way, a robot has to be endowed with a map to accomplish a certain navigation or planning task. Constructing such a map entails two crucial issues. The first one consists in the availability of sufficiently rich and diverse collected data along with an appropriate exploitation. The second is about the ability of the machine to interpret and *semantically* understand these data. The former problem is more related to close range remote sensing or *mapping*, while the latter falls under the topic of *scene understanding*. These two challenges will be kept in the foreground of this dissertation.

By taking a closer look at the rich literature, we notice that these two research axes are intertwined. As a matter of fact, in the context of autonomous navigation, a substantial improvement in scene understanding has been accomplished in the recent years thanks to the ubiquity and diversity of large volumes of mapping data. This success can be mainly explained by the surge of data-driven paradigms over hand-crafted ones notably *deep learning*.

Deep learning techniques differ from the rest of machine learning approaches in that descriptors are learnt as part of the training process. Commonly referred to as learning representations, no one can deny the wave of attention this concept brought to Convolutional Neural Networks (CNNs). This surge started when AlexNet [2] outperformed competing hand-crafted methods by a significant margin in several recognition tasks on 2D images. Three-dimensional data *e.g.* point clouds or meshes were also affected by this shift in focus towards deep learning approaches. However, unlike 2D images , learning representations from 3D data is a strenuous challenge [3]. Point clouds and 3D mesh geometries in particular can not be trivially defined as functions on the Euclidean space sampled on a grid. The lack of a grid-like structure inhibits the use of one of the CNNs fundamental core components which is convolution. Furthermore, from the perspective of autonomous driving, massive amounts of multi-modal data have to be acquired to guarantee a reliable level of robustness. According to a Rand Corporation report [5], a fleet of 1000 vehicles driving nonstop for 50 years is required to gather sufficient real testing data to conclude a 20% advantage for autonomous vehicles over human drivers. Therefore, specific methods have to be designed with the goal of constructing a precise map on one hand, and efficiently handle the consequent large volume of hybrid data on the other.

The purpose of this thesis is twofold. First, we want to explore new ways on how to reconstruct a 3D map of an urban environment from multi-modal mapping data. Second, we want to make use of the recent advances in scene understanding to learn new representations from hybrid 3D data. Throughout this dissertation, we will try to answer the following questions:

- What are the main characteristics of a reliable 3D map ?
- How can we reconstruct such map given a multi-modal mapping acquisition ?
- What is the best strategy to fuse these hybrid data ?

[5] https://www.rand.org/pubs/research_reports/RR1478.html

- How can we learn expressive representations of multi-modal 3D data while remaining efficiently able to handle their large scale ?

The remainder of this chapter is organized as the following. First in Section 1.1, we present the general context of this thesis. In section 1.3 we clearly state the objectives and challenges of our study and demonstrate how it is a valid scientific problem. Section 1.2 clarifies the main research axes. Finally the thesis outline and contributions are summarized in Section 1.4.

1.1 General context

A Geographic Information System (GIS) is a general framework for managing, analyzing and visualizing geographic spatial data. GISs represent a strategic clue for many civil and military applications. Over the past 15 years, the global GIS market has experienced a rapid growth, rising from less than 1 billion USD in 2000 to more than 10 billion USD in 2015, of which around 10% is allocated to 3D applications. Consequently, this market is becoming extremely competitive; resulting in a wide variety of geo-referenced databases. At the moment, the most well-known and probably the most used database is developed by Google, through its products Google Earth, Google Maps and Google StreetView. OpenStreetMap [6] is also a salient example of open source GIS as it allows individuals to contribute with additional content such as videos, images, or GPS tracks. At the national level, the french mapping agency (IGN) provides topographic maps on a national scale via the *Geoportail* [7]. The latter has been recently upgraded to allow the access to 3D urban models after integrating the interactive web 3D viewer *iTowns*[8]. Among the many existing GISs, few are those containing 3D representations (models of objects/ buildings or digital elevation models DEMs) along with their corresponding semantic information. In addition to their content, these GISs differ mainly by their coverage (local, global), their resolution (centimeters to kilometers), access to their data (free, paid, internet access, etc.), their update frequency, and above all their use cases.

The current usage of these geo-referenced systems is rather focused on the analysis of images or maps of a given place for applications ranging from simple virtual visit to the extraction of information related to various professions (agriculture, natural disaster management, etc.). However, they are not well suited for autonomous navigation applications mainly because they were not designed for it. The provided content, the data access procedure, and the memory volume which they occupy are not adapted for this specific use case. Indeed, when it comes to comparing the video stream of an on-board camera with an a priori digital model of the environment for example, the volume of data and processing becomes quickly considerable. As a result, the numerous works on mobility and autonomous navigation rarely use the existing 3D geo-localized databases, but rather ad-hoc data acquired for the occasion, or derived from the aforementioned GISs. This confirms the need for dedicated representations of the environment, different from those used traditionally.

In this context, the LAGADIC team at INRIA-Sophia Antipolis, the Le2i laboratory

[6] https://www.openstreetmap.org
[7] https://www.geoportail.gouv.fr/
[8] http://www.itowns-project.org/

from the university of Burgundy, INSA-Rouen university and the french mapping agency (IGN), have joined in an ANR project coined as *pLaTINUM* (Long Term MappINg for Urban Mobility) whose purpose, broadly speaking, is to provide navigation services for remote agents. This Ph.D. thesis was funded by this project and contributes to it. Hereafter, we present the *pLaTINUM* project in details.

1.1.1 *pLaTINUM* project

This thesis is part of the inter-disciplinary ANR project *pLaTINUM*. The goal of this project consists in developing a geographical information system (GIS), stored on a cloud and made of a set of 3D representations based on geometric, photometric, and semantic information. This system must be able to update and enrich its content automatically based on data transmitted by remote agents, which can be either autonomous vehicles or simply users of an application (*e.g.* augmented or mixed reality) requiring more precise localization and orientation than what GPS allows. Through this collaborative application, an agent is supposed to receive localization information during navigation. In turn, the latter will inform the cloud about the local detected changes, which will be processed in order to update the geo-localized map.

A typical functional scenario can be envisioned as follows: an agent arriving in an area covered by the cloud, issues a navigation request. The latter can be an approximate GPS position and/or one or more images of the surroundings or a semantic request (*e.g.* next to a park). In return, the cloud will send the local maps of the potential candidate places corresponding to the area around the agent. The received information will be used to precisely locate and register the underlying data with that of the selected map. The same agent is also expected to participate in updating the cloud. The latter will validate or not the new data based on the previously available information.

1.1.2 *pLaTINUM* expectations

As illustrated in the functional diagram of the project in Figure 1.1, the project partners are expecting a global geo-localized 3D map encompassing geometric, photometric, and semantic information which will serve for later applications (localization, navigation, update, etc.). More in details, we were assigned the set of work packages framed in red. First, in the work package 4.1 we have to collect multi-modal data using a mobile mapping system. The latter allows a joint acquisition of oriented RGB images and geo-referenced LiDAR scans. This data will be mutually used by our ourselves in this thesis and by the project partners for validating their localization and navigation algorithms. Second, we are supposed to design and deliver an appropriate representation of the acquired data in the form of a 3D map that is compatible with the project requirements. The goal of the work package 1.1 is to ensure the consistency between the image data and the LiDAR scans, in particular by determining the visibility of each LiDAR point in the 2D image. Visibility computation is an essential prerequisite for the joint use of LiDAR data with their metro-logical precision and 2D images with the richness of their radiometric information. Ideally, this map has to be a three-dimensional hybrid representation of the acquired environment combining both the geometric information acquired from the LiDAR and the radiometric information captured by the cameras.

Work package 1.2 concerns scene understanding. In this step, we want to enrich the

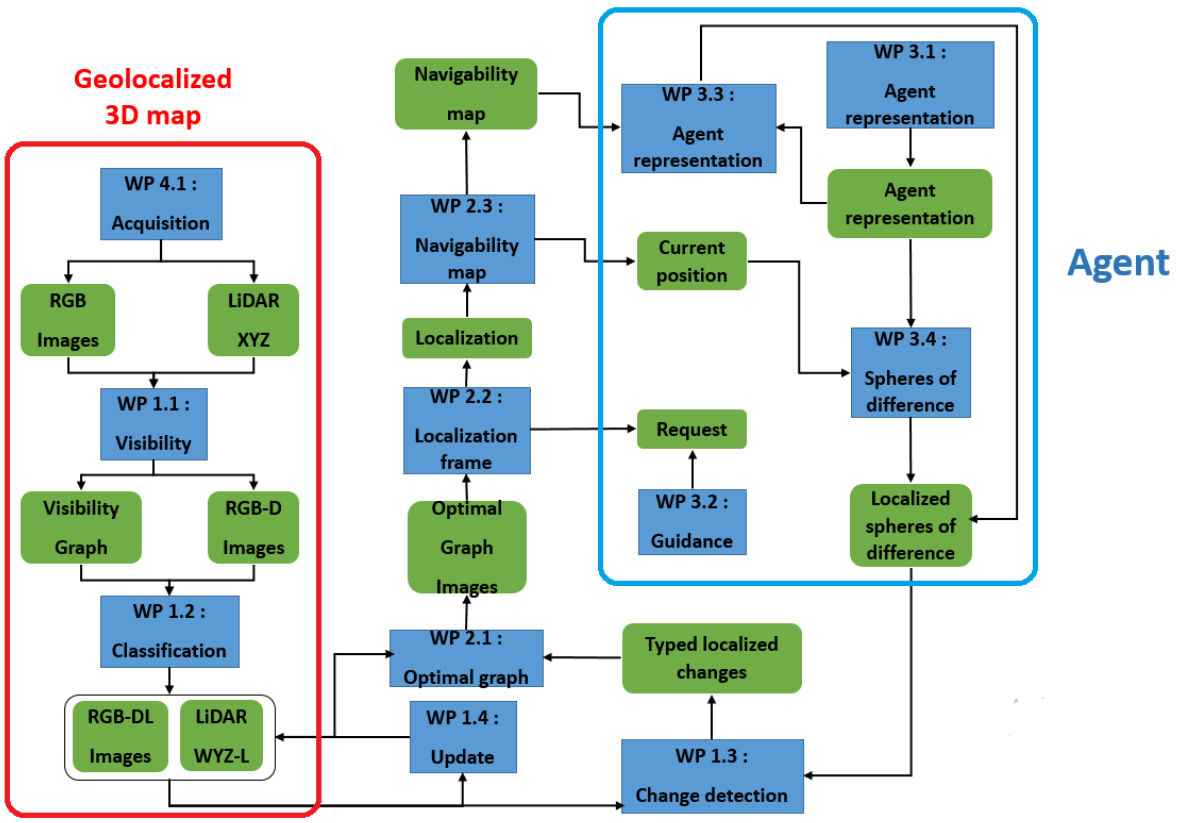


Figure 1.1: *pLaTINUM project functional diagram. Our thesis lies within the work packages framed in red.*

3D reconstructed map from WP 1.1 with semantics of objects constituting the observed scene while being able to handle its large scale. The expected qualities of this 3D map are mainly precise geo-referencing, resolution and semantic fidelity. The reconstructed 3D map should enable highly precise geo-referencing of the collected data more than what a GPS allows. Resolution determines what is the size of the smallest object/detail that should be represented in the map. Fidelity in this context means that the scanned digital objects in the scene and their real counterparts must have the same semantic label. This allows the decomposition of the scanned space into semantically navigable zones (*e.g.* sidewalks for pedestrians, roads for vehicles, etc.) along with traffic signs. In the following section, we overview the scientific problems related to our thesis subject and analyze the consequent challenges to be tackled in our work.

1.2 Research axes

As can be seen throughout the aforedescribed work packages, our thesis is centered around two principle research axes - namely mobile mapping and scene understanding. In the following sections, we detail the background related to these two aspects.

1.2.1 Mobile mapping

Mapping is the process of acquiring and managing geo-referenced spatial data of a given environment. Historically mapping was performed using manual surveying which often requires several months if not years to cover a relatively large area. With the recent technological advances, mapping can be performed using a multi-sensor system mounted on various platforms such as satellites, drones, water-based vessels among others. Terrestrial mobile mapping systems in particular make use of land-based vehicles as platforms integrating different types of sensors to collect geo-referenced data. This *close-range* mapping system takes advantage of the proximity to the observed scene enabling, thereby, a much finer and precise acquisition.

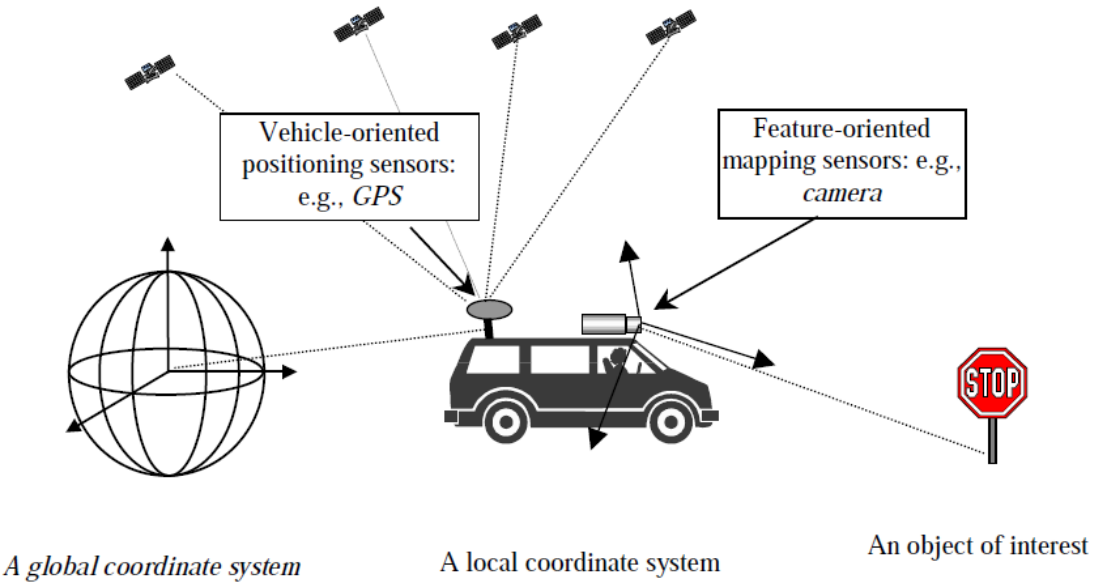


Figure 1.2: *Illustration of a terrestrial mobile mapping system. Image from [4]*

A mobile mapping system relies on *vehicle-oriented sensors* and *object/feature-oriented sensors* to compute geo-referenced data. Commonly referred to as *direct geo-referencing*, this procedure comes as an alternative to the traditional expensive methods of data geo-referencing. Instead of relying on ground control points (GCP) and the triangulation photogrammetric block, direct geo-referencing [5] uses a combination of GPS and IMU to compute the exterior orientation (position and orientation) of sensors. This concept is illustrated in Figure 1.2.

1.2.1.1 Components

A mobile mapping system is generally comprised of positioning sensors (*e.g.* Global Positioning System (GPS), Inertial Navigation System (INS)) and mapping sensors (*e.g.* LiDAR, camera, etc.). In this section we present the components of a terrestrial mobile mapping vehicle. A particular focus will be devoted to mapping sensors as the acquired data by these means will be the center of interest of our study.

- **Positioning sensors:** These sensors are used to compute the instantaneous absolute locations of the mobile mapping vehicle in a global coordinate system (*e.g.* Lambert 93)

while moving in a certain area. A combination of a GPS and an inertial measurement unit (IMU) is commonly used to determine the pose of the platform in a global frame. GPS records (X, Y, Z) coordinates while IMU records the orientation represented by the three angles (Pitch, Yaw and Roll). The joint use of GPS and IMU allows a more accurate geo-referencing procedure. When GPS tracks are not available or masked due to signal occlusion, IMU measurements serve for correcting the geo-referencing. Alternatively, IMU drifts accumulated during acquisition time are controlled by the GPS updates.

- **Mapping sensors:** are trusted for providing positional information of the scanned objects/features with respect to a local coordinate system relative to the mobile platform. Mapping sensors can be broadly divided into tow groups; passive imaging sensors (*e.g.* RGB camera) and active imaging sensor (*e.g.* LiDAR). Active sensors are different from passive sensors by their capability to provide their own source of lighting or illumination. In the following, we focus on LiDAR and camera since they are the main mapping sensors used in our study.

  Much like the human eye, RGB cameras capture wavelengths between $400nm$ to $780nm$. The acquired data using this device are represented as 2D grids with three separate channels: R, G and B, corresponding to colors in the visible spectrum. These devices are the most commonly used in autonomous navigation perception systems to extract information about the surroundings of the vehicle due to their low cost, high quality color information, and high resolution. The quality of RGB data is highly influenced by variations in illumination and weather conditions (*e.g.* rain, fog, snow etc.). Therefore, these devices are usually combined with LiDAR sensors to increase its robustness. Other types of cameras are also leveraged for mobile mapping acquisition such as near-infrared cameras (NIR) (wavelengths between 780 nm to 3 μm) or mid-infrared (MID) (wavelengths between 3 μm to 50 μm) known as thermal cameras. The use of IR cameras complements RGB cameras in particular scenario where lighting conditions are highly affected (*e.g.* driving with a sun glare) or for pedestrian detection in nighttime.

  A Light Detection And Ranging (LiDAR) is an active sensor based on the time-of-flight technology. This sensor emits laser beam towards a certain target and measures the return time of the reflected pulses to determine the distance to the scanned target. The intensities of the reflected pulses can also be measured to produce a texture-like information, referred to as *intensity*. This value quantifies how much of the emitted laser beam is back-scattered on the receiver by the hit surface. The emitted laser belongs to the *Class 1* which means it is harmless under all conditions of normal use. Using an active sensor such as LiDAR, offers unmatched advantages especially a direct access to depth information without using any complex algorithms. In contrast, similar to cameras, LiDAR is affected by weather conditions (*e.g.* rain, snow, etc.) and dusty environments because of light diffraction in these circumstances. Moreover, these devices suffer from their limited range, typically between 50 to $100m$.

1.2.1.2 Challenges

Processing mobile mapping data is a challenging task for several reasons. First, the collected data have to be geo-referenced with higher precision than GPS devices. As discussed earlier, GPS signals are frequently masked in urban environments because of occlusions. While GPS errors can be corrected by IMU measurements, sometimes this is not sufficient.

Higher precision can be achieved either by an auto-consistency check (*i.e.* registering the collected data on itself by mapping the same street multiple times for instance) or by using an additional external geo-localized source of data (*e.g.* geo-referenced landmarks, ground control points, or precise aerial images). Second, mobile mapping acquisitions result in large volumes of data. The scale of the collected data calls for efficient processing tools in terms of number of operations and memory footprint. Furthermore, as the acquisition of mapping data is commonly conducted in dynamic environments, we need to handle mobile objects (*e.g.* pedestrians, cars, etc.) and temporarily variant objects such as vegetation, roadworks, and urban furniture, etc.

Finally, the heterogeneity of the acquired data (geo-referenced point clouds and oriented RGB images) makes their fusion not straightforward. In Figure 1.3, we show a superimposition of LiDAR and camera modalities to illustrate their intrinsic geometric and radiometric structures. Usually mobile mapping data are used for building a map of the scanned environment which will serve for a wide range of applications. However it is not clear how such multi-modal data can be efficiently merged to accomplish the intended task. Moreover, an optimal exploitation of these two sources of information would require a perfect registration which is not possible most of the time. As a matter of fact, the moving platform can be subject to vibrations due to road slops or speed bumps inducing non-negligible registration errors.

1.2.1.3 Applications

The recent development of mobile mapping systems is driven by the need for large scale urban data mainly for intelligent transportation applications and geographical information systems. Many applications benefit from MMS data. Infrastructure mapping, in particular is among the most common applications of MMS data as it involves several attractive domains such as autonomous driving, 3D city modeling and urban monitoring.

- **Urban modeling and monitoring:** In the field of architecture, the availability of mobile mapping data facilitates urban modeling. Thanks to MMS data density, the extraction and reconstruction of 3D buildings become much easier. Mapping data provide a much higher level of detail on building façades, but needs to be complemented by aerial data for roof modeling. In recent years, we become able to generate building models with different level of details (LoD1, LoD2 and LoD3) at the scale of an entire city. Moreover, these data allow the development of automatic tools for monitoring permanent urban structures, inventory of road signs and urban furniture as well as detecting potential urban changes.

- **Autonomous driving:** Geographical information systems (GISs) rely on mapping data to save and update a digital version of the scanned spatial environments. Autonomous driving is a critical application for which human safety is at stake. In these applications, a GIS provides a crucial information serving as backup when perception or communication system fails. The objects of interest in these off-line maps are permanent structures such as roads, pavements, buildings, etc. This application takes advantage of the multi-modal aspect that mobile mapping acquisitions provide along with the precise geo-referencing of the acquired data. The featured tasks are mainly high level recognition problems such as object detection, semantic and instance segmentation. The latter aspects are related to a larger research axe which is scene understanding.

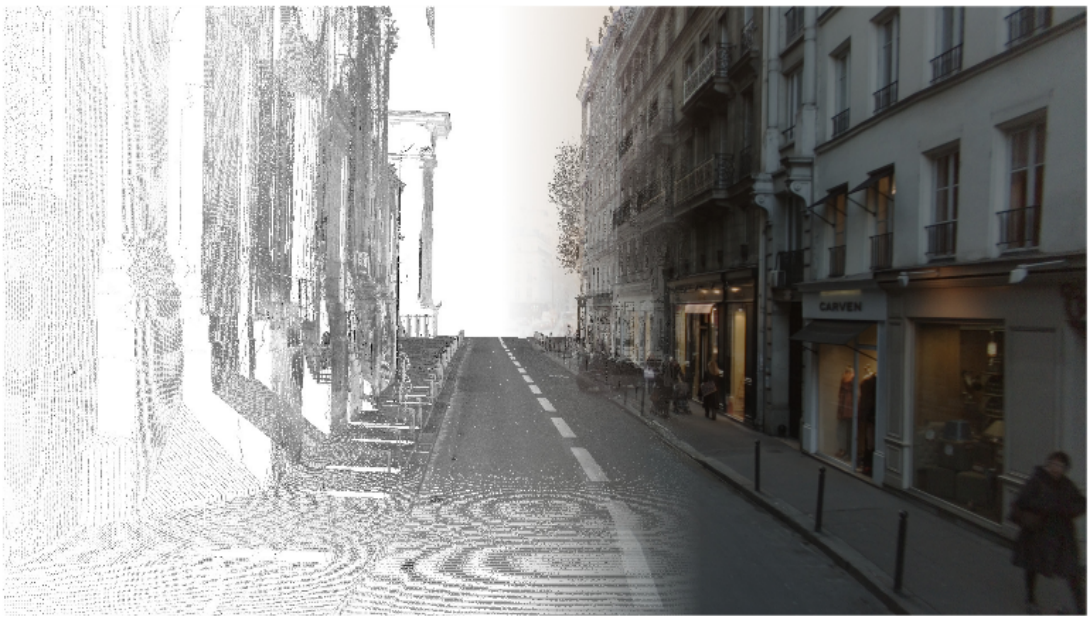


Figure 1.3: *Superimposed LiDAR 3D point cloud with a 2D image.*

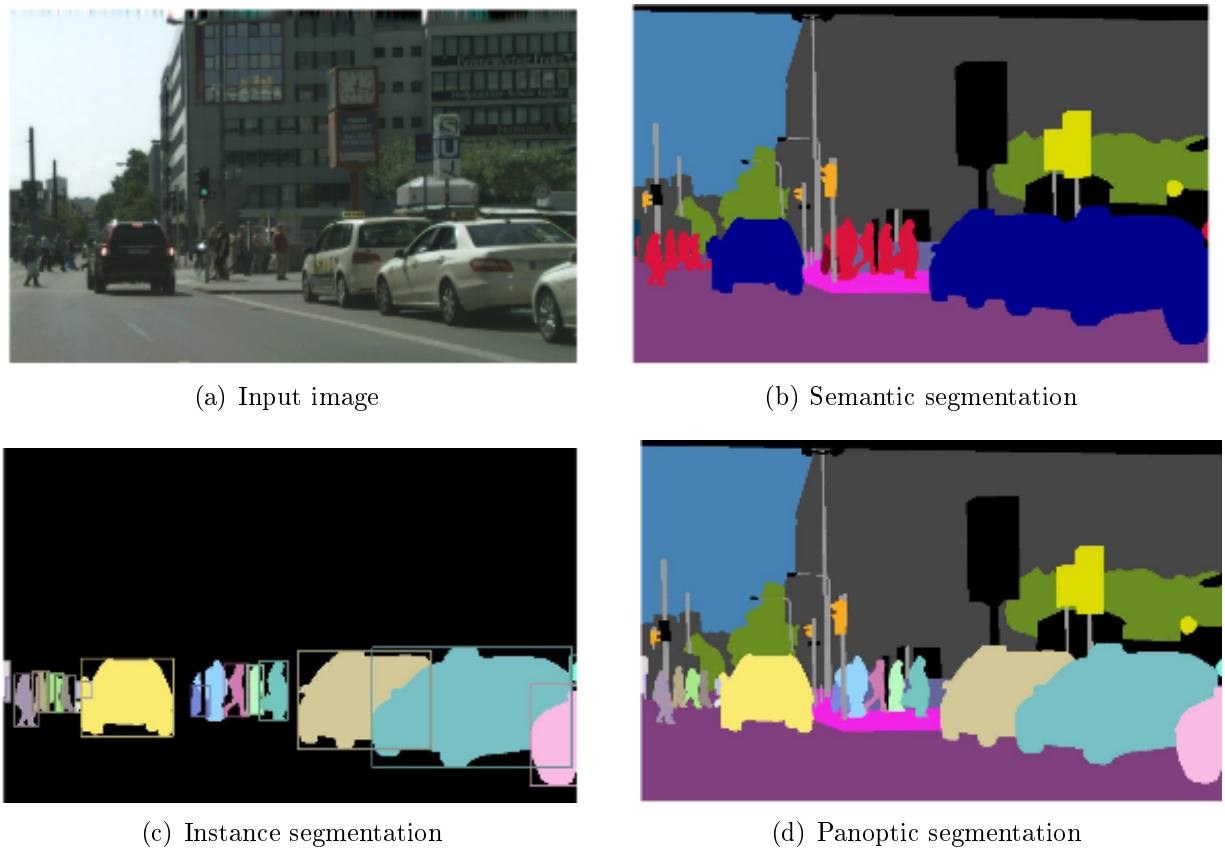
1.2.2 3D scene understanding

Scene understanding is a process by which a machine analyzes and interprets an observed scene using one or more sensors. This process requires a high level abstraction of the sensed scene which inevitably calls for relevant information extraction and filtering. This field includes, but is not limited to, segmentation and semantic segmentation. The first building block for scene understanding called segmentation consists in partitioning the scene into coherent homogeneous parts without associating them with a semantic label. Semantic segmentation extends the raw segmentation by associating each data point (*e.g.* pixel for images, point for 3D point clouds) with a label among a set of defined semantic classes. A more complicated task is instance segmentation which consists in detecting and segmenting each object instance for all the foreground objects in the scene. More recently panoptic segmentation was introduced by Kirillov *et al.* in [6]. The latter task combines semantic segmentation and instance segmentation at the same time. The end goal of this task is to identify two categories; *things* and *stuff*. Things refer to the set of countable objects *e.g.* cars, pedestrians, furniture, etc., while Stuff designate the set of uncountable objects *e.g.* road, pavement, etc. Figure 1.4 illustrates the discussed three modes of scene understanding applied to 2D images.

In the scope of *pLaTINUM* project, we are interested in a particular aspect of semantic mapping which is semantic segmentation of 3D data. The tremendous success of deep learning approaches in a wide range of 2D scene understanding tasks has motivated the community to investigate the applicability of these learning-based frameworks to three-dimensional data. This however poses important methodological challenges as the nature of the data to process is different.

1.2.2.1 Challenges

Driven by the recent advances in deep learning applied to 2D images, a considerable effort has been devoted to translating and/or adapting these 2D architectures to 3D data



(a) Input image

(b) Semantic segmentation

(c) Instance segmentation

(d) Panoptic segmentation

Figure 1.4: *Illustration of semantic, instance and panoptic segmentation on a 2D image. Images from [6]*

notably point clouds and 3D meshes. These 3D data structures are intrinsically different from classic 2D images which can be defined as functions on the Euclidean space sampled on 2D grids. The absence of a grid-like structure makes the use of convolutional networks extremely challenging. Moreover, the geometric properties of 3D data can be represented in various forms depending on the use case. This diversity inhibits the development of an unified architecture that is able to fit to the desired applications. Last but not least, a major problem of 3D data is mainly related to its large scale. Autonomous navigation applications in particular require processing large volumes of data. Before the emergence of deep learning approaches, probabilistic graphical models were the standard in 3D mapping data classification due to their ability to leverage the contextual information efficiently [7]. Although being highly optimized, these graphical models are known to be local methods in the sense that the model complexity grows with the number of graph nodes and edges encoding respectively data points and their pairwise interactions. Applying these techniques on mobile mapping data with millions if not billions of 3D points/triangles is not feasible. Finally, unlike 2D images for which providing a ground truth at large scale is relatively easy, annotating 3D data is an extremely expensive and overwhelming task.

1.3 Thesis positioning

With the recent advances in artificial intelligence applications and the proliferation of acquisition devices, we have seen the dawning of a new era of autonomous robots. The ubiquity and diversity of mapping data have opened the door to several research questions notably regarding the optimal exploitation of different data modalities for safer and more efficient navigation. The latter requires reconstructing a 3D map of the surrounding environment such that an autonomous agent is capable of localizing itself and plan a path to a certain destination. In the literature, a 3D map is commonly represented as sparse landmarks [8], 3D point clouds [9], occupancy grids or voxels [10], 2.5D elevation [11] or 3D mesh [12]. These maps are often reconstructed based on geometric keys while discarding radiometric information coming from cameras. These radiometric sensors provide practical photometric information useful for obstacle avoidance and path planning [13]. We argue that a joint exploitation of both geometry and radiometry of the scene would benefit from the complementarity of these different acquisition modalities (precision of the LiDAR and high resolution of Cameras).

While a hybrid 3D map combining geometry and photometry would offer crucial clues about the surrounding space, it is not sufficient for a reliable autonomous navigation. The autonomous agent needs also to grasp the semantic of the objects composing the environment. In the context of scene understanding, deep learning has gained an overwhelming attention among the computer vision community resulting in a wide variety of semantic segmentation approaches applied to different data modalities/representations such as AlexNet [2] for 2D images, PointNet [14] for raw point clouds, VoxelNet [15] for 3D voxels, and MeshCNN [16] for 3D meshes. Unlike 2D images, each of the aforementioned learned representations has proposed a different approach to circumvent the irregularity of 3D data and the lack of the commonly known input as a grid-like structure. However, all these methods share a common weakness notably in their inability to handle large scale data. More recently, Landrieu and Simonovsky [17] have introduced a new scalable semantic segmentation approach of point clouds. The scalability of this method was made possible thanks to a pre-processing step consisting of an over-segmentation of the input large scale point clouds resulting in a set of geometrically and radiometrically homogeneous points called *superpoints*. The latter are inspired by their 2D counterparts *i.e.* superpixels (*e.g.* SLIC superpixels [18]). These superstructures allow to process a set of data points at once instead of parsing individual points one by one. While being efficient, we argue that the way these superstructures are designed is sub-optimal. In SPG [17], 3D points that share the same geometric and photometric hand-crafted features (*e.g.* , verticality, planarity, eleveation, rgb color, etc.) are set to belong to the same superpoint while they may truly belong to objects of different semantic natures. As the subsequent semantic segmentation depends on the quality of the computed superpoints, we believe that learning the over-segmentation as well will enhance the results.

The success of deep learning approaches is partially tied to the availability of large scale rich data. In the context of autonomous navigation, several outdoor multi-modal datasets [19–21] have been proposed to evaluate qualitatively and quantitatively recognition methods. Even though most of these datasets are large scale, the majority of them come with the standard acquisition modalities typically 2D perspective images, depth maps and LiDAR point clouds. On the other hand, indoor datasets [22–24] offer more interesting data

formats using RGB-D devices. In addition to the annotated point clouds and perspective 2D images, these indoor datasets provide omnidirectional 360° RGB images and 3D textured meshes with their respective ground truth annotations. These data formats have stimulated the development of new approaches [25, 26] that contributed to the prosperity of indoor autonomous navigation. We believe similar set up should be provided for outdoor scenes to boost research in this direction.

Finally, it should be emphasized that the related work will be discussed in details in the coming chapters. More related to our thesis subject, we can conclude that, we have three main unresolved issues from the literature:

- To the best of our knowledge there is no available large scale multi-modal multi-format dataset comprised of 2D/3D data with their corresponding ground truth annotations.
- The current representations of 3D maps are limited to geometric properties. A new representation of the 3D map allowing a joint exploitation of photometric and geometric information is required.
- Current semantic segmentation methods are not able to leverage the local contextual information of 3D scenes while remaining able to efficiently handle its large scale.

Our contributions are centered around these three issues. In the next section, we first present a summary of our contributions, then we expose the organization of the rest of this document.

1.4 Thesis outline and contributions

1.4.1 Summary of contributions

As discussed earlier the objective of this thesis is to develop a global geo-referenced 3D map based on a novel representation comprising relevant geometric, photometric and semantic information. In this section, we present four contributions towards this goal.

- Our first contribution is a large scale multi-modal dataset that encapsulates a wide variety of common sensing modalities consisting of equirectangular depth maps, reflectance maps, both panoramic and perspective RGB images, 3D point clouds and 3D textured meshes along with pixel-wise, point-wise and face-wise annotations. In addition, we demonstrate the usefulness of the provided data by evaluating state-of-the-art methods in several computer vision tasks such as 2D image and 3D point cloud semantic segmentation as well as monocular omnidirectional depth estimation.
- As a second contribution, we propose a novel compact representation of a 3D map by reconstructing a textured mesh fusing the geometry of LiDAR scans and the radiometry of mobile mapping images. This work led to the following publication [27].
- Our third contribution consists in a thorough study and classification of the literature regarding over-segmentation methods of both 2D images, 3D point clouds and meshes.

- Finally we propose the first supervised over-segmentation method operating on both 3D point clouds and 3D textured meshes. We show that our method outperforms the competing algorithms in this task. We further achieve state-of-the-art results on the task of point cloud semantic segmentation when combining our method to an over-segmentation-based method designed for semantic segmentation [17]. The supervised over-segmentation of point clouds led to these two publications [28, 29].

1.4.2 Organization

As can be seen in the previous section, the research axes that we are interested in, cover different scientific problems. Therefore instead of presenting the related work in a separate chapter, we opted for a per-research axe approach. For each chapter, we review the most related work to ours before introducing the proposed contribution. The rest of this document is organized in three main chapters followed by a general conclusion.

In chapter 2, we introduce the first contribution of our thesis which is a multi-modal urban dataset. An extensive overview of the most relevant datasets close to our work is given in Section 2.1. We present our dataset in Section 2.2. First, the acquisition setup is exposed in Section 2.2.1. We explain the process of generating spherical images in Section 2.2.2, panoramic depth in Section 2.2.3 and reflectivety maps in Section 2.2.4. Data, annotation protocol and tools are presented in Section 2.2.5. Finally, we conclude this chapter by discussing potential future work.

Chapter 3 presents our second contribution which is the reconstruction of a large scale textured 3D mesh. In Section 3.1, we give an overview of the work related to the different blocks of our framework; namely − surface reconstruction and texture mapping. We present surface reconstruction algorithm in Section 3.2. The texture mapping approach is explained in Section 3.3. Finally, in Section 3.4 we expose our experimental results.

In chapter 4, we present our two last contributions which concern a supervised over-segmentation method for semantic mapping. In Section 4.1, the existing related work to over-segmentation and deep learning on 3D data is presented along with theoretical definitions of the main concepts of our method. We explain our approach in details in Section 4.2. Both Sections 4.3 and 4.4 introduce two applications of our method respectively on 3D point clouds and textured meshes.

Finally, we conclude this thesis in Chapter 5 where we discuss open problems and potential future work.

Chapter

2 A multi-modal dataset for urban scene understanding

Chapter content

Abstract

In this chapter, we present a multi-modal multi-format outdoor dataset. We start by reviewing and comparing the existing outdoor urban datasets. Then, we present the tools and methods of collecting and generating our hybrid data. Finally, we highlight its usefulness by testing several methods from the literature on the task of 3D point cloud semantic segmentation and omnidirectional monocular depth estimation.

Introduction

Deploying an autonomous navigation system in real world urban environments is far from being a trivial task. When humans are removed from the navigation equation, such fully automated systems require high performing sensors along with sophisticated algorithms

to achieve accurate and robust perception, localization, planning and control especially when safety concerns are at stake. In the last decade, the former aspect *i.e.* perception, which is mainly related to scene understanding, has witnessed a significant improvement with recent advances of data driven methods based on deep learning [30–37]. This surge in data driven approaches magnifies the need for benchmark datasets to conduct training and evaluation of deep models.

In this context, benchmarking efforts have a long tradition in the computer vision and robotics communities attempting to cover a wide range of real-life traffic scenarios using single or multiple sensors. Due to the ease of capturing and annotating 2D RGB images, there is a plethora of large scale image-only datasets [38–43]. Although cameras allow for accurate measurements of radiometric information on the image plane, this device suffer from several sensing capability limitations in terms of sensitivity to lighting conditions as well as a lack of the precise 3D geometric information. On the other hand, with recent developments in 3D range scanners, a great deal of effort has been devoted to producing LiDAR-only datasets [44–48] providing large volumes of point clouds used to solve common computer vision problems that require dense and high level of details such as mapping, localization, semantization to mention a few. However, despite the accuracy and precision a LiDAR may offer, it has been established [21] that each single sensor has its own failure mode in difficult conditions. A LiDAR in particular provides less semantic information. For instance, this sensor is unable to capture words on a sign or determine the color of the traffic lights not to mention the limited range typically up to $50 - 100m$. That is why hybrid datasets are of a paramount importance since they provide complementary multi-modal data of the scanned environment.

To address the aforementioned issues, several multi-modal datasets have been proposed [19–21, 49, 50]. However, all these datasets provide exclusively the traditional data formats; typically point clouds and/or RADAR measurements if available with RGB images and GPS/IMU data. Since there is no consensus among researchers regarding what is the best suited data format in terms of performance and efficiency for an autonomous navigation system, we believe that the community should have access to an unified but diverse multi-modal multi-format benchmark where a neat evaluation can be performed using a standalone data type or a combination of them. As part of the pLaTINUM project, we propose a new dataset that is designed with the explicit goal to spur research on how to generalize to complex unseen environments using a data fusion scheme of diverse data formats. Although indoor RGBD datasets containing similar data types exist [22, 24, 51] , to the best of our knowledge this outdoor dataset is the first one of its kind that provides annotated textured meshes along with point clouds, equirectangular images, depth and LiDAR intensity maps.

In this chapter, we present our new dataset encompassing both LiDAR and image modalities in which we provide a variety of mutually registered and geo-referenced data, namely, annotated spherical and perspective images, depth and LiDAR reflectance maps as well as annotated 3D point clouds and 3D textured meshes. This chapter is organized as the following: In Section 2.1, we give an overview of the most relevant datasets close to our work. We present our dataset in Section 2.2. First, the acquisition setup is exposed in Section 2.2.1. We explain the process of generating spherical images in Section 2.2.2, panoramic depth in Section 2.2.3 and reflectivety maps in Section 2.2.4. Data, annotation

protocol and tools are presented in Section 2.2.5. Finally, we conclude this chapter by discussing potential future work.

2.1 Related work

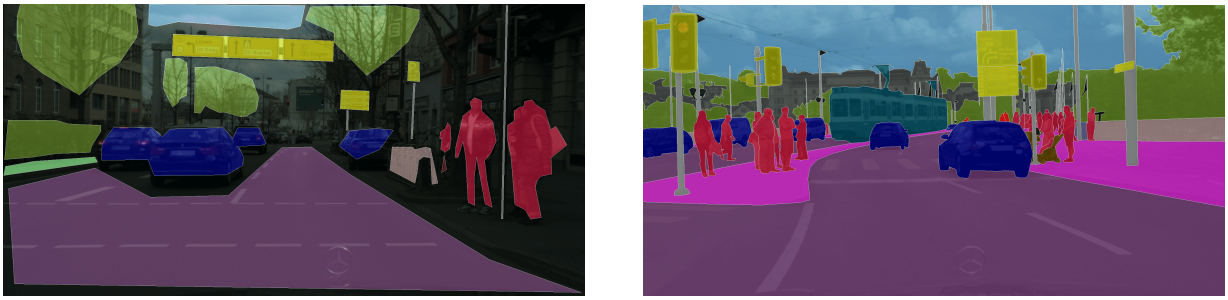
The last few years have witnessed the release of a large number of urban datasets which played a crucial role in pushing state-of-the-art research in autonomous driving making the task of newcomers to cope with the existing work and still up-to-date more challenging. Therefore, it becomes necessary to organize the sparse related literature into structured one in order to ease accessibility. In this section, we give an extensive overview of the existing datasets related to the autonomous navigation setting. Most of these datasets are focused either on 2D RGB images and video frames or 3D point clouds independently. On the other hand, few hybrid datasets have been introduced to allow a separate or joint processing of several imaging modalities.
First, we start by presenting the well known 2D datasets. Then, we overview the recent LiDAR-only datasets. Finally, synthetic and multi-modal datasets are discussed followed by an extensive comparison between all the available datasets.

2.1.1 Image-only datasets

Since the early ages of computer vision, 2D RGB image-only datasets have been widely used for scene understanding applications such as object detection, tracking, classification and segmentation. In the following, we present the most relevant 2D urban datasets used to train and evaluate algorithms in the context of autonomous navigation in their chronological order.

2.1.1.1 Cityscapes dataset [40]:

The Cityscapes dataset [1] is one of the most commonly used datasets for semantic urban scene understanding. The dataset is comprised of various set of stereo video sequences acquired by a mobile car in 50 cities in Germany and neighboring countries during the span in several months covering different seasons of the year.



(a) Coarse annotation (b) Fine annotations

Figure 2.1: *Illustration of the fine and coarse annotations from Cityscapes dataset [40]*

The dataset provides 5000 images with high quality annotations at the pixel-level including instance-level annotations for both vehicles and pedestrians in addition to 20000

[1] https://www.cityscapes-dataset.com/

coarse annotations as shown in Figure 2.1. These annotations are divided into 8 major categories; construction, nature, vehicle, flat, object, sky, human and void. Apart from the class sky, each group is split into several subgroups forming 30 visual classes in total. The proposed benchmark offers three different tasks for semantic urban scene understanding; pixel-level and instance-level semantic labeling as well as panoptic semantic segmentation more recently.

2.1.1.2 Mapillary vistas dataset [52]:

Mapillary Vistas dataset [2] is one of the largest 2D street-level image datasets. It is five times larger than Cityscapes [40]. Mapillary contains 25000 high resolution images recorded from all over the world covering the six continents at various season, daytime, and weather conditions. These road-scene images were captured from various viewpoints using a broad range of imaging devices (mobile phones, tablets, professional cameras...) by photographers with different levels of experience; hence offering a richness of details and a global geographic reach. Figure 2.2 depicts a sample of the dataset showing the accurate annotations and the diversity in the distribution of classes in different weather conditions.

(a) fine annotation of a complex scene

(b) annotations in a snowy weather

Figure 2.2: *Illustration of the fine annotations from Mapillary Vistas dataset [52]*

Dense and fine-grained annotations are available for 7 major categories (Object, nature, animal, human, void, marking and construction) leading to 152 object sub-categories with additional instance-specific labels for 100 classes.

Similarly to [40], Vistas benchmark proposes several challenges to evaluate computer vision algorithms such as the classic semantic segmentation task, object detection and more recently panoptic segmentation.

2.1.1.3 Apolloscape dataset [41]:

Released by Baidu, Apolloscape [3] is a large scale video dataset with rich annotations designed with the aim to deal with the challenges of street-scenes understanding in various traffic conditions. The data volume of this dataset is nearly 10 times greater than the

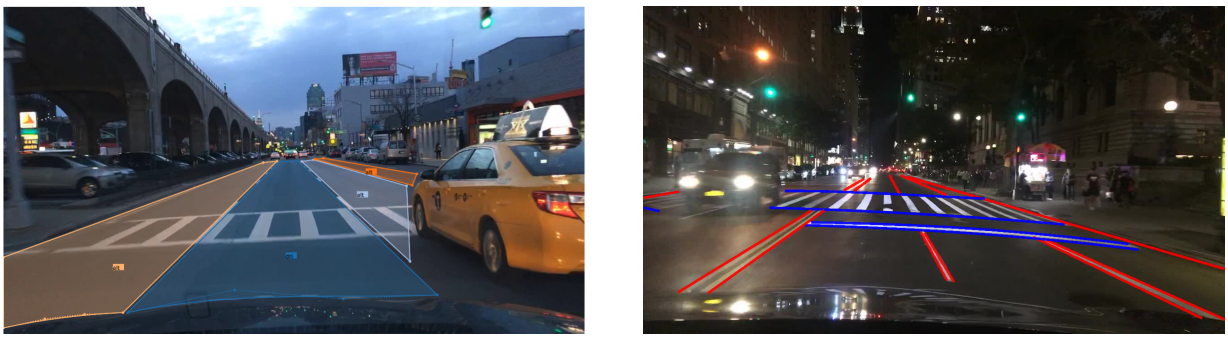
[2] https://www.mapillary.com/dataset/vistas
[3] http://apolloscape.auto/

driving dataset Cityscapes [40]. The initial release contains 143906 video frames with per-pixel dense semantic labels. Instance-level annotations of 89430 images are also provided for the movable objects in the collected scenes. Moreover, the dataset offers 28 different lane markings annotations defined based on lane boundary attributes including color and type.

The featured tasks targeted by this benchmark are mainly semantic segmentation with 25 classes divided in 5 groups (movable, nature, object, infrastructure, surface) as well as instance-level video object segmentation.

2.1.1.4 BDD100K dataset [43]:

This dataset [4] encompasses $100K$ high resolution videos ($720p$) at high frame rates ($30fps$) collected from more than $50K$ rides of normal real-world driving sessions covering Berkeley, San Francisco, New York and other regions in the United-States. Each video is about 40-second long and comes with GPS/IMU data recorded using mobile phones to show rough trajectories. To ensure a high level of diversity, BDD100K was recorded in a crowd-sourcing manner like [52] in different weather conditions including sunny, rainy and snowy as well as at daytime and nighttime.



(a) Annotation of a driveable area

(b) Annotation of lane markings

Figure 2.3: *Illustration of lane marking and driveable area annotations from BDD100K [43]*

Fine-grained annotations of the 10^{th} second keyframe of each collected video is provided leading to 120 million annotated images. In addition to the classic recognition tasks; namely object detection using bounding boxes, semantic segmentation and instance segmentation with dense pixel-wise annotations, $200K$ additional keyframes were also labeled to perform lane markings recognition as well as driveable areas segmentation as shown in Figure 2.3.

2.1.1.5 Other datasets [53–55]

Other interesting datasets were also introduced to boost research in autonomous driving. In the following, we describe these methods briefly.

In [53], Ha *et al.* propose a multi-modal dataset [5] containing 1569 RGB-thermal urban scene images (820 taken at daytime and 749 taken at nighttime) captured in both visible and thermal infrared spectrum ($814\,\mu m$) annotated exclusively with 8 classes of obstacles

[4] https://bdd-data.berkeley.edu/

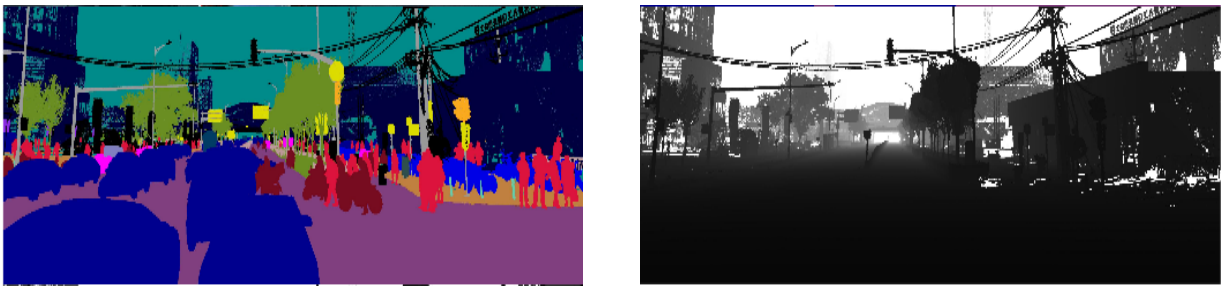
[5] https://www.mi.t.u-tokyo.ac.jp/static/projects/mil_multispectral/

usually encountered in a driving scenario (curve, bike, car, car stop, color cone, guardrail, bump and person). In [54], the same setup is adopted to generate RGB-thermal dataset but this time tailored for the task of object detection.

Built upon Cityscapes dataset [40], Zhang *et al.* introduce [55] CityPersons: a new layer from [40] dedicated exclusively for the task of pedestrian detection. For each frame of the 5000 fine-annotations, bounding boxes annotations of the pedestrians in the scene are provided.

2.1.1.6 Discussion

Compared to the existing outdoor driving datasets, Apolloscape [41] is the first dataset that provides a rendered depth map of the static background for each pixel-level annotated image along with the pose information at centimeter level. Figure 2.4 depicts a labeled scene with its corresponding static depth map. While depth information is robust to light variations, the dataset provides only static depth of urban scenes which are known to be very dynamic where there is substantial motion.



(a) pixel-level annotated scene

(b) depth map of the static background

Figure 2.4: *Illustration of an annotated scene and its corresponding depth map of the static background from [41]*

Table 2.1 shows a comparison between the different datasets in terms of the size, mean of acquisition, weather conditions and locations.

Dataset	Year	# Images	# labels	Lane labels	Location	Acquisition
Cityscapes [40]	2016	25K	30	no	50 cities (Europe)	moving car
Mapillary Vistas [52]	2017	25K	152	2	6 continents	crowdsourced
BDD100K [43]	2017	120 M	19	8	United-states	mobile mapping
Apolloscape [41]	2018	144K	25	28	China	mobile mapping

Table 2.1: *Image-only datasets comparison*

Both Vistas [52] and Cityscapes [40] datasets do not provide pose information of each image due to the way how they were collected. The lack of the geo-referencing information is a major obstacle for several computer vision applications such as image-based localization, pose estimation and mapping among others. Indeed, 2D images in general are the result of a perspective projection of a 3D scene into the camera plan. We argue that such representation does not capture the inherent structure of the recorded 3D scene. In fact, pixels in image space have different physical measurement in the 3D real-world. Consequently, several 3D datasets captured by LiDAR sensors were introduced to handle the limitations of 2D-only datasets.

2.1.2 LiDAR-only datasets

Due to the popularity of the autonomous driving technology, LiDAR-only datasets have received an increasing attention during the past few years. In this section, the most recent urban LiDAR-only datasets are discussed in details.

2.1.2.1 Oakland dataset [44]

Acquired by a mobile mapping system equipped with a monofiber LiDAR in Oakland, Pittsburgh, this dataset [6] contains nearly 1.6 Million points annotated w.r.t. 44 classes divided mainly into five categories; vegetation, wire, ground, pole/tree-trunk and facade. The goal of this benchmark is to assess the task of 3D dense point-wise semantic segmentation.

2.1.2.2 Paris-rue-Madame database [45]

This dataset [7] contains 3D MLS data acquired in the street *Rue-Madame* of the 6^{th} district in Paris, France using a mobile mapping plateform prototype called LARA2-3D equiped with a Velodyne HDL32 sensor. It consists of 160 meters of acquisition leading to 20 million points having each 4 attributes (X, Y, Z, and reflectivety) in addition the label and class attributes coming from the per-point and per-object annotations which have been carried out manually with respect to 26 classes distribution. The featured recognition tasks of this benchmark are mainly point-wise semantic segmentation and object segmentation.

2.1.2.3 IQMULUS & TerraMobilita [46]

The TerraMobilita benchmark [8] consists of a large scale 3D MLS data acquired from a dense urban environment in Paris, France using a mobile mapping system equipped with a LiDAR sensor of type RIEGL VQ-250. Among the 10 km recorded data, only 210 meters approximately were manually annotated at point-level and object-level allowing for tasks like 3D dense semantic segmentation and object segmentation. In addition to the label and class attributes, XYZ, intensity and number of echoes of the LiDAR for each point are also provided. The hierarchy of the semantic classes is defined with respect to three major classes; Surface including (ground, building, etc.), Object (natural, static and dynamic) and Others (unclassified, undefined, etc.).

2.1.2.4 Semantic3D dataset [47]

Semantic3D [9] is a large scale point cloud dataset containing nearly 4 billion annotated 3D points of rural and urban environments in central Europe. Recorded using a high resolution static scanner, dense per-point labels are provided with the goal to evaluate semantic segmentation algorithms with respect to 8 classes; man-made terrain, natural terrain, low vegetation, high vegetation, cars, scanning artifacts and buildings. Moreover, colorization is also performed using camera images captured during the scan offering thereby per-point RGB attributes.

[6] https://www.cs.cmu.edu/~vmr/datasets/oakland_3d/cvpr09/doc/
[7] http://www.cmm.mines-paristech.fr/~serna/rueMadameDataset.html
[8] http://data.ign.fr/benchmarks/UrbanAnalysis/
[9] http://www.semantic3d.net/

2.1.2.5 Paris-Lille 3D dataset [48]

Using a mobile mapping truck equipped with a Velodyne HDL-32E LiDAR mounted on its rear, this dataset [10] was collected in two different agglomerations in Lille and a third one in Paris France. Nearly two kilometres of acquisition were conducted leading to more than 140 Million points annotated manually with respect to 50 classes similar to [46] with minor changes. Indeed each annotated point has several attributes namely its XYZ-position, the position of the LiDAR as well as intensity, per-point and per-object labels.

2.1.2.6 Discussion

Oakland dataset [44] is a relatively small dataset designed before the emergence of deep learning in order to evaluate methods based on graphical probabilistic models which suffer from their limited scaling capabilities making them not suited to simulate real-world large-scale driving scenarios.

Paris-rue-Madame [45] was segmented and annotated using a semi-automatic method thereby introducing an algorithmic bias that cannot be corrected with hand refinement. An illustration of the annotation inaccuracies in this dataset is shown in Figure 2.5.

Figure 2.5: *Illustration of the annotation inaccuracies from [48]: Wheels of the vehicle are confused with the ground*

The huge size of Semantic3D dataset [47] can be explained by the type of the LiDAR used to collect this dataset. A static LiDAR scanner is known to be more precise and denser than mobile scanners. However this advantage comes with several shortcomings mainly the an-isotropic density of the obtained point cloud since it depends on the varying distance of the static LiDAR and the scanned objects in the scene as well as the induced occlusions due to the fixed position of the scanner.

IQMULUS & TerraMobilita dataset [46] proposes an innovative way to annotate the point cloud by considering the sensor-space topology projection of the 3D points into 2D images. In addition to the loss in precision during annotation because of the occlusions induced by projecting 3D objects into 2D, only a small part of this dataset was released making the task of training data-hungry learning algorithms complicated.

Paris-Lille 3D dataset [48] tries to circumvent the labeling and occlusions problems of the previously released LiDAR-only datasets by first adopting a fully manual annotation

[10] http://npm3d.fr/paris-lille-3d

policy in order to reduce the algorithmic bias of semi-automatic annotation methods and second using a multi-fiber LiDAR to ensure a watertight acquisition of the environment. However, their LiDAR measurement range is limited to only 20 meters. Additionally their dataset consists of 2 straight paths with no cross-roads or sharp turns to mimic real-life navigation scenarios. In table 2.2, we provide an overview of the urban LiDAR-only datasets in terms of size, type of the LiDAR and location.

Dataset	Year	Size (m)	LiDAR	# classes	# points	Location
Oakland [44]	2009	1510	SICK-LMS laser	4	1.6 M	Oakland
Paris-rue-Madame [45]	2013	160	Velodyne HDL32	17	20 M	Paris
TerraMobilita [46]	2014	210	RIEGL LMSQ-120i	22	15 M	Paris
Semantic3D [47]	2017	N/A	Static scanner	8	4 B	central Europe
Paris-Lille 3D [48]	2018	1940	Velodyne HDL32	50	140 M	Paris Lille

Table 2.2: *LiDAR-only datasets comparison*

Similar to the 2D case, annotating 3D data manually or semi-automatically is a hard task since it implies a trade-off between the annotation accuracy and manpower required to conduct manual annotations. That is why synthetic datasets has bloomed in recent years.

2.1.3 Synthetic datasets

Alternatively to collecting and manually annotating 2D or 3D data, synthetic datasets propose a new paradigm to assess vision algorithms in the context of autonomous driving called virtual simulation. In the literature, several approaches have been proposed to get as much as possible rich and diverse data of urban environments through simulation. In this section we review the most popular virtual datasets and simulators.

2.1.3.1 SYNTHIA dataset [56]:

SYNTHIA dataset [11] is a set of photo-realistic images rendered from a virtual city with an European style created using Unity game engine [12] acquired from different view-points in several locations. A large volume of data is provided (more than 200000 HD images with their corresponding pixel-wise annotations). Populated with realistic urban models, it offers a collection of rich and diverse scenes mimicking the real world driving scenarios in various weather conditions including rainy, cloudy and snowy weather as well as a drastic change in illumination conditions overall the dataset.

2.1.3.2 Virtual KITTI & vKITTI3D datasets [57, 58] :

Virtual KITTI [13] is a synthetic photo-realistic dataset built upon the original KITTI dataset [49] using a real-to-virtual world cloning method. It consists of 50 high-resolution monocular video sequences resulted in nearly 21160 frames generated from 5 different virtual worlds created using Unity software. These images are fully annotated for 2D object detection and tracking, instance and semantic segmentation.

[11] http://synthia-dataset.net/
[12] https://unity.com/fr
[13] https://europe.naverlabs.com/research/computer-vision/proxy-virtual-worlds/

Using the available camera parameters (intrinsic + pose), five virtual video sequences of [57] were projected into the 3D space giving birth to a new dataset vKITTI3D [14] [58] composed of a collection of 3D annotated point clouds with respect to 13 semantic classes similar to [57]; terrain, building, road, car, traffic signs, among others.

2.1.3.3 VIPER dataset [59]:

The VIsual PERception dataset referred to as VIPER [15] is a virtual dataset built based on the modern game Grand Theft Auto V (GTA) without acessing its source code. It consists of more than 250000 frames coming from a set of video sequences collected while driving, riding or walking a total of 184 kilometers in the simulated urban environment under different ambient conditions. The benchmark features diverse vision tasks since it offers annotations for 3D scene layout, optical flow object detection, tracking, semantic and instance segmentation.

2.1.3.4 CARLA Simulator [60]:

CARLA [16] is an open-source simulation-platform for urban driving. CARLA simulates a dynamic urban environment composed of 3D models of static objects such as buildings, traffic signs and infrastructure as well as dynamic objects like pedestrians and vehicles. By providing a simple interface between an agent and the virtual world, different navigation scenarios were implemented to imitate various traffic real-world configurations under different weather and illumination conditions. This tool provides also several sensors and pseudo-sensors simulations including RGB cameras, ground-truth depth maps and dense pixel-wise annotations for semantic segmentation w.r.t. 12 classes; sidewalk, building, fence, road, sign and others.

2.1.3.5 Discussion

Operating and instrumenting a mobile platform in order to collect urban driving data for training and evaluating autonomous navigation algorithms require significant funds and manpower. That is why the computer vision and robotics communities are increasingly investing much more time and effort in producing simulation-based datasets. Table 2.3 depicts a comparison between the previously described state-of-the-art virtual datasets.

Dataset	Year	Modality	# classes	Annotation	Origin
SYNTHIA [56]	2016	2D	13	220K images	rendered
Virtual KITTI [57]	2016	2D	13	21K frames	synthesized
vKITTI3D [58]	2017	3D	13	15M points	synthesized
VIPER [59]	2017	2D	12	254K frames	rendered
CARLA [60]	2017	2D + depth	12	Unlimited	simulated

Table 2.3: *Synthetic datasets comparison*

Such virtual benchmarks facilitate not only the data collection and annotation policy but also the reproducibility and controllability of certain complicated and risky driving

[14] https://github.com/VisualComputingInstitute/vkitti3D-dataset
[15] https://playing-for-benchmarks.org/
[16] http://carla.org/

real-world scenarios without any physical engagement. Besides, they offer an opportunity for standardizing driving tests under different weather conditions and locations by giving feedback instructions such as freezing and replaying – a feature that is not available in real-world conditions. However, we think that this trend suffers from several disadvantages. First, as far as we are concerned virtual datasets have a limited perceptual, physical and behavioural modeling capabilities of real urban scenes since most of them are based on game engines. Figure 2.6 shows a sample from [56] which depicts low resolution texture when zooming. Second, to the best of our knowledge, few of these studies [56] [60] have demonstrated the transferability of these simulation-based trained models to a real driving scenario in an urban environment for deployment.

Figure 2.6: *A sample from [56] that shows unrealistic low resolution texture*

On the other hand, multi-modal datasets acquired with mobile mapping vehicles have recently drawn too much attention.

2.1.4 Hybrid datasets

Generating hybrid datasets comprised of GPS/IMU data, RGB cameras and range sensors (LiDAR and/or RADAR) can be a prohibitively expensive task. In fact, such datasets are more challenging to collect and annotate compared to single-modal datasets since they require integrating, synchronizing and calibrating different sensors at the same time. In this section, we present the current state-of-the-art hybrid datasets that are related to our work.

2.1.4.1 KITTI dataset [49]:

KITTI vision benchmark [17] is one of the most well-known driving benchmarks. It is a multi-modal dataset collected using a moving car while driving in Karlsruhe Germany for nearly 6 hours. Diverse traffic scenarios of dynamic and static objects were recorded

[17] www.cvlibs.net/datasets/kitti

in rural and city areas using a variety of sensor modalities including RGB and gray-scale cameras, a 3D low-cost Velodyne laser scanner and a high-precision GPS/IMU unit resulted in $120K$ 3D point clouds and $41K$ frames. Annotations are provided in forms of 3D bounding boxes of nearly $200K$ objects. Several vision tasks can be assessed using this benchmark including stereo matching, optical flow estimation and object detection, among others.

2.1.4.2 Apollo dataset [19]:

In order to assess the task of future trajectory prediction of traffic agents in urban environments, Baidu research released a large-scale hybrid dataset [18]. To generate these traffic trajectories, a set of sensors mounted on a mobile mapping platform were used including RGB cameras, LiDAR (Velodyne HDL 64E S3), a RADAR and a localization inertial navigation system (GPS/IMU). It consists of more than $146K$ HD images with $20K$ point clouds. The featured tasks are object detection, lane segmentation, scene parsing and self-localization.

2.1.4.3 KAIST dataset [50]:

Targeting complex urban environment, KAIST [19] is a multi-spectral dataset that provides RGB/thermal images, RGB stereo images, LiDAR point clouds and GPS/IMU data at different time slots (day an night). Captured in various region in Seoul, South Korea, this benchmark was designed with the goal to deal with a wide range of vision problems in urban environments such as localization, depth estimation, object detection, scene parsing and driveable region detection. More than $8.9K$ annotated frames are available for the previously mentioned tasks.

2.1.4.4 H3D dataset [20]:

H3D [20] is a large-scale multi-modal dataset collected in diverse areas in San Francisco, USA by the Honda Research Institute using a moving car equipped with 3 full-HD RGB cameras, a Veldoyne HDL-E64 S2 and GPS/IMU unit. As opposed to KITTI dataset [49], this dataset is considered among the first ones that provide full-surround (360° view) 3D multi-object annotations. It consists of $27K$ frames collected from 160 crowded urban scene with a total of 1.1 million 3D bounding box annotations. The main featured vision tasks are 3D object detection and tracking.

2.1.4.5 NuScenes dataset [21]

NuScenes [21] is so far the largest multi-modal dataset for autonomous driving at the time of writing. Recorded in two cities that are known with their high density traffic and challenging situations (Boston and Singapore), 1000 $20s$-length driving scenes were manually selected to cover a wide range of driving manoeuvres and unexpected behaviours of the dynamic objects in the scene. Using a set of synchronized sensors (6 RGB cameras, a LiDAR, a RADAR and an inertial navigation system GPS/IMU), this benchmark offers

[18] http://apolloscape.auto/
[19] http://irap.kaist.ac.kr/dataset/
[20] https://usa.honda-ri.com/h3d
[21] https://www.nuscenes.org/

approximately 1.4 million geo-referenced images, $390K$ LiDAR spans, 1.4 million RADAR sweeps in different weather and day/night times. In addition to 360°-view annotations for the entire sensor modalities resulted in 1.4 million object bounding boxes, NuScenes provides annotations for the classes attributes such as pedestrian pose and car states. Similar to [20], the featured tasks are mainly object detection and tracking.

2.1.4.6 Other datasets [61–63]

Oxford RobotCar dataset [22] proposes a 1000Km acquisition of multi-modal data including raw 20 million RGB images along with LiDAR and INS data after traversing 100 times the same route in Oxford, UK during one year, albeit no semantic labels are available.

In [62] a large scale driving dataset was designed to assess the task of driving behaviour prediction. It is comprised of more than $10K$ frames of diverse traffic scenes collected using a mobile platform equipped with a load of perception sensors namely a Veldoyne HDL-32E LiDAR scanning with a density equal to $700K$ points per second and an RGB dashboard camera capturing video sequences at a rate of 30-frames per second.

[63] proposes an outdoor multi-modal dataset for place categorization. It encompasses two parts. The first one consists of 650 static panoramic scans of size 9 million 3D colored points each, captured using a FARO FOCUS3D scanner synchronized with an RGB camera. The second one contains more than $32K$ sparse full-surround scans ($70K$ 3D points) collected using a Velodyne HDL-32E LiDAR.

2.1.4.7 Discussion

Compared to image-only and LiDAR-only datasets, the number and size of the hybrid datasets remain relatively small. Table 2.4 shows a comparison between the most recent driving multi-modal datasets in terms of size, location, sensor diversity and the provided annotations. Indeed, this can be explained by the expensive cost of collecting and annotating such datasets. Furthermore, the majority of current multi-modal datasets feature tasks like 2D/3D object detection and tracking since providing 2D/3D annotations for these tasks is much easier than cumbersome and dense pixel-wise/point-wise annotations for semantic segmentation or instance segmentation. Moreover, despite the proliferation of imaging modalities and the tremendous advances in multi-modal deep learning techniques, most of the aforementioned datasets collected with RGB/thermal camera and static/dynamic LiDAR, provide only the traditional data formats typically 2D images and 3D LiDAR point clouds. Other interesting data formats which have been proven to be useful for indoor navigation [22, 24, 51] and better reflects the geometry/radiometry of the scene such as depth maps, LiDAR reflectance maps, panoramic images, and textured meshes, can be easily derived with almost no extra-cost. Therefore, we decided to design the first outdoor urban multi-modal driving dataset that provides the aforementioned sensing modalities.

2.2 pLaTINUM dataset

As discussed above, 3D multi-modal datasets are of a particular importance compared to single-modal ones since they provide the entire geometry of the objects and their surrounding context along with complementary RGB information which offers dense appearance

[22] https://robotcar-dataset.robots.ox.ac.uk/

Features	KITTI [49]	Panoramic3D[63]	Oxford [61]	KAIST [50]	Apollo [19]	H3D [20]	NuScenes[21]
Year	2012	2016	2017	2018	2019	2019	2019
Sensors	Stereo camera + LiDAR+ INS	Camera + static mobile LiDAR + GPS/IMU unit	fisheye/stereo + 2D/3D LiDAR + GPS/IMU	RGB/thermo cameras	Stereo cameras + 3D LiDAR + GPS/IMU	3 RGB cameras + 3D LiDAR + GPS/IMU	6 RGB cameras + LiDAR + RADAR + GPS/IMU
Annotations	2D/3D bounding box + pixel/ instance-level labels	labels for classification	not annotated	2D bounding box + drivable regions labels	2D/3D pixel/instance -level labels	3D bounding box	3D bounding box + pedestrian pose
Size	7481 frames 80256 objects	650 dense scan 32000 sparse scans	20 million images + 3 M frames	7512 frames 308913 objects	143906 images 89430 objects	27K frames > 1 million objects	1.4 million image + RADAR 390K LiDAR
Location	karlsruhe	Fukuoka	Oxford	Seoul	China	San Francisco	Boston/Singapore
RGB images	✓	✗	✓	✓	✓	✓	✓
Panorama	✗	✓	✗	✗	✗	✗	✗
Depth maps	✓	✗	✓	✓	✓	✗	✗
reflectance	✗	✗	✗	✗	✗	✗	✓
Point clouds	✓	✓	✓	✗	✓	✓	✓
Textured mesh	✗	✗	✗	✗	✗	✗	✗

Table 2.4: *An overview of the multi-modal driving datasets*

features. This type of datasets enables the development of joint and cross-modal learning models – a strategy that has been proven to be successful in various scene understanding tasks. In this section we present pLaTINUM dataset; a large scale multi-modal dataset providing 2D and 3D mutually registered data recorded using a mobile mapping vehicle (an upgraded version of Stereopolis II [64]). Our dataset belongs to the fourth category of the previously discussed datasets. In contrast to the previous work, pLaTINUM dataset encompasses several data formats that recently have drawn much attention in the vision community namely spherical images and their corresponding 360° ground truth depth maps, LiDAR reflectivety maps and textured meshes. In addition, traditional data formats such as 3D LiDAR point clouds as well as 2D front-facing images are also provided. The featured tasks are mainly point-wise and pixel-wise semantic segmentation and omnidirectional 360° monocular depth estimation.

2.2.1 Acquisition details

In this section we start first by describing the driving plan, the sensors setup, the policy of generating spherical images as well as rendering depth and reflectivity maps. Second, we explain the annotation process of the 2D images and 3D point clouds followed by overall statistics. The final geo-referenced frame which consists of a 3D textured semantic mesh is left to the next chapter as it represents a standalone contribution.

2.2.1.1 Driving plan

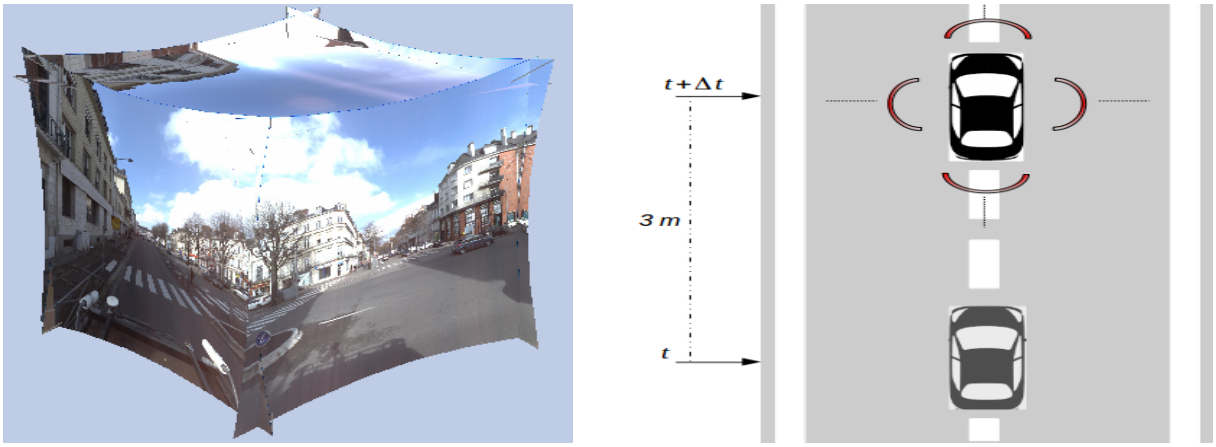
The dataset was recorded in the city of Rouen located in the north-west of France using a mobile mapping platform. It consists of nearly 6 hours of acquisition (more than 2 hours of continuous surveying) at an average speed of $15Km/h$ resulted in nearly 17 kilometers of mapped routes as illustrated in Figure 2.7. The driving streets were carefully chosen to capture a diverse set of urban and residential locations in general traffic situations with many static and dynamic objects. This acquisition was made on March 16, 2016 at two different time slots (morning and afternoon).

2.2.1.2 Sensors setup

Developed by the French National Mapping Agency (IGN), the upgraded version of Stereopolis II [64] is equipped with the following set of sensors:

Figure 2.7: *Illustration of the mapped trajectory in Rouen, France*

- **5 full HD cameras**: mounted on a panoramic head within a rigid body to ensure stability of the relative camera poses during acquisition. Each camera is facing a different viewing direction which corresponds to the five upper faces of a cube as shown in Figure 2.8. This allows the acquisition of 5 high resolution 12-bit RGB perspective images (2048 × 2048) that are thereafter linearly converted to 8-bit images. To ensure the same exposure time, the 5 cameras are perfectly synchronized in such a way they are triggered at the same time for each spacing distant equal to 3 meters.



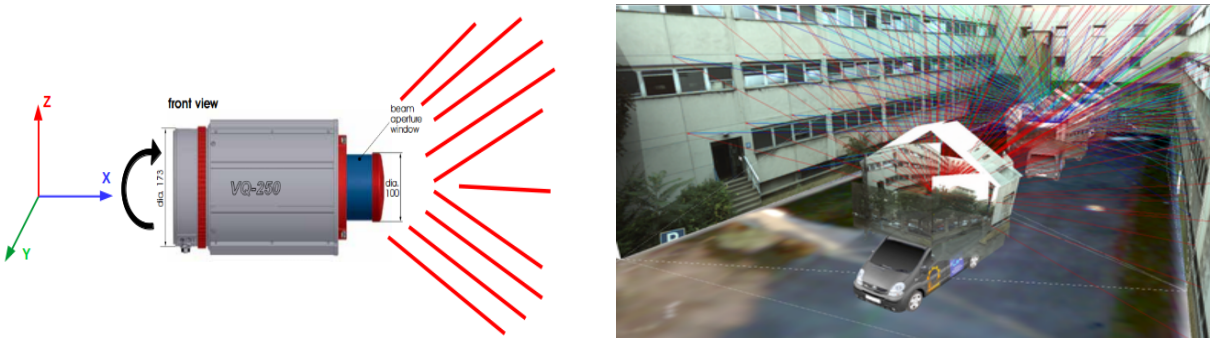
(a) Illustration of the five upper faces of a cube acquired by the 5 cameras installed on Stereopolis

(b) Top view of the acquisition geometry: 5 cameras recording 5 images each 3 meters while moving

Figure 2.8: *Illustration of the acquisition configuration of the 5 RGB cameras*

- **1 LiDAR RIEGL VQ-250**: mounted transversely on the rear of the platform, it scans the orthogonal plane to the front-facing trajectory of the moving vehicle with a 360° field-of-view as depicted in Figure 2.9. It rotates at a frequency equal to 100 Hz and emits 3000 pulse per rotation which corresponds to an approximate angular resolution equal to 0.12°. Each pulse has between 0 to 8 echoes producing thereby

an average of 250000 points per second. This scanner is able to acquire highly dense point clouds and precise/accurate measurements.



(a) the used LiDAR RIEGL VQ-250 in this survey (b) the orthogonal plan to the trajectory is scanned

Figure 2.9: *Illustration of the acquisition configuration of LiDAR while the car is moving*

- **Inertial navigation system:** comprised of two GPS devices, an Inertial Measurement Unit (IMU) and a wheel odometer. This system provides an instantaneous absolute position and orientation of the vehicle in the global system reference.

2.2.2 Spherical image generation

Spherical or panoramic images are a very popular representation especially within the robotics community. Usually captured using 360° cameras, these full surrounding images are increasingly influential in several contemporary technologies such as virtual and augmented reality (VR/AR), mobile robots and social media. However, this is not yet the case for autonomous driving. Even though, omnidirectional images are considered by far a more pertinent representation of the environment than conventional planar images since they provide an entire 360° field of view, so far the recent automotive datasets have mainly focused on producing perspective forward-facing images to assess object detection, tracking and semantic segmentation among others. To address the lack of annotated spherical images in the current urban driving datasets, [65] proposed an approach to adapt the recent deep network architectures trained on front-facing rectilinear images to operate on equirectangular panoramic imagery producing thereby a new benchmark based on CARLA simulator [60] to evaluate object detection. To the best of our knowledge, there is no publicly available real-world benchmark to assess semantic segmentation based on spherical 360° urban images. Therefore we provide a collection of annotated equirectangular images to deal with the lack of such type of data in the driving datasets. In this section, we explain our method of generating the spherical images and we show the obtained results.

2.2.2.1 Problem statement

Recent mobile devices and VR/AR headsets provide directly omnidirectional images. In the autonomous driving setting such images could be obtained directly using a a dual-fisheye pair or 360° action camera. As shown in Section 2.2.1, our mobile mapping vehicle is equipped with *five* cameras, each one of them is facing a different direction. This results in a set of five non-overlapping perspective images that are not able to perceive

the entire 360° field of view if treated separately. One way to get a full-surrounding view is to smoothly stitch these images into a single one following a planar, cylindrical or spherical geometry. Most of the traditional methods use the following pipeline introduced by Brown and Lowe [66] to reconstruct a panoramic image from different perspective images:

- Extract and match features between a pair of images
- Compute a transformation from the second to the first image
- Warp the second image so it overlays the first one
- Blend the images where they overlap
- Repeat the same procedure iteratively for the rest of the images to be stitched.

Thanks to the GPS/IMU system, we are able to provide the position and orientation in the camera and world frames for all the recorded perspective images. Indeed, having the pose information allows us to frame the stitching problem as an equirectangular projection of the rectilinear images sampled in a 3D spherical or cylindrical geometry. In Figure 2.10, we show an illustration of the process of generating our panoramic images from the raw perspectives images. This methodology enables us to bypass the slow features' computation and matching step of the traditional methods.

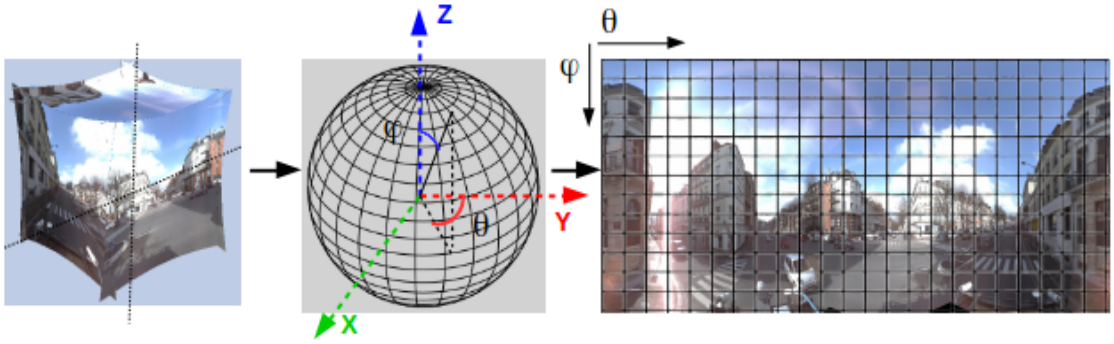


Figure 2.10: *Full-surrounding 360° image generation process. Left: the 5 input perspective images. Middle: Resampling of the input in the spherical geometry. Right: Equirectangular projection to flatten the sphere*

Formally, let I_s be an image sampled on a spherical coordinate system defined by (θ, ϕ) where $\theta \in [0, 2\pi], \phi \in [0, \pi]$ are respectively its longitude and latitude.

Let $I_f \in \mathbb{I}^{W_f \times H_f \times 3}$ be the corresponding flat RGB image in equirectangular projection defined by the set of 2D pixels (u_{equi}, v_{equi}) as

$$I_f = \{(u_{equi}, v_{equi}) | u_{equi} \in [0, W_f], v_{equi} \in [0, H_f]\}$$

W_f and H_f being respectively its width and height.

For a 3D point $P(x, y, z) \in \mathbb{R}^3$ on the viewing sphere I_s defined by its polar coordinates (θ, ϕ), the corresponding 3D position in the Cartesian coordinate system following the convention in Figure 2.10 is defined as

$$\begin{cases} x = \cos(\theta)\sin(\phi) \\ y = \sin(\theta)\sin(\phi) \\ z = \cos(\phi) \end{cases} \tag{2.1}$$

In contrast the equirectangular projection is conveniently defined in terms of the longitude θ and latitude ϕ of the sphere. From Eq. 2.1 we can compute

$$\begin{cases} \theta = \arctan(x/z) \\ \phi = \arcsin(y/r) \text{ with } r = \sqrt{x^2 + y^2 + z^2} \end{cases} \tag{2.2}$$

For the sake of simplicity, the computation of (θ, ϕ) can be rewritten in a matricial form using the function Ψ followed by an homogeneous normalization :

$$\begin{bmatrix} \theta \\ \phi \end{bmatrix} = \Psi \,.\, \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \Psi \,.\, \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \tag{2.3}$$

Typically the projection of P into a 2D image pixel (u, v) using a traditional rectified rectilinear camera model is defined by

$$\begin{bmatrix} u \\ v \end{bmatrix} = C_{rect} \,.\, \begin{bmatrix} x \\ y \\ z \end{bmatrix} = C_{rect} \,.\, \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \text{ where } C_{rect} = \begin{pmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.4}$$

f, (c_x, c_y) are respectively the focal length and the principal points of the camera.

In a similar way to the rectilinear projection, using Eq. 2.5 the equirectangular projection I_f of the spherically sampled image I_s can be written as

$$\begin{bmatrix} u_{equi} \\ v_{equi} \\ 1 \end{bmatrix} = C_{equi} \,.\, \begin{bmatrix} \theta \\ \phi \\ 1 \end{bmatrix} = C_{equi} \,.\Psi \,.\, \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \text{ where } C_{equi} = \begin{pmatrix} \gamma & 0 & c_\theta \\ 0 & \gamma & c_\phi \\ 0 & 0 & 1 \end{pmatrix} \tag{2.5}$$

In the projection matrix C_{equi} Eq. 2.5, γ is the angular resolution parameter akin to the focal length in rectilinear cameras and can be defined as the ratio between the horizontal field of view fov_θ and the image width W_f or the vertical field of view fov_ϕ and the image height H_f:

$$\gamma = fov_\theta / W_f = fov_\phi / H_f \tag{2.6}$$

To get a plausible equirectangular projection covering the 360° scene, γ needs to be tuned accordingly with respect to our mobile mapping vehicle configuration.

By combining the definitions from Eq. 2.4 and Eq. 2.5, we can easily establish the relationship between the rectilinear and equirectangular image coordinates projections:

$$\begin{bmatrix} u_{equi} \\ v_{equi} \\ 1 \end{bmatrix} = C_{equi} \,.\, \Psi \,.\, \left(C_{rect}^{-1} \,.\, \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \right) = \Gamma \,.\, \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \text{ with } \Gamma = C_{equi} \,.\, \Psi \,.\, C_{rect}^{-1} \tag{2.7}$$

This linear mapping in Eq. 2.7 allows the back and forth projections between the perspective rectilinear image and the equirectangular image – a key enabler of our approach to generate and annotate the panoramic images.

2.2.2.2 Results and discussion

Generating a full-surrounding 360° image from an equirectangular projection of the sampled rectilinear images on a spherical geometry is much faster than the traditional methods that require computing handcrafted features. Moreover, by using such sampling strategy, the equirectangular projection becomes agnostic to the number of input perspective images as long as they cover the 360° field of view. However, by doing so the final reconstructed panorama depicts a radiometric distortion and visible seams because of the overlap between the perspective images having interlaced fields of view as shown in Figure 2.11. In our application, we simply compute an average value of the pixels that are in the overlap regions normalized to the RGB frame $[0, 255]$. More sophisticated methods such as Poisson editing [67] can be used to get better visual results, albeit at the expense of the computation time.

(a) Visible seams at the overlap between perspective images

(b) The average of the pixels at the overlap partially correct the final panorama

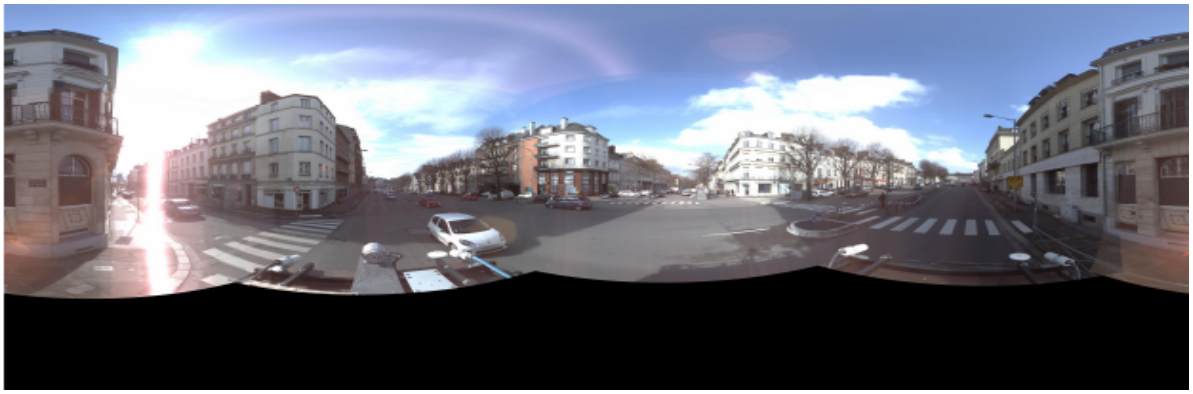
Figure 2.11: *Illustration of the visible seams in the overlap between perspective images at a vertical field of view* $fov_\phi \in [10°, 80°]$ *(best viewed on screen)*

In Figure 2.12, we show the equirectangular projection of the 5 perspective images with different angular resolutions γ. By considering the altitude and longitude of the 5 cameras with respect to the vehicle coordinate system, we find experimentally that for a panoramic image of size 8000×3200 the appropriate value for the vertical field of view is between $fov_\phi \in [10°, 130°]$.

As depicted in Figure 2.10, in an equirectangular projection, the longitude and latitude of the sphere are mapped to a vertical and horizontal grid coordinates. As the sphere is a non-developable surface, this mapping induces a significant distortion near the pole

The generated RGB panorama with $fov_\phi \in [10°, 80°]$

The generated RGB panorama with $fov_\phi \in [10°, 180°]$



The generated RGB panorama with $fov_\phi \in [10°, 130°]$

Figure 2.12: *The generated panoramic images at different vertical fields of view*

regions of the image which results in stretched pixels that make objects of the scene appear differently depending on their latitude in the sphere. As a result, classic convolutional neural networks CNNs, for instance, fail to encode the invariance related to this geometric transformation, hence reducing their discriminative powers.

2.2.3 Depth maps reconstruction

One of the fundamental challenges in 3D vision is understanding the three dimensional geometry of the real-world scenes. Known as depth estimation, this problem has been extensively studied by the computer vision community. Historically performed using a

stereo-pair of images [68], several overlapping images from different view-points [69] or by changing lighting in a static scene with a fixed camera [70], in the recent years, a particular focus has been directed towards monocular depth estimation where the goal is to estimate the depth of a 3D scene from a single RGB image [71–74]. With the democratization of 360° omnidirectional content, it becomes necessary to investigate the transferability of the recent advances in scene understanding from classic 2D perspective images to the full-surrounding panoramic images. However, the main bottle-neck for monocular depth estimation, in particular, remains the scarcity of datasets containing enough ground truth of equirectangular depth maps coupled with RGB panoramic images. In this context, only recently, a first attempt to tackle this problem has been introduced by Zioulis *et al.* [25] in which the authors proposed for the first time a network that is able to learn the task of monocular depth estimation from a single equirectangular RGB image. In order to evaluate the performance of the proposed architecture, [25] handle the unavailability of data by virtually rendering panoramic images from existing datasets (Matterport3D [22] and S3DIS [75]) in addition to their corresponding depth maps extracted from the z-buffer using Blender software [23]. While they show impressive results on indoor data, to the time of writing there is no outdoor dataset that we are aware of, which allows monocular depth estimation from spherical images of urban environments. Therefore, extrapolating these methods to the autonomous driving context is unfeasible for the moment.

To circumvent this issue, we present the first outdoor dataset containing both spherical RGB images and their corresponding ground truth depth maps. In this section, we explain the process of generating the spherical depth maps from our recorded data.

2.2.3.1 Rendering depth maps

To render the spherical depth maps from 3D indoor scenes, [25] used a path tracing renderer software by placing a point light source and a virtual spherical camera at the same position in the scene. This offers a colored image with its corresponding depth map extracted from the z-buffer that was generated as a result of the graphics rendering process. In our case, we implemented our own renderer using the *PBRT* library [24]. We consider the dense mesh [25] generated from the LiDAR acquisition as our 3D ground truth as its precision is at centimeter level. Since images and the mesh modalities are mutually registered and precisely timestamped, for each depth map we choose the origin O of the ray to be traced into the 3D mesh as the barycenter of the five perspective images' optical centers recorded at an instant t during the acquisition. In this way, the extracted depth values are the set of distances between the origin of the traced rays and the intersections with each triangle in the 3D mesh as shown in Figure 2.13. Akin to the panoramic images generation process, the reconstructed depth maps are sampled on a spherical geometry followed by an equirectangular projection.

2.2.3.2 Results and discussion

In a similar way to the panoramic images, equirectangular depth maps are reconstructed every spacing distance equal to 3 meters. To cover as much objects as possible in the rendered scene, we set the maximum depth value to 100 meters which corresponds to the

[23] https://www.blender.org/

[24] https://www.pbrt.org/

[25] The mesh reconstruction process is explained in details in the next chapter

maximum observable range of the used LiDAR in this survey. In Figure 2.14, we show the reconstructed equirectangular depth maps at different maximum range values. The images are saved as 16-bit PNGs. All depths beyond this maximum range assumes a value of 65535. Inversely, when the traced ray does not intersect any triangle in the 3D mesh, we assume a zero depth value (a vacuum).

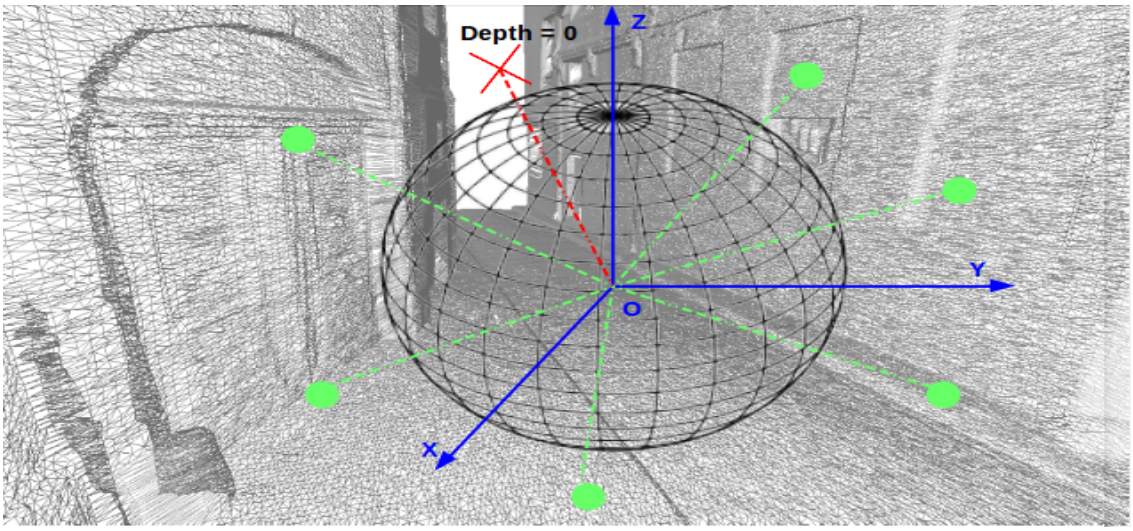


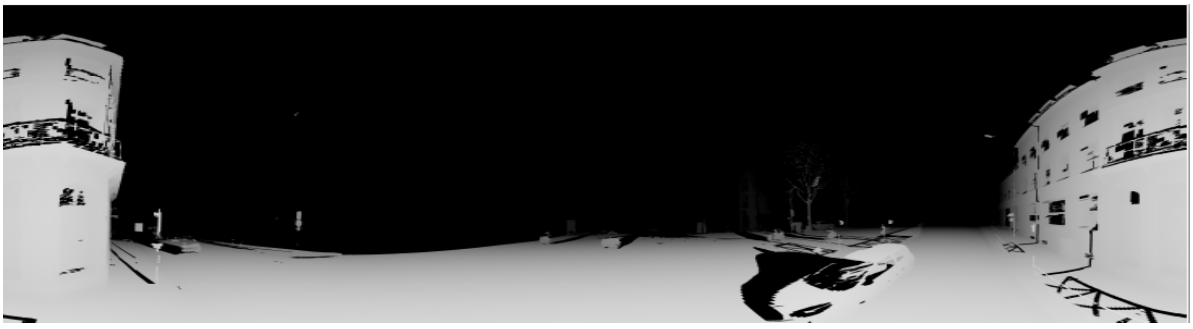
Figure 2.13: *Rays are traced from the center O of the spherical depth map to the triangular mesh. When there is no intersection depth is equal to zero (the red ray).*

Adopting a ray-tracing-based strategy to compute the ground truth depth maps offers complete depth measurements even for the moving objects in the scene which commonly results in holes and artifacts when stereo-based techniques are used instead. That is why Apolloscape dataset [41] provides exclusively depth maps of the static background. Nevertheless, this implies that the quality of the depth reconstruction depends mainly on the geometry of the mesh. Typically, the rendered depth maps are affected by two crucial factors; the mesh resolution and its water-tightness. For the former, the used LiDAR RIEGL VQ-250 is considered dense and accurate enough to be reliable since it provides nearly $250K$ points per second with a high precision at centimeter level. At this stage we abstain from decimating the dense mesh in order to preserve its density which comes at the cost of computation time which itself grows linearly with the number of triangles in the mesh. On the other hand, reconstructing a watertight mesh from a terrestrial mobile mapping LiDAR data is not guaranteed. In fact, due to the acquisition configuration illustrated in Figure 2.9, the final reconstructed mesh is likely susceptible to have occlusions. This can occur especially in both sides of the mapped streets where there are parked cars hiding the opposite side of the sidewalks or whenever the mobile mapping platform scans objects that are orthogonal to the scanning plan of the LiDAR. As a result some of the depth maps might be incomplete. More discussion about this aspect is available in the next chapter.

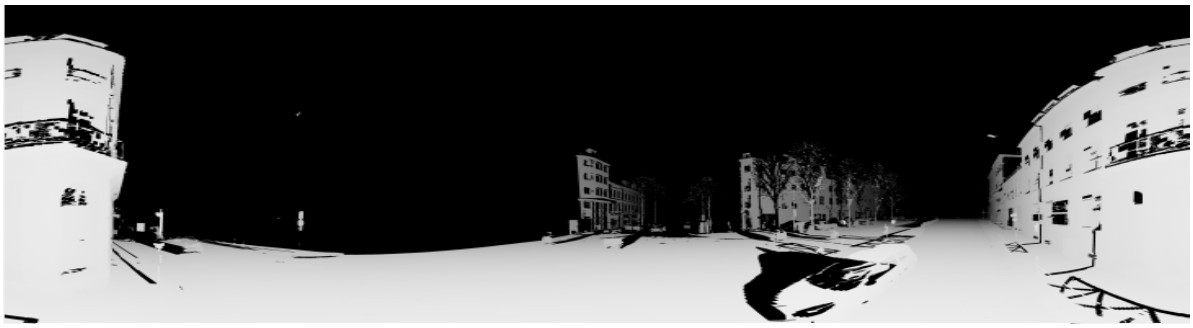
2.2.4 Reflectance maps generation

In the context of autonomous navigation, restraining the urban scene analysis exclusively to a geometric point of view could be insufficient for a comprehensive scene understanding. In the literature, the main radiometric property that has been extensively explored is the

Maximum depth range = 30 meters



Maximum depth range = 60 meters



Maximum depth range = 100 meters

Figure 2.14: *The reconstructed depth map at different maximum ranges. Distant buildings appears at a maximum range of 100 meters*

RGB information recorded basically from traditional cameras in the visible spectrum. In order to solve various driving perception problems such as lane markings and road detection, several studies [76, 77] proposed to leverage the radiometric information in the near-infrared domain provided by LiDAR sensors. Called reflectance, this measurement corresponds to the intensity of reflection observed on the photo-detector of a LiDAR sensor when a laser beam hits the scanned surface. A very popular feature in the remote sensing community, this radiometric measure is essentially affected by 3 factors; the distance to the object, the angle between the emitted ray and the normal to the scanned surface as well as the photometric properties of the scanned material. For instance, road markings such as lanes and pedestrian crossings are characterized by a higher reflectivity compared to the road pavement which make such information useful for the task of detecting these objects. Therefore, in addition to the equirectangular RGB images and depth maps, we provide also in our dataset the LiDAR intensity maps.

2.2.4.1 Rendering reflectivity maps:

Akin to the process of rendering 360° full surrounding depth maps, we use a ray-tracing-based approach to render reflectivity maps from the 3D LiDAR mesh considered as our ground truth. As illustrated in Figure 2.13, rays are traced from the barycenter of the five optical centers of the perspective images to the 3D mesh sampled in a spherical geometry. In fact, real-world surface materials are typically subject to local variations over the surface due to illumination changing or their physical inherent properties. In order to be able to appropriately model these spatial variations, we need to compute a mapping between the scanned surface parameters' space (u, v) and the rendered image space defined by (x, y). Then the reflectivity is modeled by the local spacial variation at the intersection point between the traced ray and the closest primitive in the 3D mesh computed using the partial derivatives of the mappings $u(x, y)$ and $v(x, y)$ from the image space (x, y) to the parametric space (u, v); $\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}$

2.2.4.2 Results and discussion

As for the case of depth maps and panoramic images we render a reflectivity map each 3 meters. In Figure 2.15, we show the LiDAR reflectivity map obtained by casting rays with a maximum range equal to 100 meters into the 3D mesh sampled in a spherical geometry and followed by an equirectangular projection.

Figure 2.15: *Illustration of the obtained LiDAR intensity map*

Even though the LiDAR intensity maps are considered a pertinent radiometric information that enables a neat discrimination between several urban semantic classes such as road markings and vegetation, in practice, it remains counter-intuitive how to make use of raw reflectivity. In essence, this radiometric feature deteriorates with difficult acquisition geometry (decreased incidence angle and increased distance to the scanned objects), poor environmental conditions (rainy and sunny weather), or material surface properties (specular or diffuse). Such complicated conditions results in unevenly-distributed LiDAR intensity across the scene making this feature irrelevant most of the time. Hence, the task of identifying reflectivity-sensitive objects based on the LiDAR radiometry becomes more challenging. Meanwhile, several studies tried to ensure a uniformly-distributed reflectivity. For instance, Guan *et al.* [78] propose to use a multi-thresholding technique which partitions the road into subsets by computing an optimal threshold for each subset. Yet,

this approach remains not robust to noise. Furthermore, the reflectivity maps rendering process depends essentially on the resolution and water-tightness of the 3D reconstructed mesh akin to the case of depth maps. As a result some of the rendered maps might depict holes or missing intensity values.

2.2.5 Data overview

The overall acquisition resulted in nearly 2 billion geo-referenced 3D points and about $40K$ oriented perspective images which corresponds to more than 500 GB of raw data. The dataset is organized in 3 sessions. Each session encompasses between 3 and 6 sections. A session is a time period during which a certain trajectory has been surveyed. A section, on the other hand, consists of a time interval which starts when triggering the acquisition software and ends by shutting it down. In each section, we provide raw RGB perspective images in PNG format, their corresponding pose and meta-data files in XML format, Spherical panorama, depth and reflectivity images in equirectangular projection in PNG format. In addition, 3D annotated point clouds and textured meshes are divided into temporal chunks and stored respectively in PLY and OBJ formats. Figure 2.16 illustrates a sample of our LiDAR point cloud and the corresponding registered images (in blue) collected in Rouen and visualized using Geolabx [26].

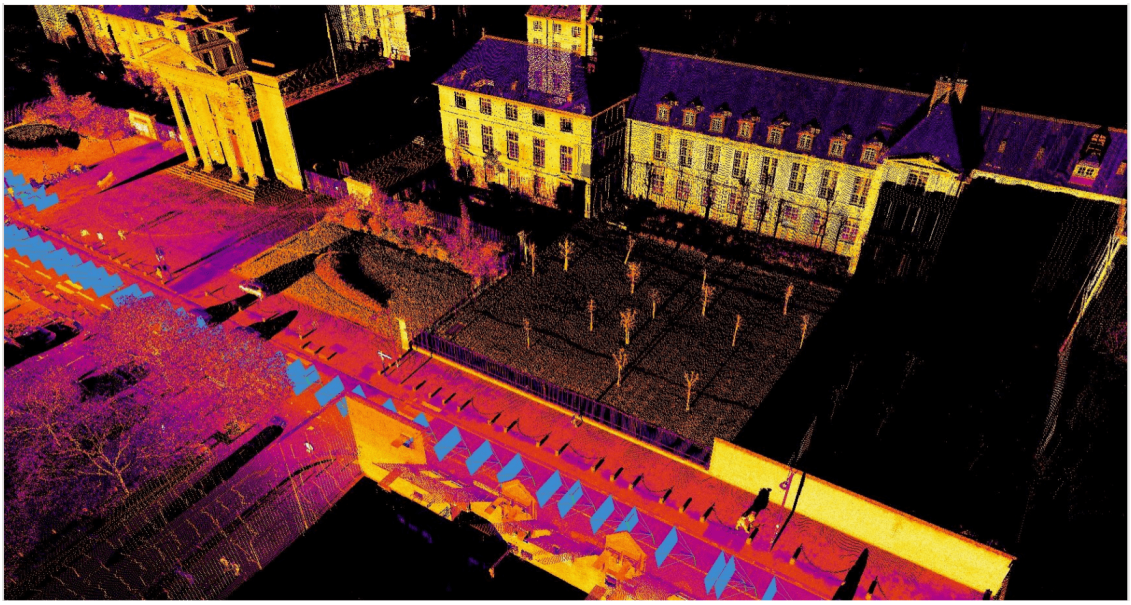


Figure 2.16: *Top view of a sample from our dataset that shows LiDAR point cloud and the recorded images (in blue)*

2.2.5.1 Annotation Policy

Generating ground truth data for training and evaluating learning-based algorithms is an extremely challenging task. In order to alleviate the prohibitive cost of manually annotating data, there has been several attempts to produce automatic or semi-automatic annotation tools to generate ground truth for different computer vision tasks as discussed in this survey [79]. However, as soon as such methods are adopted, ground truth will include algorithmic bias resulting in an extra-effort to handle labels inaccuracies and its

[26] Geolabx is a geo-visualization tool developed in LASTIG

errors rate. In our dataset we provide both dense pixel-wise 2D annotations of spherical and perspective images as well as dense point-wise 3D point clouds and 3D textured meshes. We annotated 15 classes covered by 5 groups. Table 2.5 shows the hierarchy of the annotated classes. The ID 0 indicates the ignored labels that correspond to none of the aforementioned classes which are not evaluated during testing. The specification of the classes is similar to common driving datasets such as Apolloscapes [41] and Cityscapes [40] with minor differences. In the following, we explain the details of the annotation methodology.

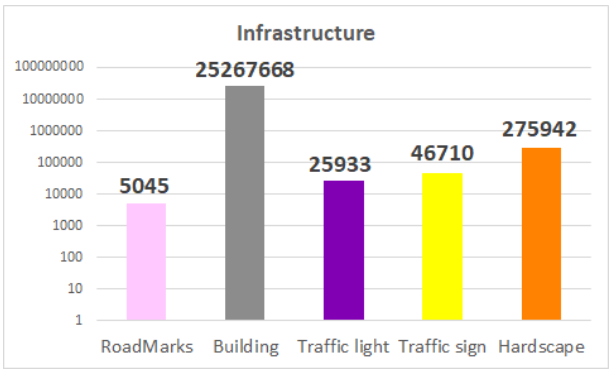
Groups	Classes	IDs	Details
Movable objects	Person	7	includes pedestrian/rider
	motorbike	10	
	bicycle	9	
	Vehicle	8	includes bus car and trucks
Surface	Road	1	
	Sidewalk	3	
Infrastructure	Building	4	includes walls/fences
	Road marks	2	
	Traffic light	11	
	Traffic sign	12	
	hardscape	13	includes pole, trash can, barriers
Nature	Low vegetation	5	
	Tree	6	
	Sky	14	
Void		0	other classes or confusing objects

Table 2.5: *Hierarchy of classes in our dataset*

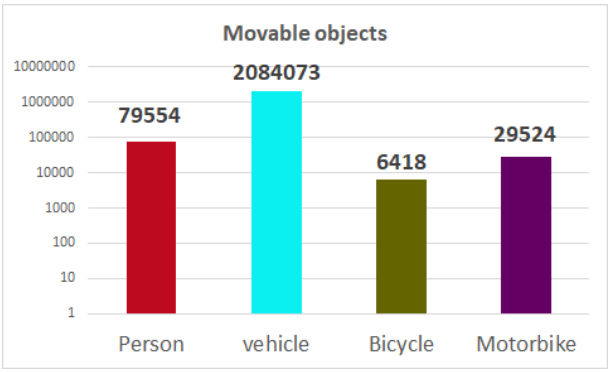
In Figure 2.17, we show statistics of the annotated 3D point clouds for each semantic class. We observe that the distribution of the annotated points w.r.t. to different semantic classes is unbalanced. For instance, the number of annotated points for the *building* class represent more than 47% of the entire dataset. In contrast, the *movable objects* category represent no more than 4% of the dataset. This is expected in a typical urban environment where the majority objects belong to the categories *surface* and *infrastructure*.

- **2D Annotations:** For 2D annotations, we used a web-based tool introduced in Tangseng *et al.* [80]. The online tool starts by computing SLIC [81] superpixels on the fly via a web browsing interface. The user can easily adjust the resolution of the over-segmentation to fit to the complexity of the annotated scene and subsequently associate labels to each segment.

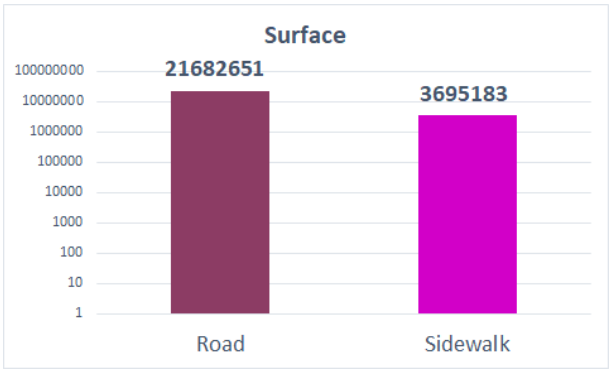
- **3D Annotations:** 3D manual annotation is considered by far a more cumbersome task than 2D annotations since there is no proper definition of occlusion and borders



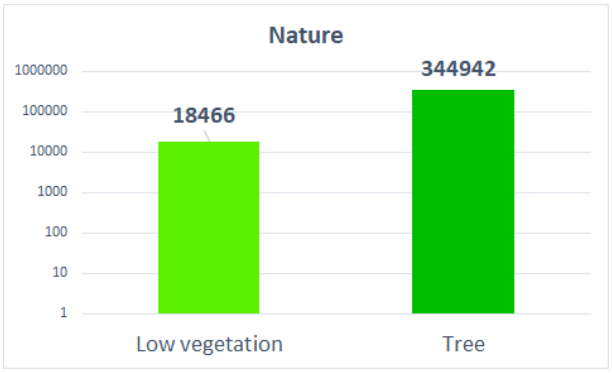
(a) Infrastructure



(b) Movable objects



(c) Surface



(d) Nature

Figure 2.17: *Statistics on the distribution of the annotated point clouds per-class*

of objects in the 3D space. Besides, 3D data has usually a very sparse topology. Traditionally, annotating 3D data can be performed manually as in Roynard *et al.* [48], or semi-automatically Vallet *et al.* [46]. Akin to [48], we used CloudCompare tool [27] to annotate 3D point clouds. To transfer these labels to the 3D mesh, we compute for each vertex in the mesh its first nearest neighborhood in the labeled point cloud. Afterwards, the label of each triangle will be a major vote between the labels of its 3 vertices. In case where each vertex of the same face has a different label, we attribute a miscellaneous label for that triangle.

In Table 2.6, we show the statistics of the ground truth 3D and 2D modalities in our dataset at the time of writing.

Modality	perspective RGB images	Spherical RGB images	Depth maps	Reflectance maps	3D point clouds	3D mesh faces
Statistics	750	150	4000	4000	53562109	106477878

Table 2.6: *Statistics on the annotated modalities in our dataset*

Compared to the performance of SPG achieved on benchmark datasets such as S3DIS [51], we observe a clear decrease in terms of mIoU. We believe that this stems from the imbalanced distribution of classes in our dataset. Several strategies exist to mitigate

[27] https://www.danielgm.net/cc/

this issue. However, in this study, we simply want to highlight the usefulness and the challenging nature of our data.

2.2.5.2 Evaluation

Given 3D per-point, per-face annotations along with 2D pixel-wise annotations of perspective/panoramic images with their camera pose information as well as omnidirectional ground truth depth maps, numerous computer vision tasks could be defined. In the current release, we mainly focus on 3D point clouds semantic segmentation, as well as 360° monocular depth estimation. For each task, we choose a method from the literature to evaluate on our dataset. The selected approaches are not necessarily the current state-of-the-art in the underlying task but they are known to be competitive.

- **Semantic segmentation of 3D point clouds:** To assess the task of 3D point cloud semantic segmentation on our dataset, we use the SuperPoint Graph algorithm (SPG) introduced by Landrieu and Simonovsky in [17]. We consider the Overall Accuracy (OA), and the Intersection over Union (IoU) defined w.r.t. to the confusion matrix CM of size $k \times k$ where $k = 14$ corresponds to the number of classes. A detailed definition of these metrics is provided in Chapter 4.
  Overall Accuracy: OA is a global metric defined as the ratio of correct predictions divided by the overall number of (annotated) points.

$$OA = \frac{Trace(CM)}{Sum(CM)} \tag{2.8}$$

  Class Intersection over Union IoU: this per-class metric is defined as the ratio between true positives divided by the sum of false positives, false negatives and true positives.

$$IoU_i = \frac{CM_{i,i}}{CM_{i,i} + \sum_{i \neq j}(CM_{i,j} + CM_{j,i})} \tag{2.9}$$

  We split the dataset into train and test subsets such that the test set represents nearly 20% of the overall dataset. We use the implementation of SPG [28] [17] with the default parameterization.
  In Table 2.7, we show the obtained OA, and IoU on our dataset. The detailed per-class IoU scores are presented in the appendix along with illustrations of semantic segmentation results.

Method	**OA**	IoU
SPG [17]	87.89	31.69

Table 2.7: *Semantic segmentation results on our dataset using the method of [17]*

- **Omnidirectional monocular depth estimation:** To investigate the usefulness of our dataset for 360° monocular depth estimation, we leverage the deep model proposed by Payen *et al.* in [65]. This network architecture is mainly based on the pioneering work of Godard *et al.* [82] for unsupervised monocular depth estimation. The network

[28] https://github.com/loicland/superpoint_graph

was retrained on the omnidirectional domain-adapted KITTI dataset [49] by projection and style transformation using CycleGAN [83]. We directly evaluate the pre-trained model on our test subset composed of 200 full-surrounding depth images. To ensure an equivalent evaluation setup as in [65], the maximum range depth is set to 50 meters. We also crop our panoramic images and their corresponding depth maps to get panoramas of size 2048×300. Following the evaluation protocol of Eigen *et al.* in [84], for each predicted depth map d in the test set T and its corresponding ground truth depth map d^*, we report the following scores for:

Depth accuracy:

$$\% \text{ of } d_i \text{ s.t. } max(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}) = \delta < threshold \tag{2.10}$$

Absolute relative difference:

$$\text{Abs.Rel} = \frac{1}{|T|} \sum_{d \in T} |d - d^*|/d^* \tag{2.11}$$

Squared relative difference:

$$\text{Sq.Rel} = \frac{1}{|T|} \sum_{d \in T} \frac{||d - d^*||^2}{d^*} \tag{2.12}$$

Root mean square error:

$$\text{RMSE} = \frac{1}{|T|} \sum_{d \in T} ||d_i - d_i^*||^2 \tag{2.13}$$

Logarithmic root mean square error:

$$\log \text{RMSE} = \frac{1}{|T|} \sum_{d \in T} ||\log(d_i) - \log(d_i^*)||^2 \tag{2.14}$$

Train data	**Test data**	**Abs.Rel**†	**Sq.Rel**†	**RMSE**†	**log RMSE**†	**Depth Acc.*** $\delta < 1.25$
KITTI	CARLA [60]	0.247	7.652	3.484	0.465	0.697
KITTI	Ours	**0.238**	**7.23**	**3.216**	0.484	0.654
KITTI.proj	CARLA [60]	0.251	7.381	3.451	0.445	0.732
KITTI.proj	Ours	**0.242**	**5.743**	**2.97**	**0.354**	**0.741**

Table 2.8: *Depth recovery quantitative results using the method of [65] trained on KITTI dataset [49] and tested on our own test set and synthetic data generated from [60]. (⋆): higher better, (†) lower better*

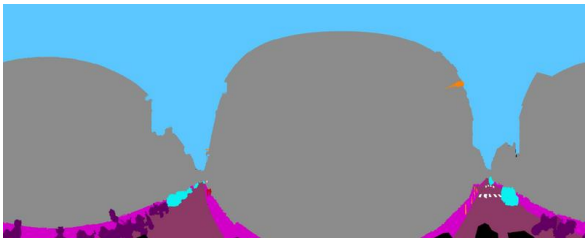
Due to the absence of a real world outdoor dataset which provides ground truth depth maps, it was not possible to perform a quantitative evaluation of depth recovery methods. Payen *et al.* [65] mitigate this by generating synthetic depth maps using CARLA simulator [60]. Despite we have a considerable number of panoramic depth maps (4000)

in our dataset, we are not able to perform training from scratch nor fine-tuning as the model requires rectified stereo pairs of equirectangular images as input. Since we do not have such set up in our dataset, the evaluation is based on a direct inference of the pre-trained model on our test set. In Table 2.8, we present the quantitative results of the depth recovery method of [65] trained on two different version of KITTI dataset [49] and tested on the proposed synthetic dataset as well as on our own dataset. KITTI in the first column of Table 2.8 denotes that the model was trained using the set of rectilinear perspective images from [49]. KITTI.proj means that the network is trained on the panoramic images which are the result of equirectangular projection of the rectilinear images from [49]. It should be noted that the network proposed in [65] was neither pre-trained nor fine-tuned on the simulated data [60]. Both the synthetic depth maps and our depth maps are used for dataset cross-validation in this experiment.

We observe that testing the model on our validation set gives competitive results compared to the simulated data, especially when the model is trained on equirectangular images (KITTI.proj). This confirms the usefulness of our computed depth maps.

In Figure 2.18, we show illustrations of equirectangular samples from our dataset – namely a panoramic image and the corresponding ground truth annotation, spherical depth and intensity maps. Figure 2.19 depicts a textured scene along with both 3D point clouds and 3D mesh.

(a) Panoramic image



(b) Ground truth annotation of the panoramic

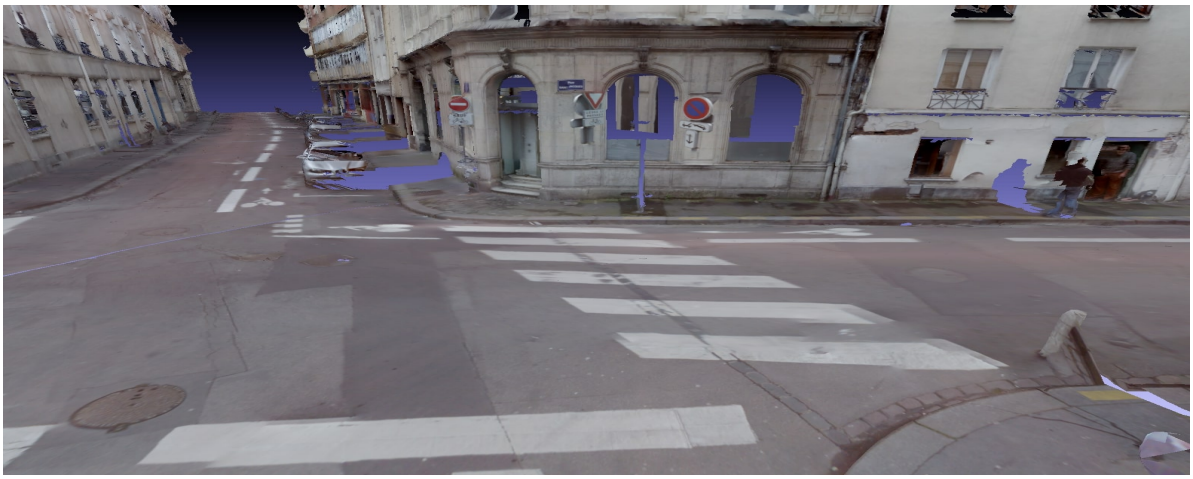
(c) Equirectangular depth map

(d) Equirectangular intensity map

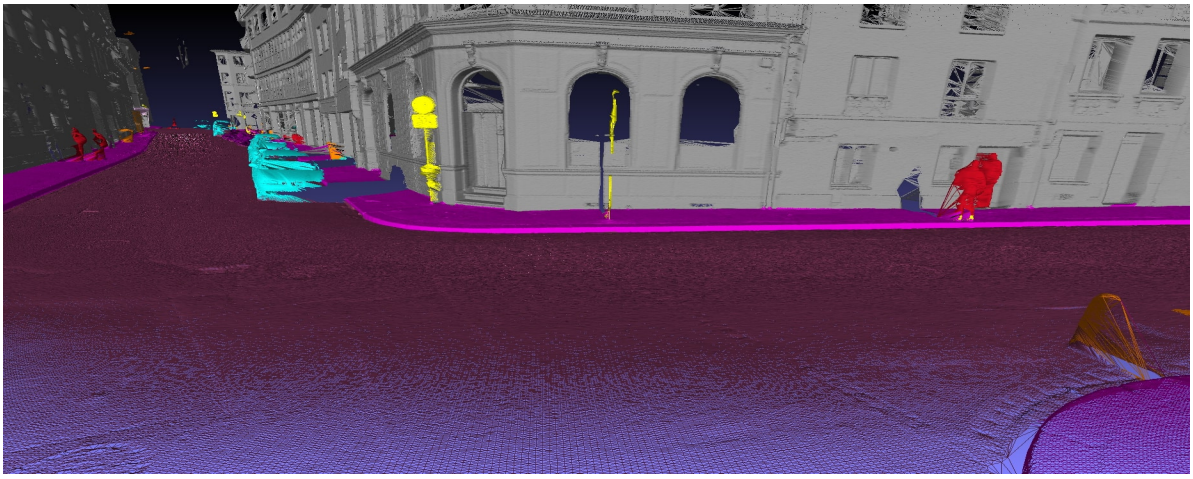
Figure 2.18: *Illustration of panoramic data in our dataset.*

2.3 Conclusion

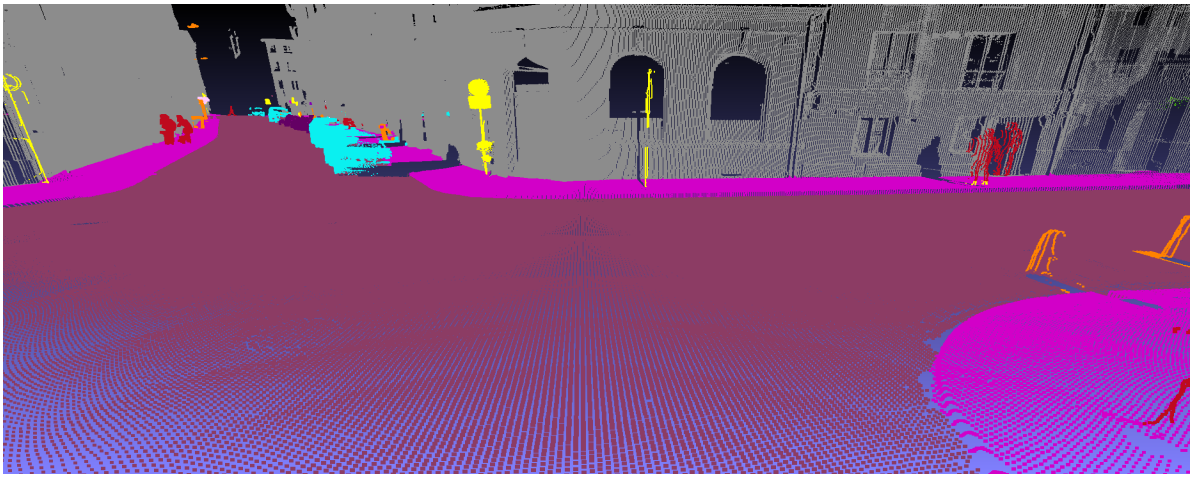
As an effort to facilitate the task of the computer vision and robotics communities in terms of easy and organized access to up-to-date information in order to keep truck of the recent outdoor urban datasets, we have conducted throughout this chapter an extensive investigation of the current state-of-the art datasets. This study allowed us to identify an urgent need of a multi-modal benchmark in which imaging modalities could be



(a) 3D textured mesh



(b) 3D mesh annotation



(c) 3D Point cloud annotation

Figure 2.19: *Illustration of the quality of annotations in our 3D multi-modal dataset*

assessed separately or jointly using a wide range of data-fusion-based methods to achieve accurate and robust semantic segmentation and monocular depth estimation. To narrow this gap, we made available to the community a large scale dataset of urban environments

recorded in the region of Rouen, France using a mobile mapping vehicle. This hybrid dataset can serve not only for the infrastructure management and monitoring but also for applications related to autonomous driving context. At the time of writing, our dataset contains 4000 ground truth 360° depth maps, 4000 full surrounding LiDAR intensity maps, 150 annotated high resolution spherical RGB, 750 perspective images with their pose information, more than 53 million point-wise and instance-wise annotated 3D points and nearly 1.5 kilometers of per-face annotated 3D textured mesh resulted in more than 106 million triangles. For now the released subset represents only around 10% of the full acquisition. Our intention is to gradually release the rest of the annotated data to the community. Another acquisition in the same location has been conducted in 2019 three years after the initial one, albeit this time it will be leveraged to evaluate the task of 3D change detection. The source code [29] to generate spherical RGB images, depth and reflectivity in equirectangular projection is now part of an open source library called LibOri developed at LASTIG laboratory and available online.

In the next chapter, we discuss the process of generating 3D geolocalized textured meshes from the collected mobile mapping data.

[29] https://github.com/IGNF/libOri

Chapter

3 Large scale textured mesh reconstruction

Chapter content

Abstract

In the previous chapter, we presented a new multi-modal dataset for outdoor scene understanding. We explained in details the process of generating depth and intensity maps as well as 2D panoramic images and 3D point clouds along with their semantic ground truth. In this chapter, we first introduce a novel representation of the space as a 3D textured mesh which is suitable for mobile robots navigation. Second, we explain the process of reconstructing this 3D map. Finally, we evaluate and discuss the full pipeline.

Introduction

To be able to plan a path and navigate towards a target, an autonomous agent, being a robot or a driver-less car, has to apprehend the operating environment the way a human does. Such ability cannot be acquired without a high-level abstraction of the surrounding space along with tools for robot-human communication. A first step towards achieving this goal is mapping. Building a 3D map of the surrounding environment is a critical component of indoor/outdoor perception and navigation. Historically, localization and navigation within a map is performed using Simultaneous Localization And Mapping (SLAM) techniques [85–87]. These methods have spawned a considerable and an ongoing body of work regarding the appropriate geometric modeling of explored environments. As a result, several approaches have been introduced to answer the question of what is the most suitable representation in terms of fidelity and robustness guarantying a reliable navigation.

In the literature [88–90], we can find two fundamental representations of a 3D map: a metric map and a topological map. In the former type of maps, the objects of the environment are represented by their 3D coordinates in an absolute frame, whereas in the latter type (*i.e.* topological) the map is abstracted as a graph where the nodes represent special entities (*e.g.* co-visibility, distance) and the edges express the accessibility between the different nodes. Topological maps are by far less precise than metric maps as the objects of the environments are substantially abstracted to a high level representation (a graph) which impact the precision of the localization and navigation process. As discussed earlier in the introduction chapter, the goal here is to build an off-line 3D map that encapsulates as much information as possible about the explored space. Therefore, in this study we are only concerned with the approaches which represent a 3D map as a metric map.

Among the different types of metric maps, we can mainly distinguish between four families of representations each of which relies on a different geometry based on landmarks [8, 91–94], surfel/point clouds [9, 95–99], occupancy-grids/voxels [10, 100–104] and surface meshes [12, 105–109].

From the autonomous driving perspective, the first layer of a high definition map (HD map) being the geometric layer, must be able to efficiently and faithfully represent the surrounding world at an unprecedented centimeter resolution. That is, navigable zones, sidewalk curbs and boundaries, road slopes, sharp edges as well as surface deformations have to be precisely reconstructed with the lowest possible rate of noise and artifacts. We argue that one of the most appropriate representation among the aforementioned ones is the mesh-based approach where the operating space is represented as an explicit triangulated surface. 3D triangulated surfaces are a widely used and extensively studied concept since the early stages of computer vision [110, 111]. Their ability to reproduce the required high-level of details while remaining efficient in handling massive data has been successfully proven in many occasions [112, 113]. In the context of urban mobile mapping, most of these mesh-based maps are built using either multiple images from different views or raw LiDAR point clouds [114]. However, since the recent acquisition platforms are often equipped with LiDAR and RGB cameras, it would be a waste if these two modalities remain essentially exploited independently. As a matter of fact, a joint exploitation would benefit from the high complementarity of these two sources of information:

- High resolution of the images *v.s* high precision of the LiDAR range measurements.
- Passive RGB images VS active intensity measurements in near infrared.
- Different acquisition geometries.

A straightforward way to combine both sources of information consists in projecting RGB images onto the 3D mesh reconstructed from the LiDAR point clouds such that each vertex has a color attribute. Nonetheless, per-vertex color encoding means that the radiometric information is mainly tied to the mesh resolution which results in a photometrically poor representation in case of sparse triangulation of the surface. Therefore, a more compact and convincing fusion strategy of RGB images and LiDAR point clouds into a single representation is required. To this end, we propose to use a popular concept within the computer graphics community in order to represent a 3D map of the environment:*textured meshes*. Being the central representation of virtual scenes, textured meshes have been mainly leveraged in video games and animation movies industry where graphics cards are highly optimized for their processing and visualization.

While 3D textured meshes are a common representation in controlled indoor environments [22, 24] thanks to hand-held consumer-grade devices such as *Matterport* [1] which systematically generates a 3D textured mesh using the collected RGB-D data, this remains a very challenging task for outdoor scenes due to numerous difficulties. First, acquisition constraints are more complicated in the case of urban mapping due to its dynamic nature and extremely changing lighting conditions. Second, a confusing trade-off between quality of the reconstructed textured scene and the scalability of the used method has to be always taken into consideration. Finally, akin to the indoor case, a full automatic pipeline that does not involve the end-user interaction is not guaranteed. At the time of writing our paper, a similar work to ours Romanoni *et al.* [109] was published corroborating the growing importance of such maps in which authors propose to represent the 3D map of an urban environment as a textured mesh. While their method handle large scale scenes, this is accomplished at the expense of the output quality and the level of automation. The produced textured mesh is a low resolution representation where fine details were discarded failing, thereby, to satisfy the requirements of modern HD maps. The latter method will be discussed in detail in the related work section.

In this chapter, we present a visibility consistent 3D mapping pipeline to *automatically* reconstruct *high quality large scale* urban textured mesh using both oriented images and geo-referenced point clouds coming from a terrestrial Mobile Mapping System (MMS). In Section 3.1, we give an overview of the work related to the different blocks of our framework. A simple yet fast and scalable algorithm for surface reconstruction is presented in Section 3.2. The texture mapping approach is explained in Section 3.3. Finally, in Section 3.4 we expose our experimental results and conclude the chapter in Section 3.5.

3.1 Overview

In this section, we first start by discussing the main representations of a 3D scene used for mapping and navigation purposes. Second, we present the previous work related to the

[1] https://matterport.com

textured mesh representation taking a broad perspective across computer vision, robotics, computer graphics and photogrammetry.

3.1.1 3D metric map representations

In autonomous navigation, a 3D scene can be represented by various geometries. In the literature, these representations can be grouped into four categories based on the geometry of elements constituting the scene, namely, sparse landmarks, dense surfels [2] or point clouds, occupancy grids or voxels and triangulated meshes. Other interesting representations were also proposed such as elevation/plane-based maps. In Figure 3.1 we show some of these representations (landmark-based, surfel/point-based, voxel-based and 2.5D elevation-based. In table 3.1, we briefly report the advantages and drawbacks of each of the discussed representations.

3.1.1.1 Landmarks-based representations [8, 91–94]:

The scene is reconstructed using a set of sparse and distinguishable landmarks corresponding to discriminative primitives (*e.g.* corners, segments, lines) with known absolute pose. Whilst such representation has been investigated since the early stages of localization and mapping as it allows for a fast processing, it remains a low-resolution representation of the 3D geometry. In practice, the sparsity feature is unsuitable for obstacle avoidance not to mention visualization and rendering.

3.1.1.2 Surfel/Point clouds-based representations [9, 95–99]:

In contrast to the latter representation, dense representations which model the 3D scene by means of point clouds or *surfels* are widely used. Thanks to their density, these representations offer rich and visually ergonomic models. However, their unstructured nature in addition to their low level modeling capabilities easily affected by noise make them neglect the topology and boundary of objects in the scene.

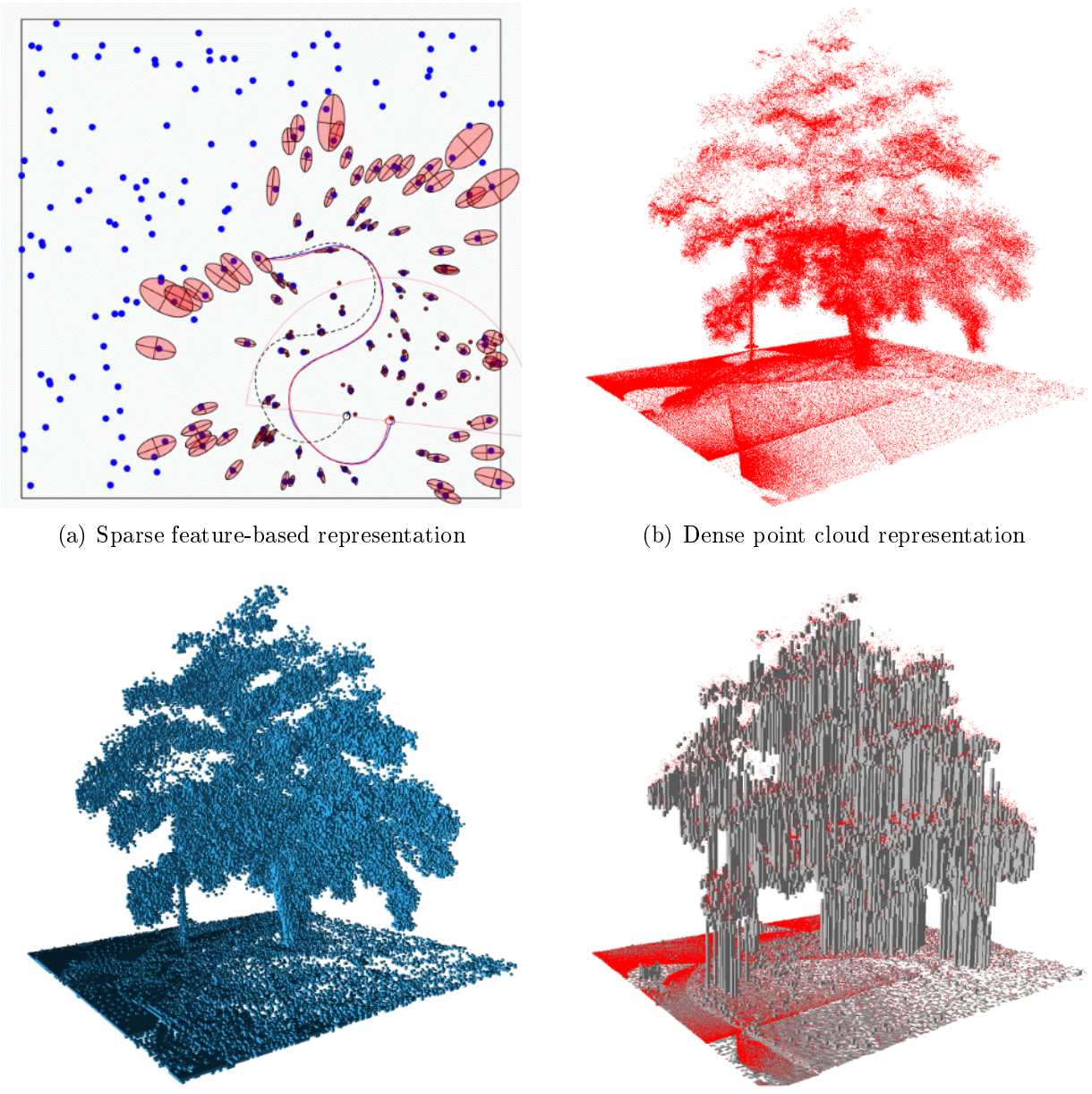
3.1.1.3 Occupancy-grid/voxel-based representations [10, 100–104]:

Alternatively occupancy-grid/voxel-based representations have gained popularity in the few past years. In this case, the 3D scene is decomposed into cubes (*voxels*) arranged in a regular grid or in an adaptive octree where each node stores the binary occupancy value (occupied, empty) or the distance to the surface commonly referred to as the Signed Distance Function (SDF), or more recently the Truncated Signed Distance Function (TSDF). Despite their versatility and constant time access, voxels or occupancy grids are prone to erroneous discretization and heavy memory footprint as the entire map has to be allocated in memory.

3.1.1.4 Surface mesh-based representations[12, 105–109]:

A more efficient spatial-partionning strategy which has been shown to be more compact consists of surface mesh models where the scene is a set of vertices connected by edges forming triangles. This representation has the advantage of being a continuous geometry

[2] a set of small surface elements (usually discs as in [9]) which store local statistics of 3D points (*e.g.* mean, covariance) and orientation.



(a) Sparse feature-based representation

(b) Dense point cloud representation

(c) Voxel-based representation

(d) 2.5D elevation-based representation

Figure 3.1: *Illustration of various 3D representations used for mapping*

offering not only an explicit surface deformation modeling, but also a well defined objects' boundaries and most importantly its ability to efficiently represent and smoothly render unbounded extended areas with well-established graphics pipelines.

3.1.1.5 Other representations [11, 115]:

In addition to the aforementioned representations, other types of maps were also used. For instance, 2.5D elevation maps [11] consisting of 2D grids where each cell stores an estimated local height of the scene. While such maps allows for fast constant time access, robots are not able to distinguish between free or unknown space when navigating making it inappropriate for obstacle avoidance. Plane-based maps [115] were also introduced. In this case, entities of the scene are approximated by simple planes. While such

plane-based mapping approaches have demonstrated a faster and more stable optimization during navigation, its expressiveness and modeling abilities is limited compared to dense representations.

Representation	References	Advantages	Drawbacks
Landmarks-based	[8, 91–94]	- a light-weight representation - Fast and efficient processing	- Sparse representation -Unsuitable for obstacle avoidance
Surfel/points-based	[9, 95–99]	- Dense/detailed representation - Subsampling without loosing details	- Unstructured representation - Topology and boundary of objects are neglected
Voxel-based	[10, 100–104]	- Versatility - Constant time access	- Discretization errors - Heavy memory requirement
Mesh-based	[12, 105–109]	- Explicit surface deformation modeling - Well defined object boundaries	- Handling loop closure is not straightforward
Elevation-based	[11]	- Fast and constant time access - Probabilistic height estimation	- Unsuitable for obstacle avoidance - 2.5D representation
Plane-based	[115]	- Fast and stable optimization during navigation	- Low level expressiveness and modeling abilities

Table 3.1: *Comparison between different representations of a 3D scene*

3.1.2 Textured meshes representation:

Recently, textured meshes are gaining more and more attention in the geospatial industry as Digital Elevation Models (DEMs) coupled with orthophotos, which were well adapted for vertical high altitude airborne and space-borne acquisition. However, such representation is not suited for the newer means of acquisition: closer range platforms (drones, terrestrial mobile mapping vehicles) and oblique imagery requiring a finer level of details and a flexibility in handling scalability issues. A key component in generating textured meshes is reconstructing a surface from 3D point clouds. Depending on the mean of acquisition, 3D point clouds can be directly extracted by LiDAR acquisition during surveying or indirectly using image-based techniques [114].

As shown in the previous chapter, our mobile mapping platform provides two imaging modalities, namely, RGB geolocalized images and LiDAR point clouds. Since the configuration of the acquired RGB images allows to generate a dense 3D point cloud using Multi-View-Stereo (MVS) techniques, we are facing two choices regarding which type of point cloud to utilize in order to reconstruct the surface; the one derived from MVS or the raw point clouds collected using the LiDAR. To answer this question, we conducted a comparative study in which we discuss image-based and LiDAR-based point clouds in terms of density, accuracy, cost and sensitivity to external factors (illumination, surface materials).

3.1.2.1 Image-derived point clouds

These are a set of methods that rely on 2D images in order to generate indirectly 3D point clouds. Inspired by the way a human perceive objects, stereo vision relies on multiple images (at least two) to restore the depth information using photogrammetry principles. MVS reconstruction methods [116] have reached a high level of maturity making it possible even for city scale reconstruction [117, 118]. The proliferation of these methods has been further boosted thanks to the democratization of the acquisition low cost devices. However, since 2D images are always acquired by passive sensors (electro-optical RGB cameras), the quality and precision of the reconstructed scene is influenced by illumination variation and textureless objects (*e.g.* water, snow).

3.1.2.2 LiDAR point clouds

Light Detection And Ranging (LiDAR) is an active surveying technique that uses the time between the emission of laser pulse and its reception on the sensor after reflection by the scene to measure the distance to the scanned objects. Depending on the surveying technique, density and resolution varies dramatically between $10 pts/m^2$ for Aerial LiDAR Scanner (ALS) to around $1000 pts/m^2$ for Mobile LiDAR Scanner (MLS) [119]. While such devices allow for a robust acquisition of accurate 3D point clouds regardless the lighting condition, it may contain noise and artifacts because of the scanned speculate surfaces (*e.g.* mirror, windows).

As depicted in Table 3.2, we think that the LiDAR 3D point clouds are more suitable for our mapping application as it provides high accurate georeferenced 3D points without strong assumptions neither about the accuracy of cameras calibration nor about the precision of matching algorithms and image quality as in the case of MVS.

Origin	Density	Accuracy	Advantages	Drawbacks
Image-based	Depends on the spatial resolution of the multi-view images	Depends on the camera model accuracy, matching algorithms, image quality and stereo angle	Low cost; available RGB color	Precision depends on several factors results affected by illumination
LiDAR-based	High density closer the distance, higher the density	High accuracy ($1cm$)	Active intensity measurements; robust to illumination variation; High precision is guaranteed	Expensive devices speculate surfaces results in scanning artifacts

Table 3.2: *Comparison between Image-derived point clouds and MLS point clouds*

Figure 3.2 shows the proposed framework to reconstruct the textured mesh. Our approach is two fold: First we start by reconstructing a triangulated surface from the raw LiDAR point clouds. Second, after filtering and repairing the resulted 3D mesh, the registered oriented images are subsequently mapped to the reconstructed surface by selecting for each triangle which is the best view. In the following, we review the related work to our two-step approach (surface reconstruction and texture mapping) for reconstructing the 3D textured map.

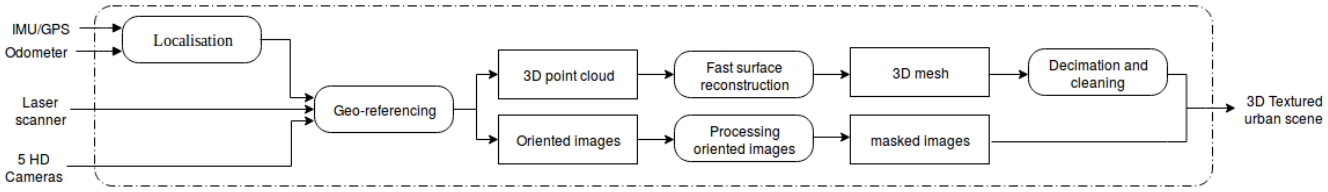


Figure 3.2: *3D Textured map reconstruction pipeline*

3.1.2.3 Surface reconstruction: a review

Surface reconstruction is a well established research area that has been extensively investigated during the two last decades [110, 111]. As the main contribution of this chapter is not a new surface reconstruction algorithm, we will settle for a brief review of the related work especially methods used in evaluation. Following the nomenclature proposed by Berger *et al.* [110], surface reconstruction problem can be essentially grouped into two major categories with respect to their type of smoothness prior: *Global* (Indicator function [120, 121], Radial Basis Function [122]) and *local* methods (point-set surface [123–125], Multilevel Partition of Unity (MPU) [126, 127]). Other popular surface reconstruction algorithms exist, but they are out of the scope of this study.

- **Radial Basis Function** [122]: In this class of methods, the surface is reconstructed as a zero level set [3] of a signed scalar field Φ estimated using a well-known method originally designed for scattered data interpolation called Radial Basis Functions (RBFs). Using a linear combination of radially symmetric functions RBFs, the implicit surface can be defined as:

$$\Phi(x) = \sum_i w_i \phi(||x - x_i||) + p(x) \tag{3.1}$$

  where $p(x)$ is a low degree polynomial that improves the extrapolation capability of RBFs centered at nodes $x_i \in \mathbb{R}^3$. One way to find the coefficients w_i consists in imposing interpolation constraints satisfying the definition of an implicit surface $\Phi = 0$. However by doing so, this will lead to the trivial solution given by the identical zero function. That is why [122] proposed to use additional constraints based on off-surface points defined by the normals to the surface as illustrated in Figure 3.3. RBFs-based reconstruction methods have the advantage of producing a globally smooth surface. Moreover in the case of missing data and non-uniform sampling, the extrapolation power of RBFs ensure a watertight surface. However, since the implicit function is based on interpolation constraints, the reconstructed surface depicts topological artifacts when inconsistent off-surface points are provided due to bad normal estimation in case of non oriented point clouds.

- **Indicator function** [120, 121] : These set of methods estimate the surface using a labeling that discriminates the interior and exterior of a solid shape called indicator or scalar function. Operating exclusively on oriented [4] point clouds, an implicit function χ is computed by ensuring that the gradient of the scalar function is aligned with the normal field N. An illustration of the approach is showed in Figure 3.4.

[3] S is a zero level set of a signed distance function $d \Rightarrow S = \{x |\ d(x) = 0\}$
[4] with available normal information

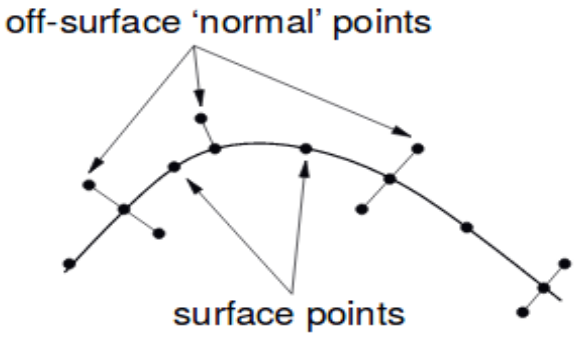


Figure 3.3: *Illustration of off-surface points. Image from [122]*

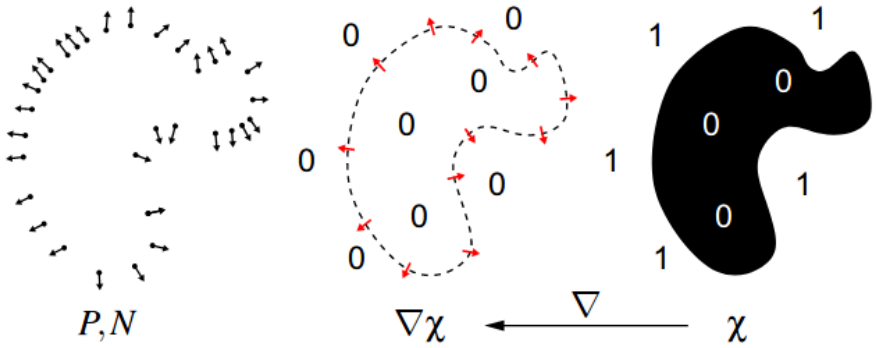


Figure 3.4: *The gradient of the indicator function χ is connected to the normal field N of the point clouds. Image from [111]*

Formally this can be solved by minimizing the following equation:

$$argmin_{\chi} \int ||\nabla\chi(x) - N(x)||_2^2 dx \tag{3.2}$$

The surface is subsequently generated by isocontouring. Popular methods of this category are *Poisson* [120] and its extension the *Screened Poisson* [121] algorithms. Whilst this gradient-based formulation ensure the robustness of reconstruction to data imperfections such as non-uniform sampling, noise and outliers, fitting directly the gradient of a scalar function may result in an undesirable over-smoothing [120] or over-fitting [121]. In addition, since these methods are based on normal orientation, in case of a densely distributed normal flips (especially surface details), this category of methods fail to faithfully reconstruct the surface.

- **Point-set surface** [123–125]: This class of methods define a smooth surface as a set of stationary points [5] computed using a projection operator of points in the ambient space or as an implicit surface based on Moving Least Square (MLS) [128]. The first work of Levin [123] starts by fetching for each point in the ambient space its closest projection in the 3D point cloud such that its normal goes through its projection. Figure 3.5, illustrates the problem setting. Afterwards, the approximation order is improved using a controllable polynomial fit making the approach will suited for noise filtering. A subsequent work of Amenta and Kil [124] has shown that

[5] a point is considered stationary if its projection is the identity $x = P(x)$

the polynomial fitting step can be omitted to improve efficiency while producing the same results by fitting the best plane defined by a weighted average of local neighborhood normals. The surface is subsequently defined as a scalar field equal to the distance between evaluation point and the prescribed fitted plane. However the plane fitting operation is subject to instability in high curvature regions with low sampling rate inducing, thereby, errors in the final reconstruction. To alleviate this problem Guennebaud and Gross [125] propose a higher order fitting strategy based on an algebraic sphere. In this way stability is significantly improved where plane-fitting based methods fail.

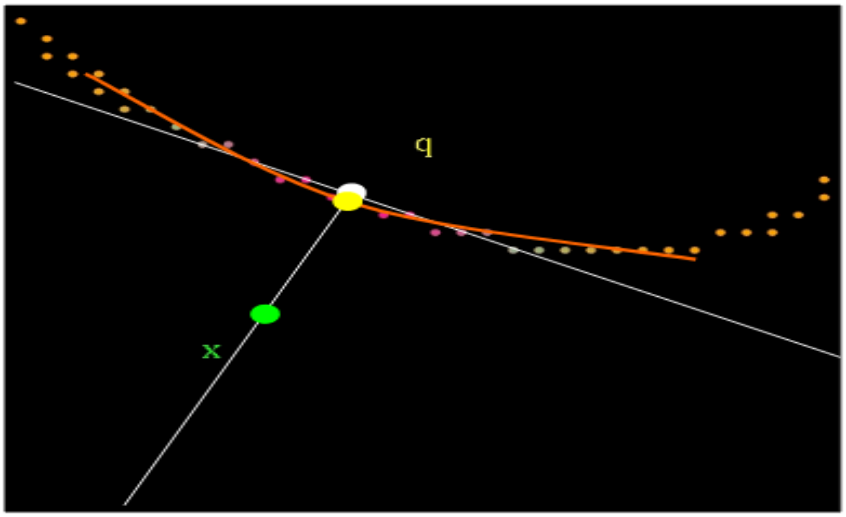


Figure 3.5: *q is the projection of the evaluation point x. The orange curve is the polynomial fit [123]. It is replaced by a planar fit in [124] and an algebraic sphere in [125]*

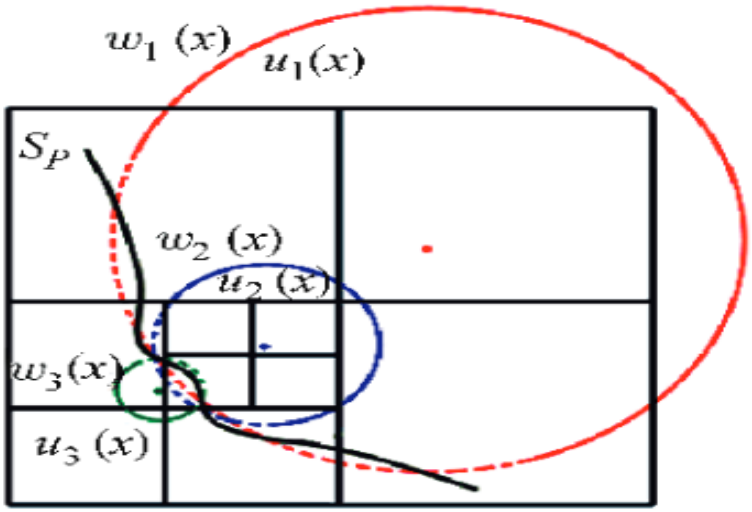


Figure 3.6: *The algebraic spheres (blue, red and green) are fitted per cell in [127] instead of per-point as in [125]. Weighted distance fields $(w_1(x).u_1(x))$ and $(w_2(x).u_2(x), w_3(x).u_3(x))$ are subsequently blended to give the final implicit surface represented by the black curve S_p. Image from [127].*

- **Multi-level partition of unity** [126, 127]: This set of techniques is based on a hierarchical partitioning. Ohtake *et al.* [126] first build an octree on top of the point

cloud. A set of local distance fields are subsequently defined in such a way the points in each cell and nearby are approximated by a bi-variate quadratic polynomial. The same procedure is recursively repeated for each cell in case of a large residual error. Merging these per-cell distance fields results in a smoothly global implicit surface. Inspired by the success of algebraic high order primitive fitting approach [125], Xiao [127] propose to use the same hierarchical partitioning as in [126], albeit, per-cell points are instead approximated by algebraic spheres. While MPU-based approaches are known to be faster and more efficient compared to Point-set surface based methods, they fail to address point clouds with missing data using extrapolation before fusing local approximated functions. In Figure 3.6 we show an illustration of the Multi-level partition of unity reconstruction.

3.1.2.4 Texture mapping: a review

From the computer graphics perspective, texturing a 3D mesh is typically a two-step approach. First for each triangle in the 3D surface, we need to pick up the best view to be used as a preliminary texture. Second, to minimize seams between adjacent texture patches, the mapping resulted from step one has to be locally and globally optimized for photo-metric consistency. A large body of work has been achieved to solve the first step, coined as *the view selection problem*. These methods can be classified into two groups:

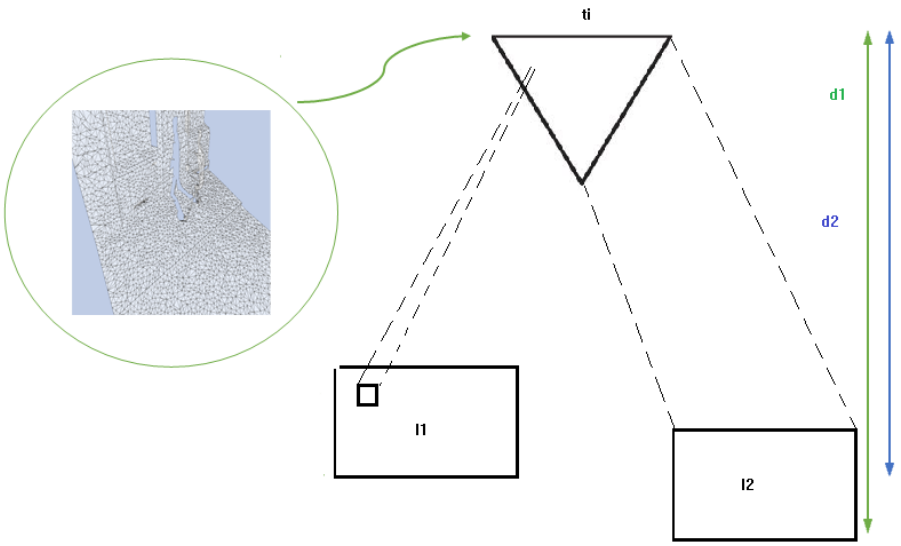


Figure 3.7: *Both images I_1, I_2 are in the field of view of triangle t_i at different respective distances d_1, d_2 $(d_1 < d_2)$ such that only few pixels in I_1 are visible to t_i while the distant view I_2 is entirely visible to t_i.*

- **Blending-based selection [129, 130]:** In this category of methods, for each face in the 3D mesh, multiple views are selected and subsequently blended to achieve a seamless texture across the borders of patches. While such fusion strategy ensures consistency of adjacent texture patches to a certain extent, it comes with several shortcomings. If the reconstructed geometry is slightly inaccurate, which is almost

always the case, texture patches will be misaligned resulting in visible seams in the final model. Moreover, during a mapping survey, most of the oriented images are collected at different distances to the scanned objects in the scene. As illustrated in Figure 3.7, when blending these views together the distant ones will blur out the closest resulting in an over-smoothed texture depicting ghosting effects.

- **Best view selection [131–133]:** In contrast to the blending-based view selection, these methods propose to exclusively select one view per-triangle. This strategy has the advantage of being faster compared to blending-based techniques. Besides, only borders of texture patches are concerned by a radiometric adjustment in opposition to blending where this step has to be locally performed for each triangle. Hence, in our application, we abstain from blending and we choose the per-face best view selection category.

  We note that few attempts proposed hybrid approaches [134] where they select a single view per-face then blend at borders of patches.

3.2 Surface reconstruction

As discussed above, surface reconstruction has been extensively studied from the computer graphics perspective Berger *et al.* [111] where the input point clouds are basically limited in size and most of the proposed methods do not cope with the level of challenges a mobile mapping acquisition imposes. In practice, most of the proposed algorithms do not handle typical mobile mapping acquisitions characterized with their huge size (city scale), particular noise and artifacts, miss-aligned scans, occluded objects, and extremely varying density. With the recent advances in autonomous navigation, the interest of reconstructing a surface out of a mobile mapping acquisition has gained an increasing attention over the few past years [109, 135, 136]. Older studies [137, 138] have been also conducted to solve this problem. For instance, Brun *et al.* [137] propose to use an incremental 2D Delaunay triangulation of a 2D parametrization of the 3D points in a cylindrical coordinate system. Even though this approach is fast and efficient enough to handle large scale data, the produced 3D mesh is not only over-smoothed (objects like cars and sidewalk are barely distinguishable), but also it depicts elongated triangles and isolated pieces. In the work of Carlberg *et al.* [138], a nearest neighbor approach is adopted to identify adjacent points on consecutive line scans, then edges of triangles are incrementally extended with respect to their chronological order until detecting a discontinuity fixed by a threshold. While several artifacts were efficiently handled such as holes and redundant triangles, other common acquisition challenges remain unsolved especially varying density and line scans intersections (when the vehicle follow sharp turns).

More sophisticated approaches have been recently introduced. The closest work to ours Romanoni *et al.* [109] propose a LiDAR based method that partitions the space into a set of tetrahedra which are subsequently classified according to visibility rays as occupied or free tetrahedron. The boundary between these two classes is then the final surface mesh. However, prior to the visibility consistency-check step, moving objects and cars need to be explicitly removed otherwise the final surface will contain noise and artifacts due to transparent objects traversed by the LiDAR beams. Roldao *et al.* [135] start by approximating local planar surfaces using an adaptive neighborhood. A global implicit surface is

reconstructed afterwards by computing the Truncated Signed Distance Function (TSDF) from the confidences of the statistical distribution stored in a voxel representation build on top of the LiDAR acquisition. The explicit surface is finally extracted using Marching cubes [139]. While the adaptive neighborhood kernel make the method robust to noise and varying density of the LiDAR point clouds, a trade-off between density and accuracy of the reconstruction, which is tied to the size of the neighborhood kernel, needs to be considered for each dataset.

In summary, we believe that the inherent topology of a mobile LiDAR sensor provides pertinent information regarding the adjacency relationship between 3D points that is most of the time ignored. In our work, we build on top of the methods of Brun *et al.* [137] and Carlberg *et al.* [138] which exploit the sensor topology. We propose a scalable and fully automatic surface reconstruction method out of a mobile mapping acquisition based on LiDAR measurements projected into the *Sensor space topology*.

3.2.1 Sensor topology

The sensor space topology was firstly introduced by [140] as an intermediate 2D representation of LiDAR acquisition that serves for semi-automatic annotation of 3D point clouds. This spatial geometry originates from the particular LiDAR configuration ensuring a constant time interval not only between consecutive emitted pulses but also between each rotation. Such parameterization allows the recovery of a regular topology out of the point cloud stream in such a way for an emitted pulse at time t the neighboring pulses are the immediately preceding and succeeding pulses respectively at time $t - \nabla t$ and $t + \nabla t$ and the closest ones on the preceding and succeeding rotations. Thanks to the continuous sampling of the used scanner, the last pulse of each rotation is connected to the first pulse in the following one. In our application we make use of this particular topology to extract a dense mesh out of the acquired 3D point cloud. This process is explained in details in the following section.

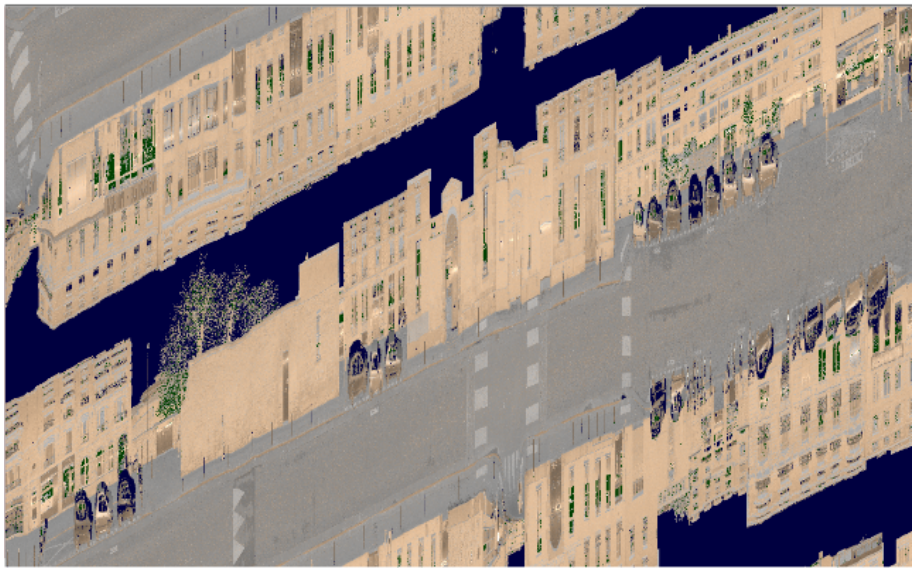


Figure 3.8: *LiDAR acquisition viewed in sensor space: vertical axis represent the rotation angle θ while the horizontal axis corresponds to time t. Image from [140]*

3.2.2 Mesh extraction

During urban mapping, the mobile platform may stop for a moment because of external factors (*e.g.* road sign, red light, traffic congestion ...) which results in massive redundant data at the same scanned location. Thus, a filtering step is mandatory to get a uniform distribution of scan-lines. To do so, we fix a minimum distance between two successive line scans and we remove all lines whose distances to the previous (unremoved) line is less than a fixed threshold. In practice, we use a threshold of $1cm$, close to the LiDAR accuracy.

Once the regular sampling is done, we consider the resulting point cloud in the afore-described sensor space where one dimension is the acquisition time t and the other is the rotation angle θ. Let θ_i be the angle of the i^{th} pulse and E_i the corresponding echo. In case of multiple echoes, E_i is defined as the last (furthest) one, and in case of no return, E_i does not exist so we do not build any triangle based on it. In general, the number N_p of pulses for a 2π rotation is not an integer so E_i has six neighbors E_{i-1}, E_{i+1}, E_{i-n}, E_{i-n-1}, E_{i+n}, E_{i+n+1} where $n = \lfloor N_p \rfloor$ is the integer part of N_p. These six neighbors allow to build six triangles.

In practice, we avoid creating the same triangle more than once by creating for each echo E_i the two triangles it forms with echoes of greater indexes: E_i, E_{i+n}, E_{i+n+1} and E_i, E_{i+n+1}, E_{i+1} (if the three echoes exist) as illustrated in Figure 3.9. This allows the algorithm to incrementally and quickly build a triangulated surface based on the input points of the scans. In practice, the (non integer) number of pulses N_p emitted during a 360 deg rotation of the scanner may slightly vary. To ensure robustness, we check if $\theta_{i+n} < \theta_i < \theta_{i+n+1}$ and if it doesn't, we increase or decrease n until it does. This was made possible and convenient thanks to not only the geometry of acquisition but also to the constant timestamps between each emitted pulse and each rotation of the LiDAR.

3.2.3 Cleaning

The triangulation of 3D measurements from a mobile mapping system usually comes with several imperfections such as elongated triangles, noisy unreferenced vertices, holes in the model, redundant triangles to mention a few. In this section, we focus on three main issues that frequently occur with mobile terrestrial systems and affect significantly the texturing results if not adequately dealt with, namely, elongated triangles, isolated pieces and holes.

3.2.3.1 Elongated triangles filtering

In practice, neighboring echoes in sensor topology might belong to different objects at different distances. This generates very elongated triangles connecting two objects (or an object and its background). Such elongated triangles might also occur when the MMS follows a sharp turn. We filter them out by applying a threshold on the maximum length of an edge before creating a triangle, experimentally set to $0.5m$ for the data used in this study.

3.2.3.2 Isolated pieces removal

In contrast with camera and eyes that captures light from external sources, the LiDAR scanner is an active sensor that emits light itself. This results in measurements that are

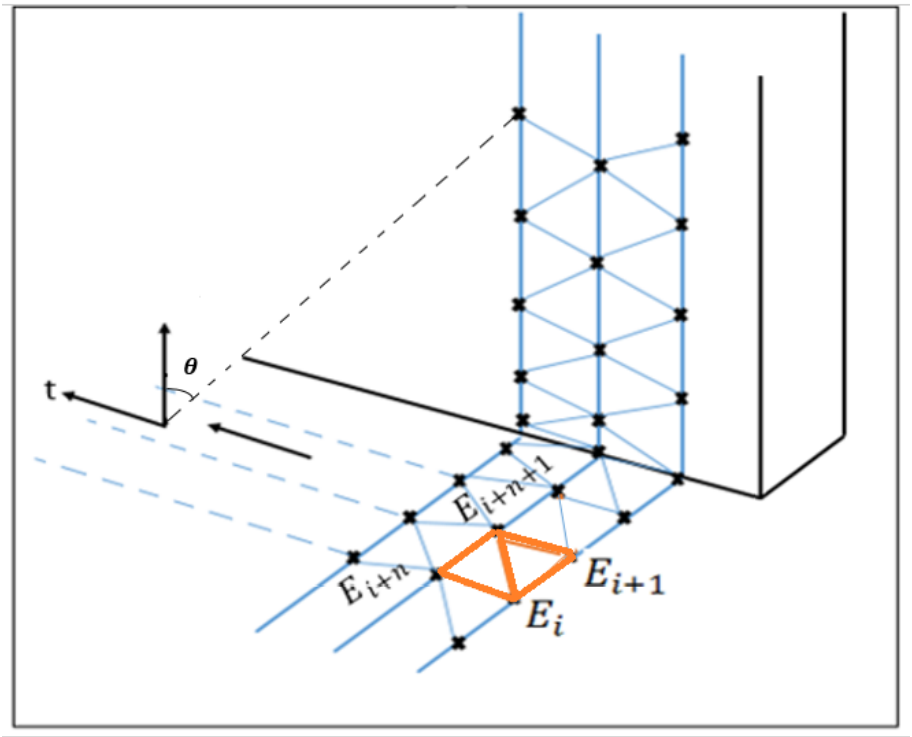


Figure 3.9: *Triangulation based on the sensor space topology*

dependent on the transparency of the scanned objects which cause a problem in the case of semitransparent faces such as windows and front glass. The laser beam will traverse these objects, creating isolated pieces behind them in the final mesh. To tackle this problem, isolated connected components composed by a limited number of triangles and whose diameter is smaller than a user-defined threshold (set experimentally) are automatically deleted from the final model.

3.2.3.3 Hole filling

After the surface reconstruction process, the resulting mesh may still contain a consequent number of holes due to speculate surfaces deflecting the LiDAR beam, occlusions and the non-uniform motion of the acquisition vehicle. To overcome this problem we use the recursive hole filling method introduced in [141]. The algorithm takes a user-defined parameter which consists of the maximum hole size in terms of number of edges and closes the hole in a recursive fashion by splitting it until it gets a hole composed exactly with 3 edges and fills it with the corresponding triangle.

3.2.4 Scalability

The interest in mobile mapping techniques has been increasing over the past decade as it allows the collection of dense, accurate and detailed data at the scale of an entire city with a high productivity. However, processing such data is limited by various difficulties specific to this type of acquisition especially the very high data volume (up to 1 TB by day of acquisition [142]) which requires efficient processing tools in terms of number of operations and memory footprint. In order to perform an automatic surface reconstruction

over large distances, memory constraints and scalability issues must be addressed. To do so, we adopt a slicing-based strategy. First, the raw LiDAR scans are sliced into N chunks of $10s$ of acquisition which corresponds to nearly 3 million points per chunk. Each recorded point cloud (chunk) is processed separately as explained in the work-flow of our pipeline presented in Figure 3.10, allowing a parallel processing and faster production. Yet, whereas the aforementioned filtering steps alleviate the size of the processed chunks, the resulting models remain unnecessarily heavy as flat surfaces (road, walls) may be represented by a very large number of triangles that could be drastically reduced without loosing in detail.

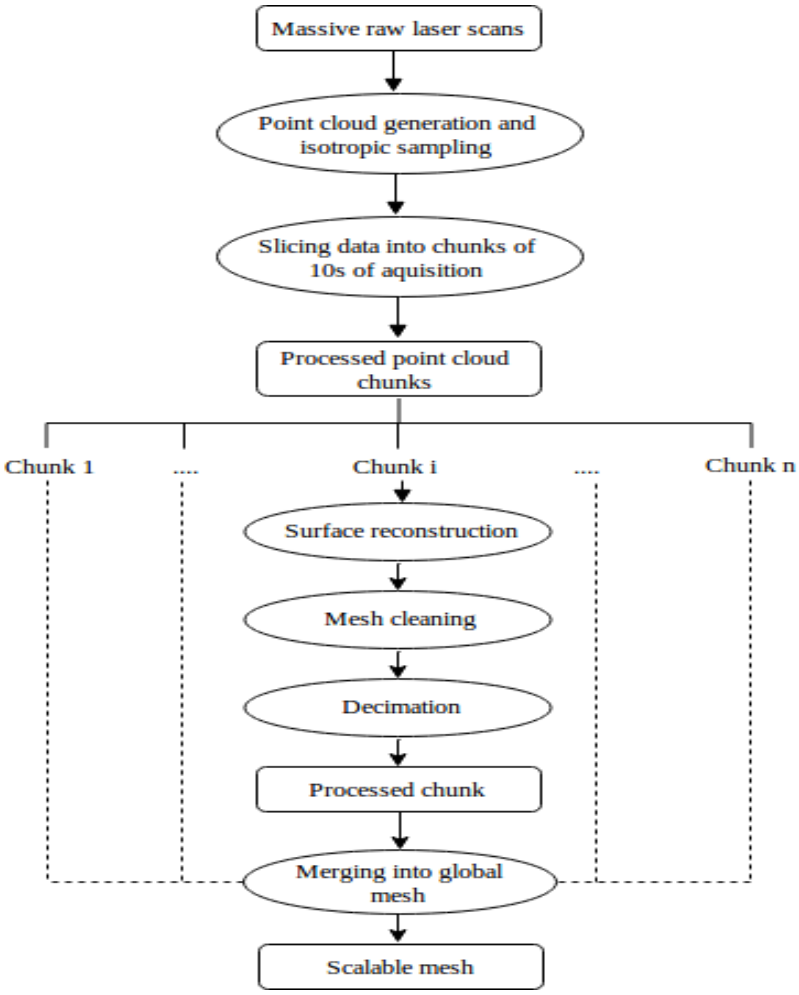


Figure 3.10: *The proposed work-flow to produce large scale models*

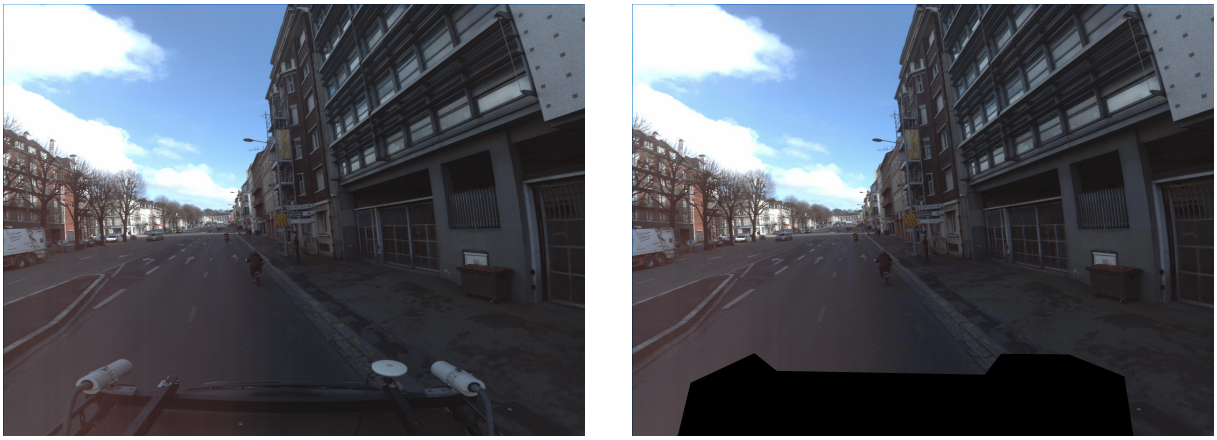
To this end, we apply the decimation method of [143, 144]. The algorithm proceeds in two stages. First, an initial collapse cost, given by the position chosen for the vertex that replaces it, is assigned to every edge in the reconstructed mesh. Then, at each iteration the edge with the lowest cost is selected for collapsing and replacing it with a vertex. Finally, the collapse cost of all the edges now incident on the replacement vertex is recalculated. In the following section, we present the adopted texture mapping approach based on the work of Waechter *et al.* [145].

3.3 Texture mapping

In this section, we expose the used approach for texturing large scale 3D realistic urban scenes acquired by our mobile mapping platform. Based on the work of [145], we adapt the algorithm so it can handle our camera model (with five perspective images). In the following, we give the outline of this texturing technique and its requirements.

3.3.1 Preprocessing

To work jointly with oriented images and LiDAR scans acquired by a mobile mapping system, the first requirement is that both sensing modalities have to be aligned in a common frame. Thanks to the rigid setting of the camera and the LiDAR mounted on the mobile platform yielding a simultaneous image and LiDAR acquisition, we assume that we have a reasonable accurate registration. However, such setting entails that a visible part of the vehicle appears in the acquired images. To avoid using these irrelevant parts as texture, an adequate mask is automatically applied to the concerned images (back and front-facing images) using the known pose of the vehicle in the camera space. The results of applying the mask on the front-facing images are shown in Figure 3.11.



(a) Before applying the mask (b) After masking

Figure 3.11: *A mask is automatically applied on the collected images to avoid using the visible part as a texture for the final model*

Following the approach of [145], texturing a 3D model with oriented images is a two-stage process. First, the optimal view per triangle is selected with respect to certain criteria yielding a preliminary texture. Second, a local and global color optimization is performed to minimize the discontinuities between adjacent texture patches.

3.3.2 View selection

Let us consider a triangular mesh M defined by its set of faces such that $M = \{t_1, ..., t_m\}$ where m are the number of triangles in M along with a set of registered camera views $V = \{v_1, ..., v_n\}$ where $|V| = n$. Our objective is to compute the visibility of each triangle $t \in M$ in the set of camera views V. This can be framed as computing a labeling $\mathcal{L} = \{l_1, ..., l_m\} \in \{1, ..., n\}^m$ prescribing to each triangle its corresponding camera view. Since a given triangle t can lie within the field of view of multiple cameras, not all these

views are equally suited for texturing t. Therefore, an appropriate cost α_i^j which reflects the quality of the camera view v_j w.r.t. a triangle t_i should be carefully designed. Thus an optimal labeling $\mathcal{L}^\star$ can be defined as $\mathcal{L}^\star = \left\{l_i \mid l_i = \ \text{argmin}_i \alpha_i^j\right\}$. In the literature, the quality of a view can be assessed by its proximity to the triangles, a fronto-parallel viewing direction and its high resolution [131]. However, restraining the optimality of view selection to the quality of camera views means that undesirable texturing effects especially visible seams will not be handled. In practice, if two adjacent triangles $t_i, t_j \in M$ are textured from different camera views, severe seams that alter the quality of the final texture will appear as demonstrated by Lempitsky and Ivanov [131]. To circumvent this effect, we need to maximize the quality of views while simultaneously minimizing the seams' visibility.

Akin to the work of Waechter *et al.* [145], a two-terms energy formulation is adopted to compute a labeling $\mathcal{L}$ that assigns a view l_i to be used as texture for each mesh face t_i:

$$E(l) = \sum_{t_i \in M} E_d(t_i, l_i) + \lambda \sum_{t_i, t_j \in Edges} E_s(t_i, t_j, l_i, l_j) \tag{3.3}$$

where λ is the weight balancing the smoothness and the data term and

$$E_d = -\int_{\phi(t_i, l_i)} ||\nabla(I_{l_i})||_2 dp \tag{3.4}$$

$$E_s = [l_i \neq l_j] \tag{3.5}$$

The data term E_d (3.4) computes the gradient magnitude $||\nabla(I_{l_i})||_2$ of the image into which face F_i is projected and sum over all pixels of the gradient magnitude image within face F_i's projection $\phi(F_i, l_i)$. This term is large if the projection area is large which means that it prefers close, orthogonal and in-focus images with high resolution. The smoothness term E_s (3.5) minimizes the seams visibility (edges between faces textured with different images). In the chosen method, this regularization term is based on the Potts model ([.] the Iverson bracket) which prefers compact patches by penalizing those that might give severe seams in the final model and it is extremely fast to compute. Finally, $E(l)$ (3.3) is minimized with α-expansion [146].

3.3.3 Color adjustment

After the view selection step, the obtained model exhibits strong color discontinuities due to the fusion of texture patches coming from different images and to the exposure and illumination variation especially in an outdoor environment. Thus, adjacent texture patches need to be photometrically adjusted. To address this problem, first, a global radiometric correction is performed along the seam's edge by computing a weighted average of a set of samples (pixels sampled along the discontinuity's right and left) depending on the distance of each sample to the seam edge extremities (vertices). Then, this global adjustment is followed by a local Poisson editing [67] applied to the border of the texture patches. Finally, the corrections are added to the input images, the texture patches are packed into texture atlases, and texture coordinates are attached to the mesh vertices.

3.4 Experimental results

In this section, we present the qualitative and quantitative results of each step of our pipeline namely surface reconstruction and texture mapping.

3.4.1 Mesh reconstruction

To highlight the interest of our sensor topology-based reconstruction method, we use point clouds coming from RIEGL-VQ250 LiDAR used in the pLaTINUM project mapping survey. Compared to a Velodyne HDL64-E, this sensor has a much simpler acquisition geometry making, thereby, the task of restoring its topology straightforward. In Figure 3.13, we show qualitative results of the reconstructed mesh based on the sensor topology technique. As we can notes from Figure 3.13, the resulting mesh is extremely dense even in flat regions where geometric information can be substantially simplified without altering the precision of the reconstruction. We define the compression rate of the reconstructed mesh as $C = 1 - \frac{\theta_{dec}}{\theta_{raw}}$ where θ_{raw} is the initial size of the mesh (in number of triangles) and θ_{dec} is the size of the decimated mesh.

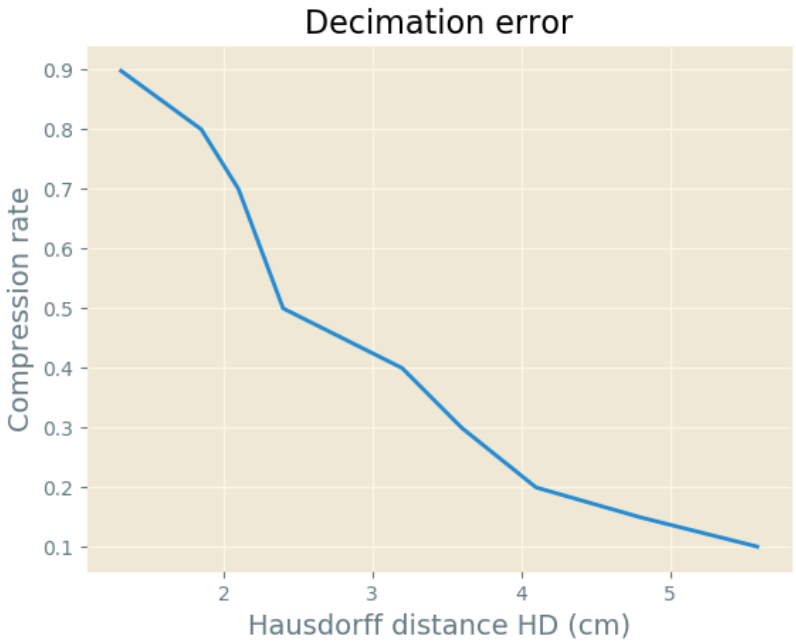
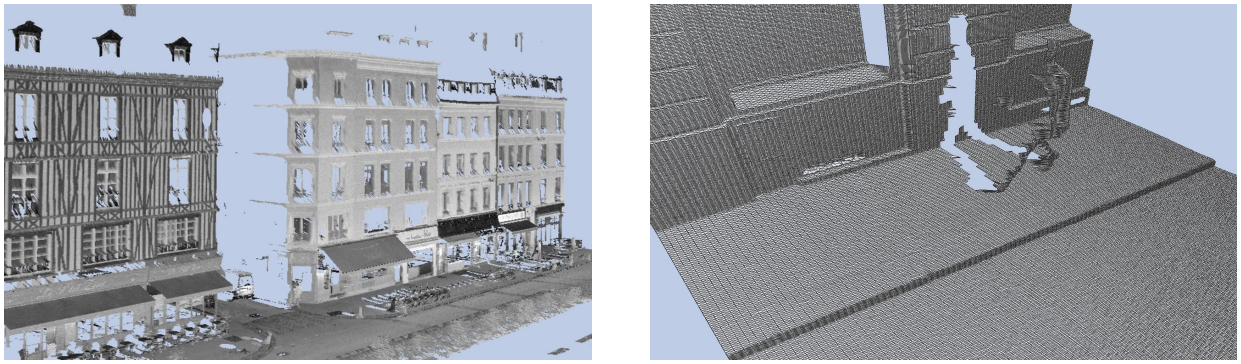


Figure 3.12: *The compression rate with respect to the Hausdorff distance HD*

In Figure 3.12, we illustrate the compression rate of the mesh with respect to its reconstruction error. In practice, we configure the algorithm such that the approximation error is below $3.5cm$, which allows in average to reduce the number of triangles to around 30% of the input number of faces.

Figure 3.14 exhibits a part of the reconstructed mesh with and without the constraint on the maximum length of the triangle edges. This is an important pre-processing step since a regular triangulation allows for an efficient view selection

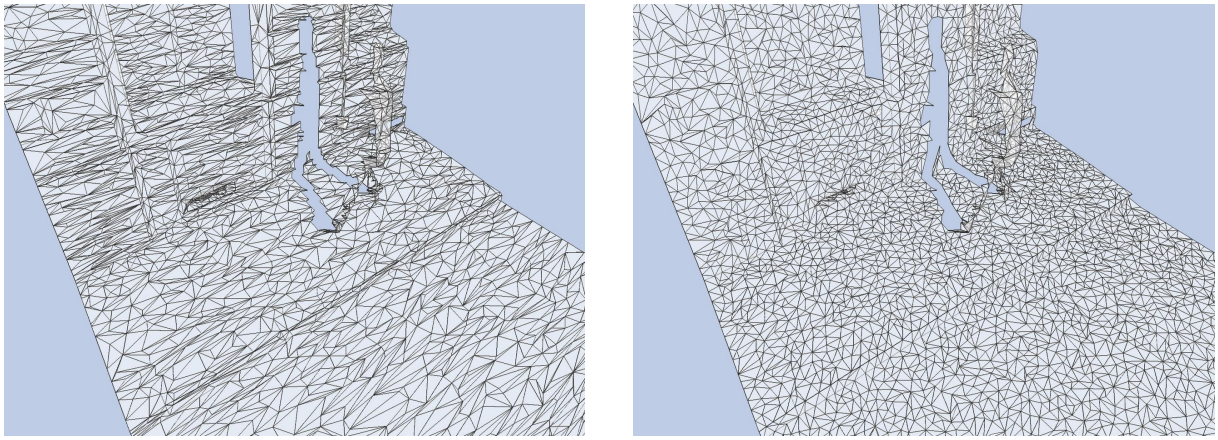
In order to evaluate quantitatively the proposed approach, we compare it against the acquired point cloud. For a fair evaluation, we follow the same procedure as in



(a) the reconstructed mesh from RIEGL-VQ250 acquisition

(b) zoomed region before decimation

Figure 3.13: *Illustration of the reconstructed mesh using the sensor topology of RIEGL-VQ250 LiDAR*



(a) Reconstructed mesh with elongated triangles

(b) Reconstructed mesh with regular triangles

Figure 3.14: *Illustration of the triangles' edge maximum length constraints*

[109], where the raw acquired LiDAR point clouds are considered as a ground truth since they are accurate and dense enough at least locally. The reconstruction is subsequently performed on a down-sampled version of the point cloud. Our reconstruction results are compared against the produced mesh by Poisson [120], Screened Poisson [121] and Ball Pivoting algorithms. In the literature two metrics are mainly leveraged to evaluate surface reconstruction; accuracy and completeness. Accuracy is defined as the distance of the reconstruction to the reference (the ground truth). Completeness, in contrast, is the distance from the reference to the reconstruction. In practice, the evaluation is framed as a set-to-set distance problem. We use three variants of distances; the average distance (AD), the residual mean square distance (RMSD) and the Hausdorff distance (HD).

Formally, let P be the set of vertices of the reconstructed mesh with $|P| = n$ using the aforementioned methods and GT the set of ground truth LiDAR point clouds such that $|GT| = m$. The distance between the two sets P and GT in a norm $||.||$, is defined as:

$$d(P, GT) = \inf\{d(p, p') \mid p \in P,\ p' \in GT\} \text{ where } d(p, p') = ||p - p'||. \tag{3.6}$$

The Average distances (AD) from the set P to GT denoted $AD_{P \to GT}$ and inversely

from GT to P denoted $AD_{P\rightarrow GT}$ are defined respectively as:

$$AD_{P\rightarrow GT} = \sum_{p\in P} \frac{1}{n} \min_{p'\in GT} ||p - p'|| \tag{3.7}$$

$$AD_{GT\rightarrow P} = \sum_{p'\in GT} \frac{1}{m} \min_{p\in P} ||p' - p|| \tag{3.8}$$

The Residual Mean Square distances (RMSD) from the set P to GT denoted $RMSD_{P\rightarrow GT}$ and inversely from GT to P denoted $RMSD_{P\rightarrow GT}$ are defined respectively as:

$$RMSD_{P\rightarrow GT} = \sqrt{\sum_{p\in P} \frac{1}{n} \min_{p'\in GT} ||p - p'||^2} \tag{3.9}$$

$$RMSD_{GT\rightarrow P} = \sqrt{\sum_{p'\in GT} \frac{1}{m} \min_{p\in P} ||p' - p||^2} \tag{3.10}$$

The one-sided Hausdorff distances (HD) from the set P to GT denoted $HD_{P\rightarrow GT}$ and inversely from GT to P denoted $HD_{P\rightarrow GT}$ are defined respectively as:

$$HD_{P\rightarrow GT} = \max_{p\in P} \min_{p'\in GT} ||p - p'|| \tag{3.11}$$

$$HD_{GT\rightarrow P} = \max_{p'\in GT} \min_{p\in P} ||p' - p|| \tag{3.12}$$

The accuracy is defined w.r.t. $AD_{P\rightarrow GT}$, $RMSD_{P\rightarrow GT}$, $HD_{P\rightarrow GT}$ while the completeness is defined w.r.t. $AD_{GT\rightarrow P}$, $RMSD_{GT\rightarrow P}$, $HD_{GT\rightarrow P}$. We also use the Symmetric Hausdorff Distance defined as the average distance of the two one-sided HD distances:

$$SHD = \frac{1}{2}(HD_{P\rightarrow GT} + HD_{GT\rightarrow P}) \tag{3.13}$$

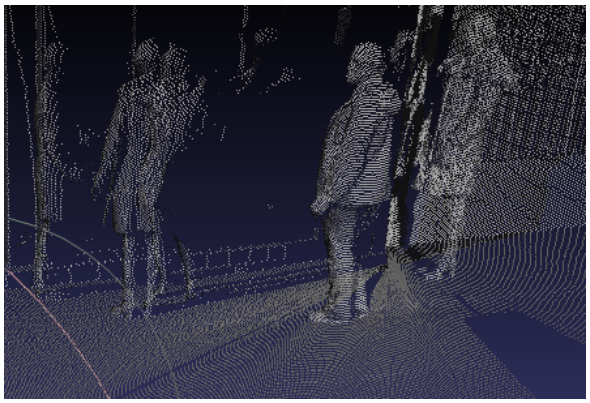
These scores are reported in table 3.3 for a chunk of 10^6 points from the down-sampled 3D point cloud. Figure 3.15 shows qualitative reconstruction results on a challenging scene using the scalable methods of the literature.

Method	**AD(m)**		**RMSD(m)**		**HD(m)**		**SHD(m)**
	$P \rightarrow GT$	$GT \rightarrow P$	$P \rightarrow GT$	$GT \rightarrow P$	$P \rightarrow GT$	$GT \rightarrow P$	
Poisson [120]	0.11	0.20	0.39	0.35	8.15	8.142	8.146
Screened Poisson [121]	0.08	0.15	0.23	0.17	3.72	4.61	4.165
Ball pivoting [147]	0.0071	0.0098	0.019	0.064	0.16	5.53	2.845
Ours	**0.0022**	**0.0042**	**0.0052**	**0.012**	**0.05**	**0.37**	**0.21**

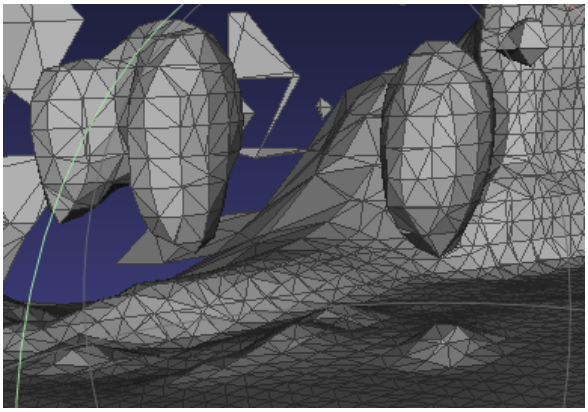
Table 3.3: *Surface reconstruction evaluation*

3.4.1.1 Discussion:

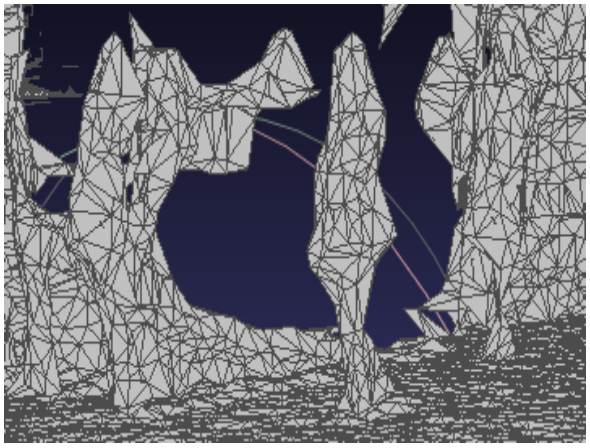
We choose to evaluate the reconstruction error using one-sided distances as they provide additional information. Increasing distances from the reconstruction to the reference indicate the areas that have been deviated from their true position in the reference while higher distances from the reference to the reconstruction represent unreconstructed parts



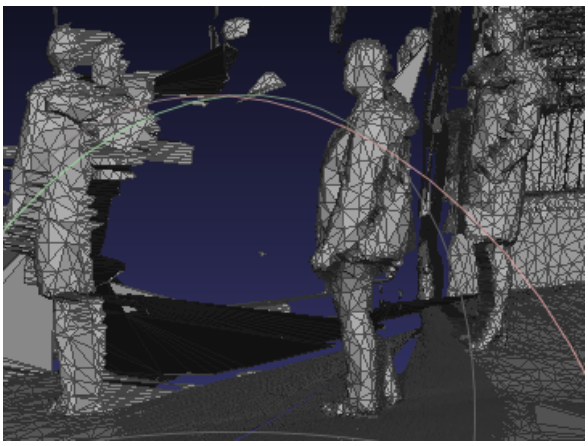
(a) the input point cloud



(b) Poisson reconstruction [120]



(c) Ball pivoting [147]



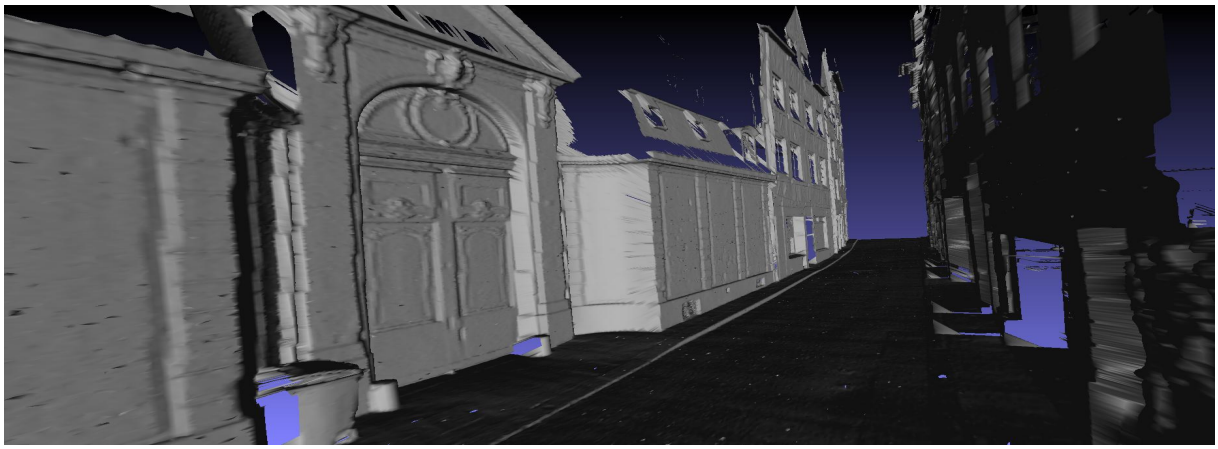
(d) Our sensor topology based reconstruction

Figure 3.15: *Qualitative reconstruction results from a point cloud of size* 10^6 *using Poisson [120], Ball pivoting [147] and our sensor-topology based reconstruction. We show a small part to illustrate the accuracy of our reconstruction.*

due to missing or sparse data in the reference. From Table 3.3, we observe that our method has the lowest distances achieving a good compromise between accuracy ($P \rightarrow GT$) and completeness ($GT \rightarrow P$) compared to other methods. This is was expected as our reconstruction is a simple triangulation of the geo-referenced 3D points w.r.t. the scanner rotation and the time of acquisition. In contrast to global methods; Poisson [120] and Screened Poisson [121] where a prior normal estimation step is required which it has its share of error in the final reconstruction, our method relies on accurate information provided by the sensor (time and rotation angle for each echo/pulse). Being a local method, Ball pivoting [147] is able to reconstruct a more precise mesh than Poisson and Screened Poisson, but it comes at the expense of the reconstruction time and limited scale. While being accurate, the sensor-topology-based reconstruction is not able to handle occlusions as the LiDAR has a fixed pose in the vehicle coordinate system such that orthogonal objects to the moving direction of the car whose geometry is planar (co-planar to the scanning plan of the LiDAR) are not reconstructed. Finally, our method remains essentially dependent on the known sensor geometry and configuration making it unsuitable for arbitrary point clouds. Therefore, our algorithm suffers from its reduced generalization capabilities.

3.4.2 Texture mapping

In this section, we show qualitative texturing results (Figure 3.16). The influence of the color adjustment step on the final textured models is shown in Figure 3.17. Before the radiometric correction, the textured model exhibits severe color discontinuities especially on the border of the door and on some parts of the road (best viewed on screen). More results are presented in the appendix to illustrate the high quality textured models in different places in Rouen, France.



(a) the reconstructed mesh

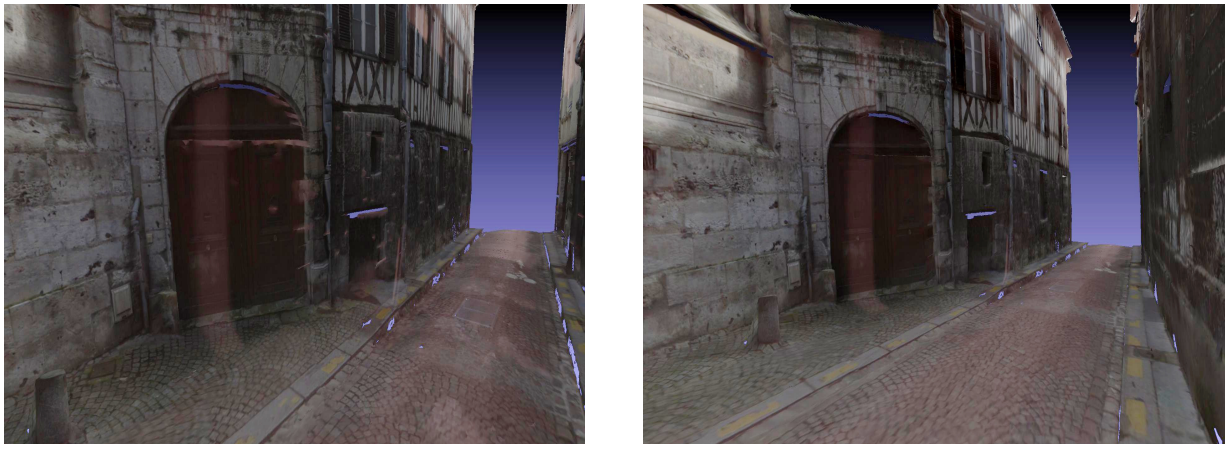
(b) the corresponding mesh after texturing

Figure 3.16: *Texturing result on a small part of street in Rouen, France*

In Figure 3.18 we show the influence of the regularization parameter λ on the final textured model. Through a grid search, we found experimentally that a value of $\lambda = 0.75$ produce more compact texture with less seams. Smaller values result in small patches while larger values tend to over-smooth the texture over the surface resulting in large imprecise patches.

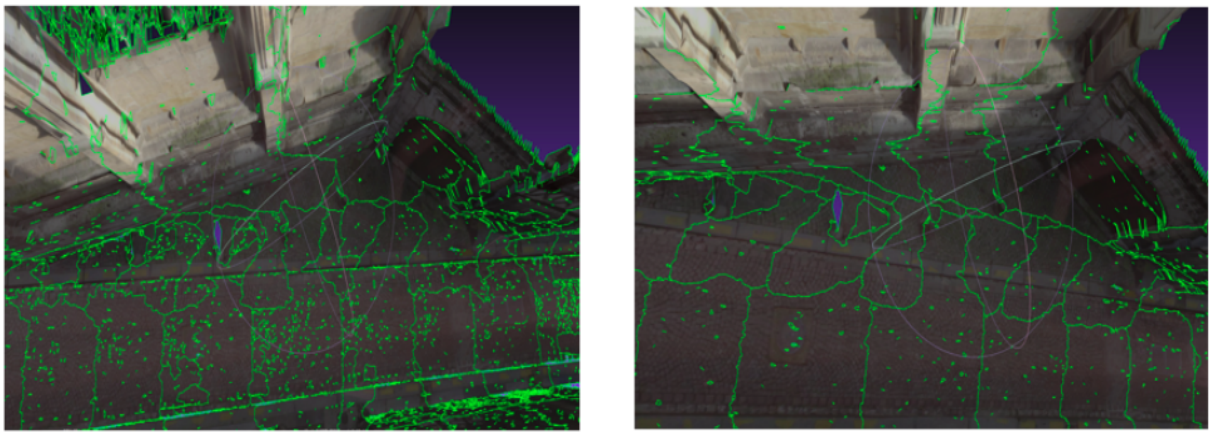
3.4.2.1 Discussion:

As argued by Waechter *et al.* [145], the chosen data term E_d must account not only for the geometry of the image, but also it needs to consider its radiometric content in order to select the best view for each triangle. In recent work of Fu *et al.* [148], the chosen data term



(a) Before color adjustment (b) After color adjustment

Figure 3.17: *The effect of color adjustment on the textured mesh. One can observe radiometric artifacts on border of the door and along the road (best viewed on screen). While some of the artifacts were partially adjusted, others can not be corrected due to the gradient magnitude of the data term.*



(a) $\lambda = 0.3$ (b) $\lambda = 0.75$

Figure 3.18: *The regularization effect of the weight λ on the smoothness term E_s. Texture patches are colored with green*

favors views with the largest projection area. However this implies that blurred views in addition to background-facing views are going to be also selected as texture. Therefore, we stick to the original formulation of [145]. However, whilst the chosen E_d accounts for views with large projection area and a high gradient magnitude at the same time, it over-fits to radiometric artifacts such as shadows or severe variation in illumination as depicted in Figure 3.17. Moreover, this data term is more susceptible to select views with high-frequency content to texture the background which corresponds most of the time to occluders.

Finally, we note that while this energy-minimization-based method produce accurate and visually compelling texture mapping results, they remain limited to the scale of the input data. That is why a temporal slicing strategy is adopted to texture the entire dataset. However, it should be noted that using local methods requires an explicit merging

approach of these temporal chunks to avoid inconsistencies between adjacent textured triangles in consecutive chunks.

3.4.3 Performance evaluation:

We evaluate the performance of each step of our pipeline on the dataset described in the previous chapter acquired by Stereopolis II [142] during this project. It consists of 17km of 6 hours of acquisition of both LiDAR and images yielding nearly to 2 billion georeferenced points and 40000 full HD images (more than 500 Gigabytes of raw data). The input data consists of raw LiDAR point clouds with their mutually registered perspective RGB images.

Acquisition	# Views	# Faces	Image resolution
10s	120	1.8 Million	2048×2048

Table 3.4: *Statistics on the input data per chunk*

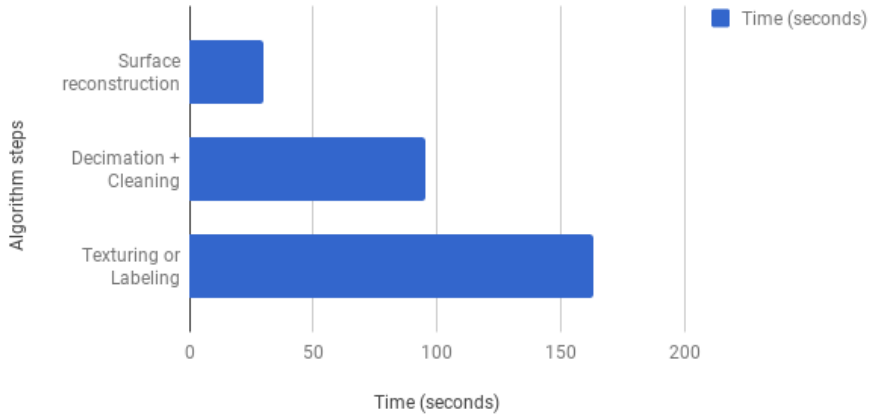


Figure 3.19: *Performance evaluation of a chunk of 10s of acquisition*

In Table 3.4, we present the required input data to texture a chunk of acquisition ($10s$); the average number of views and the number of triangles after decimation. Figure 3.19 shows the timing of each step in the pipeline to texture the described setting. Using a 16-core Xeon E5-2665 CPU with 12GB of memory, we are able to generate a 3D mesh of nearly 6 Million triangles in less than one minute compared to the improved version of Poisson surface reconstruction Kazhdan *et al.* [121] which reconstruct a surface of nearly 20000 triangle in 10 minutes. Moreover, in order to texture small models with few images (36 of size (768×584)) in a context of super-resolution, the algorithm of Goldlucke *et al.* [149] takes several hours (partially on GPU) compared to the few minutes we take to texture our huge models. Finally, all the dataset ($17Km$ of LiDAR acquisition + $40K$ images) is textured in less than 30 computing hours.

3.5 Conclusion

In this chapter, we have demonstrated a full pipeline to produce textured mesh from mobile mapping images and LiDAR scans data at city scale. It is mostly based on state-

of-the-art techniques that have gained a level of maturity compatible with such large scale processing. While being simple, the sensor topology-based reconstruction is quite novel. Through this reconstruction method, We have shown that available sensor information, which is ignored most of the time, can be adequately leveraged to produce dense and accurate large-scale surface. Apart from autonomous navigation, we believe that such textured mesh can find multiple applications, directly through visualization of a mobile mapping acquisition, or more indirectly for processing jointly image and LiDAR data: urban scene analysis, structured reconstruction, among others. This work leaves however prominent topics unsolved, and most importantly the handling of overlaps between acquired data, at intersections or when the vehicle passes multiple times in the same scene. Dealing with these issues poses numerous challenges:

- Precise registration over the overlaps, referred to as the loop-closure problem.
- Change detection.
- Data fusion over the overlaps, which is strongly connected to change detection and how changes are handled in the final model.

Moreover, our work proposed a reconstruction method based exclusively on LiDAR information, albeit we believe that the images hold a pertinent geometric information that could be used to complement the LiDAR reconstruction, in areas occluded to the LiDAR but not to the cameras (which often happens as their geometries are different). The recent work of Li *et al.* [136] adopting this multi-modal approach, has shown promising reconstruction results.

Finally, an important issue that was partially tackled in the texture mapping step which is the presence of mobile objects. Because the LiDAR and images are most of the time not acquired strictly simultaneously, mobile objects might have an incoherent position between image and LiDAR, which is a problem that should be tackled explicitly. The code source used to generate the 3D map as a textured mesh is publicly available online [6]. In the next chapter, we present a new approach to incorporate semantic attributes to the current representation which is, for now, based on geometric and photometric information.

[6] https://github.com/mboussah/MMS-texturing

Chapter

4 Supervised over-segmentation for 3D semantic mapping

Chapter content

Abstract

In the previous chapter, we introduced a new concept regarding the representation of a 3D map with a textured mesh. First, we explained how we accurately reconstructed our geometric map as a 3D mesh. Second, we enriched the reconstructed mesh with radiometric information through texture mapping. Geometry coupled with radiometry, however, is not sufficient for a reliable navigation. In addition to the latter attributes, autonomous agents need also to grasp the semantics of the scene. In this chapter, we present a new approach that enrich a 3D map with semantic information while handling its large scale. We demonstrate the performance and efficiency of our method on two types of indoor/outdoor map representations based on 3D point clouds and textured meshes.

Introduction

The lack of a high level understanding of the surrounding environments is one of the major stumbling blocks towards achieving reliable autonomous navigation. The next level of autonomous agents need to be endowed with the capacity of apprehending the scene in a human-centric manner. That is a wide knowledge of what objects are, their types as well as their spatial arrangements is required. As such, 3D maps to come have to extend beyond geometric and photometric layers to grasp also semantics of objects constituting the observed scene. In practice, the inclusion of rich semantic attributes involving human concepts within these maps is a key enabler for self-controlled agents across a wide variety of tasks ranging from obstacle avoidance [150] to path planning [151]. The importance of semantic knowledge has been further demonstrated via the large existing body of work devoted to solve the semantic mapping problem [152].

With the recent advances in 3D sensing technologies and the ubiquity of affordable 3D acquisition devices such as time-of-flight cameras and low cost LiDAR, the access to 3D data has never been easier. Driven by the breakthrough in deep learning applied to 2D data, a great deal of effort has been directed towards translating these techniques to the 3D setting. As seen in the previous chapter, 3D data comes in different representations each of which with varying structural and geometric properties. The wave of attention dedicated to three-dimensional space, has resulted in a plethora of recognition and classification techniques applied to data represented as point clouds [14, 153], voxels [15, 154], octrees [155], and meshes [16]. Inferring semantics directly on 3D data has a prominent advantage over 2D representations since they are oblivious to view-dependent effects of 2D images such as background clutter, perspective, varying lighting conditions and occlusion. However, since most of 3D data is either acquired using structured-light 3D scanners (LiDAR) or generated using multi-view stereo and structure-from-motion, compared to 2D images with sub-centimetric resolution, the generated point clouds suffer from low resolution (up to 1 cm in the best case). Therefore, the developed 3D deep networks fail to capture fine-scale semantic patterns compared to their 2D CNNs counterpart.

To guarantee the high resolution properties of 2D images along with occlusion-free and view-agnostic properties of 3D mesh surfaces, we propose to use textured meshes as our 3D representation ensuring, thereby, a combined benefit from 2D and 3D. Nonetheless, as opposed to 2D images which are defined as functions on an Euclidean space (plane), sam-

pled on a 2D grid structure, 3D textured meshes do not preserve such regularity making the definition of convolution and pooling operators difficult. Moreover, a major problem that needs to be efficiently dealt with is the large scale of 3D data which remains an obstacle to most of state-of-the-art 3D deep networks.

Before the emergence of deep learning, one of the pioneering approaches in the literature introduced by Riemenschneider *et al.* in [156], adopt a probabilistic graphical model formulation in order to infer semantics to a map represented as a 3D mesh. Whilst the method proposed in [156] is highly optimized to circumvent data redundancy, energy-based methods in general are known to be local algorithms that operate on a limited scale of data. Consequently, an extra-merging step is required as a post-processing to ensure consistent semantics across the segments of the entire scene. Alternatively, several methods in the literature have proposed a pre-processing step before inferring semantics. The latter consists in computing an over-segmentation leading to a set of segments, namely superpixels for 2D images [18], superpoints [157] for point clouds and superfacets [158] for 3D meshes. Thanks to these *superstructures*, semantic segmentation is carried out at a higher level by associating a label to an entire segment instead of each single pixel/point/triangle unit. To handle large scale scenes while taking advantage of powerful deep learning networks, a recent work of Landrieu and Simonovsky [17] is among the first attempts that has achieved extremely competitive performance by making use of these hand-crafted-based superstructures. Even though, all these over-segmentation methods substantially simplify the subsequent task of semantic segmentation, we argue that, all of them rely on the assumption that if the produced segments are geometrically and/or radiometrically homogeneous, they are semantically homogeneous. As far as we are concerned this assumption should be challenged since the quality of semantic segmentation results depends on the quality of the computed over-segmentation. To overcome this limitation, two recent works of Jampani *et al.* [159, 160] have reported significant improvements upon hand-crafted approaches by introducing for the first time a supervised approach to generate task-specific superpixels in 2D images.

To this end, motivated by the success of learned over-segmentation techniques in 2D images, we propose in this chapter a deep-learning-based framework to extend this concept to 3D point clouds and 3D textured meshes in order to be able to handle large scale data while appropriately dealing with the irregularity of these 3D representations. More in details, we propose to frame point clouds and 3D mesh oversegmentation as a deep metric learning problem structured by an adjacency graph defined on the input point cloud or the textured mesh. We introduce the *graph-structured contrastive loss*, a loss function which learns to embed 3D points homogeneously within objects and with high contrast at their interface. This loss can be adapted to the non-differentiable task of oversegmentation by using our cross-partition weighting strategy. The points / triangles'embeddings themselves are computed from the local geometry and radiometry by lightweight models inspired from PointNet [14] for 3D points and MeshCNN [16] for 3D textured meshes. Finally, the superpoints / superfacets are defined as a piece-wise-constant approximation of the learned embeddings in the adjacency graph.

The remainder of this chapter is structured as follow: In Section 4.1, the closely related work in the literature is presented along with theoretical definitions of the main concepts

of our method. Our approach is explained in details in Section 4.2. Both Sections 4.3 and 4.4 introduce two applications of our method respectively on 3D point clouds and textured meshes. Finally, we conclude this chapter in Section 4.5 giving some insights and discussing potential future work.

4.1 Overview

In this section, we start by defining the key concepts of our approach as well as the existing literature regarding this subject. The most related methods to our work are explained in details with a particular focus on methods used in our evaluation experiments.

4.1.1 Over-segmentation

A clear definition of what an over-segmentation might be was first introduced by Ren and Malik [161] for 2D images. According to [161], an over-segmentation is a partitioning or grouping of 2D image pixels into meaningful regions w.r.t. to color, texture and other low-level properties. While in the latter work, authors employ the term *superpixels* to refer to the result of an over-segmentation, others such as Veksler *et al.* [162] draw a line between over-segmentation algorithms and superpixels algorithms. Stutz *et al.* [163] use the convention that superpixel algorithms allow fine control over the number of generated superpixels, while over-segmentation methods do not. However the terms superpixels and over-segmentation are employed interchangeably as in [161] in most of the rest of the literature. Since both of these concepts share the same goal which is facilitating subsequent tasks by substantially reducing the number of primitives, we believe that there is no use in differentiating between the two concepts. As a matter of fact, the large majority of over-segmentation methods, by design, allow to control the number of segments regardless to the used modality being 2D images, 3D point clouds or 3D meshes as will be discussed in the coming section. The only subtle nuance lies in the number of generated segments whether it is known in advance or not. For this reason, in the rest of this study, we refer to the result of an over-segmentation method as the set of *superstructures*.

It should be emphasized that this ambiguity resulted in a divergence among computer vision and photogrammetry researchers about a unique designation of an over-segmentation result in 3D. In some studies, segments generated from a point cloud over-segmentation are called superpoints [164], in others [165], they are referred to as supervoxels. For sake of clarity, we propose an unified definition regarding the over-segmentation output. For 2D images, an over-segmentation results in superpixels. In videos we refer to them as *supervoxels*. Superstructures computed by an over-segmentation of point clouds are called *superpoints*. For 3D meshes, we refer to them as *superfacets* or *supertetras* depending on the type of the mesh whether it is a triangular or tetrahedral mesh.

As a generalization of what is advocated in [161], we believe that pixels, points and triangles are respectively the consequence of discretization of digital 2D images and 3D scenes which make them unnatural entities. Superstructures come as an alternative to these units by representing objects of a scene by a set of perceptually meaningful atomic regions. The usage of over-segmentation in the literature is mainly justified by the gain of computational efficiency. For instance, in semantic segmentation of a 3D point cloud P containing n points, the solution space of a labeling L of P has a dimension equal to

L^n. An over-segmentation resulting in m superpoints (with typically $m \ll n$) will greatly reduce the solution space to L^m. While investigating these methods, we observed that the community have proposed a set of properties superstructures should satisfy. We can conclude that most of the researchers share the idea that a reliable over-segmentation must fulfill the following three criteria:

($\mathcal{P}1$) **object-purity**: the computed superstructures must not straddle different objects having different semantic labels.

($\mathcal{P}2$) **border recall**: the interface between superstructures must coincide with the borders of ground truth objects;

($\mathcal{P}3$) **regularity**: the shape and contours of the extracted segments must be regular and simple as naturally objects do not exhibit wiggly shapes.

In the following, we review the existing work concerning over-segmentation techniques applied to 2D images, 3D point clouds and 3D meshes. For each of these modalities we can roughly classify these methods as graph-based or cluster-based w.r.t. their conceptual design.

Over-segmentation methods	Modality	Output	Graph-based	Cluster-based	Object purity	efficiency	Learned features
Entropy rate [166]	2D images	superpixels	✓	✗	+++	++	✗
Veksler *et al.* [162]	2D images	superpixels	✓	✗	+++	++	✗
Grundmann *et al.* [167]	Videos	supervoxels	✓	✗	+++	++	✗
SLIC [18]	2D images	superpixels	✗	✓	++	+++	✗
DASP [168]	2D images	superpixels	✗	✓	++	+++	✗
Benshabat *et al.* [164]	point clouds	superpoints	✓	✗	+++	++	✗
Guinard *et al.* [157]	point clouds	superpoints	✓	✗	+++	++	✗
VCCS [169]	point clouds	superpoints	✗	✓	++	+++	✗
Lin *et al.* [165]	point clouds	superpoints	✗	✓	+++	+++	✗
Wu *et al.* [170]	mesh	supefacets	✓	✗	+++	++	✗
Simari *et al.* [171]	mesh	superfacets	✗	✓	++	+++	✗
Picciau *et al.* [172]	tetrahedral mesh	superfacets	✗	✓	+++	++	✗
SEAL [173]	2D images	superpixels	✓	✗	++++	++++	✓
SSN [160]	2D images	superpixels	✗	✓	++++	++++	✓

Table 4.1: *An overview of several over-segmentation methods in the literature.*

In Table 4.1, we show an exhaustive summary of the most known state-of-the-art over-segmentation methods classified w.r.t. their modality, output, the type of the used features (learned or computed) as well as two different evaluation criteria (*i.e.* Object purity and computational efficiency).

4.1.1.1 2D image/video over-segmentation (Superpixels/Supervoxels):

Before the surge of deep-learning-based approaches, unsupervised 2D superpixel segmentation has been extensively studied in the scope of many investigations and applications. These methods can be broadly split into two groups:

- **Graph based methods:** Superpixels are computed by partitioning the image considered as an undirected graph where pixels are the graph nodes and the pixels' affinities are the

edges. Felzenszwalb and Huttenlocher [174] construct a graph such that the nodes are the image pixels and the edges correspond to its 8-neighborhood adjacency. Nodes are merged subsequently w.r.t. to the edge weights defined by pixels' color differences and variations. The work of Grundmann *et al.* [167] is a generalization of the method of [174] for image superpixels to videos. A 3D graph including the temporal dimension of images is build on top of video sequences and partitioned with respect to the optical flow of regions to ensure the consistency of the grouped pixels into spatio-temporal supervoxels. However, both [167, 174] tend to produce superpixels/supervoxels with irregular size and shapes and straddle different objects violating, thereby, the object purity property ($\mathcal{P}1$). Liu *et al.* [166] formulated superpixel computation as a graph partitioning problem w.r.t. an objective function based on the entropy rate of a random walk on the graph. In contrast to this category of algorithms where a bottom-up approach is adopted by merging pixels into superpixels, [161, 162] are top-down methods where the graph is partitioned into disjoint subgraphs constituting the superpixels. In Ren and Malik [161] the graph built on top of the image is partitioned with respect to hand-crafted texture and contour features using the normalized cut algorithm. While [161] produces regular superpixels, it is expensive to compute for small images. Using a more efficient partitioning algorithm, an energy minimization formulation solved using graph-cut [175] is adopted by Veksler *et al.* [162] to generate compact superpixels for 2D images and consistent supervoxels for video sequences.

- **Clustering based methods:** These methods start with identifying initial cluster centers over the image and progressively refine them until the specified criteria are met. The most famous method in this category is the *Simple Linear Iterative Clustering* SLIC introduced by Achanta *et al.* in [18]. Based on k-means clustering, the algorithm is initiated by a set of seed pixels serving as initial cluster centers. Then all pixels are assigned to their nearest center and a new cluster center is computed. The latter steps are repeated several times until stability of cluster centers. Pixel-superpixel assignment is based on color and spatial information. In addition to the latter features, [168] proposed to leverage depth information for computing superpixels using the same clustering method.

Figure 4.1: *Illustration of superpixels generated on 2D images using the method in [166]. Image from [166]*

In Figure 4.1 we show an illustration of superpixels generated using [166] method. Despite their conceptual diversity, these methods share the fact that they exclusively rely on hand-crafted features to generate superpixels. Although, the assumption that

objects having similar geometric and/or radiometric properties share the same semantic label is not always guaranteed. Besides, recent studies (*e.g.* surveyed by Garcia *et al.* in [176]) have demonstrated that hand-crafted features were largely outperformed by deeply learned features in many occasions. Meanwhile, inspired by the success of deep learning supervised applications, a minority has recently started to exploit deep techniques for 2D superpixel segmentation. Jampani *et al.* [159] proposed SEAL a non differentiable graph-based superpixel segmentation approach which predicts 4-connected pixel affinities on image graph by incorporating the segmentation error into the loss during affinity learning to generate better boundary-preserving superpixels. Later, superpixel sampling networks (SSN) were proposed [160]. In this work, Jampani *et al.* developed an end-to-end trainable framework by tailoring a clustering-based approach SLIC [18] to be back-propagable using a soft assignment instead of k-means pixel-superpixel hard association during clustering akin to the works surveyed in Aljalbout *et al.* [177]. On the other hand, many extensions of 2D superpixel segmentation methods have been proposed to generate 3D superpoints by over-segmentation.

4.1.1.2 3D Point cloud Over-segmentation (Superpoints):

While there exists a prominent work on 2D superpixel generation, superpoint segmentation remains in the development stage. Much like the 2D domain, the literature in 3D can be divided into two main approaches:

- **Graph-based methods:** These variants of approaches start with constructing a connectivity graph that defines the space of possible superpoints. Second, for each 3D point a descriptor is estimated to encode the local geometry and colorimetry. A dissimilarity measure based weight is subsequently assigned to each edge of the graph. Finally, a sequential subgraph merging [164] or cutting [157] criteria are chosen to obtain the over-segmentation. Ben-Shabat *et al.* in [164] proposed three extensions of 2D local variation graph-based method to 3D superpoint segmentation and studied different strategies for constructing the graph, edge weights assignment and subgraph merging. In the work of Guinard *et al.* [157], a non-parametric segmentation model formulated as an energy minimization problem is proposed to partition the 3D point cloud to simple yet geometrically homogeneous shapes and solved using Cut-pursuit algorithm [178] (an efficient graph partitioning method introduced later). However, similar to the 2D case only hand-engineered features are used as descriptors to produce the partition (spatial coordinates and color in [164], local features (linearity, planarity, verticality) in [157]).

- **Clustering-based methods:** In this category of methods, the pioneering work of Papon *et al.* Voxel Cloud Connectivity Segmentation VCCS [169] starts by uniformly partitioning the point cloud using an octree into voxels that serve as an initialization for a local k-means clustering method. A subset of points are chosen as superpoint centers which are iteratively grown afterwards based on computed handcrafted features of adjacent points (*e.g.* x, y, z spatial coordinates and L, a, b color channels). Apart from its five hard adjustable parameters, the main disadvantage lies in its inability to properly deal with point clouds having non-uniform density since it requires a careful choice of voxel resolution so that more than one object cannot overlap within the same voxel. In this case, VCCS leads to badly preserved boundaries violating border recall property ($\mathcal{P}2$). Similar works followed the latter attempting to address the problem. Song *et al.* [179]

firstly detect the boundary points in LiDAR point clouds by estimating the discontinuity of consecutive points. Second, a clustering process is performed on a neighborhood graph constructed upon the point cloud and excluding edges connected by the detected boundary points. However, the assumption that the points are sequentially ordered reduces its generalization capabilities to other types of point clouds. In the work of Lin *et al.* [165], superpoints generation is formulated as a subset selection problem aiming to find a representative point for each superpoint to reduce the problem dimension. This involves an energy minimization based on dissimilarity distances between points and the superpoints representatives. Solving this energy does not require the initialization of seed points yielding to an adaptive resolution that preserves object boundaries more efficiently. Figure 4.2 illustrates over-segmentation results using the algorithms in [165, 169]

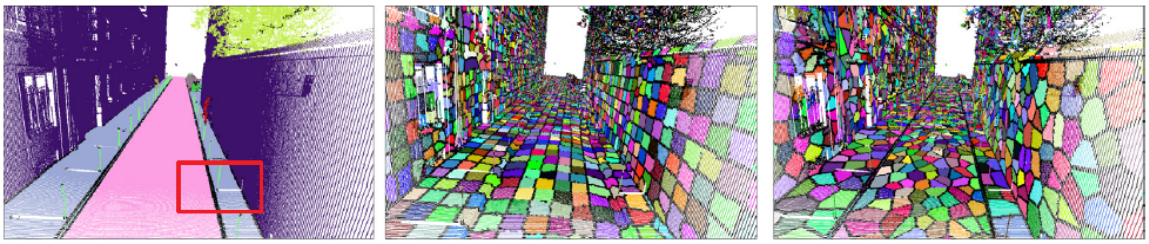


Figure 4.2: *Illustration of an over-segmentation on point clouds. Left: ground truth, middle: VCCS superpoints [169], right: superpoints constructed using Lin et al. [165]. Image from [165]*

4.1.1.3 3D mesh over-segmentation (Superfacets / Supertetras)

Much like superpoints and superpixels/supervoxels, superfacets/supertetras can be computed using two types of approaches:

- **Graph-based:** In this category, 3D mesh is represented by its dual graph where faces correspond to graph nodes and mesh adjacency constitute the set of edges. In the work of Wu *et al.* [170] , superfacets are computed as a pre-processing for a co-segmentation [1] task. the over-segmentation is the result of graph partitioning using normalized cuts with respect to geometric descriptors (*e.g.* signed distance function (SDF) [180], average geodesic distance (AGD) [181] associated with each face of the mesh. However partitioning a graph using normalized cuts is computationally expensive as discussed for 2D images.

- **Cluster-based:** Inspired by the SLIC approach introduced in [18] for superpixel image segmentation, Simari *et al.* [171] present a clustering method for 3D mesh over-segmentation. Based on *k-means* algorithm, the proposed method is a three-step approach. First superfacets centroids are iteratively initialized all over the input mesh. The first one being the closest triangle to the center of the entire mesh, consecutive centroids are placed subsequently w.r.t. to the maximum euclidean distance to the nearest already placed center. Second, all the faces of the mesh are assigned to their nearest centroids by computing its shortest path distance making thereby a first over-segmentation.

[1] Co-segmentation is the task of jointly segmenting the same object/shape in different set of 2D images or 3D mesh

The latter is refined in an iterative fashion in such a way centroids are updated w.r.t. to the mean centroid of the said superfacet. This procedure is repeated until superfacets centroids are stabilized over the algorithm iterations. A similar extension of this algorithm have been proposed by Picciau *et al.* in [172], albeit this time for tetrahedral mesh over-segmentation to produce *supertetras*.

While this class of methods allows for a good trade-off between efficiency and accuracy, clustering based methods depend on seed initialization which are tuned with regards to the topology of the underlying 3D mesh. Besides they do not handle the non-uniform sizes and shapes of mesh faces which influence substantially the over-segmentation quality. Cohen-Steiner *et al.* proposed VSA (variational shape approximation) in [182], which is a k-means style algorithm. VSA circumvent the latter problems by alternating between a geometric partitioning of the mesh using a region growing approach with respect to geometric similarities of triangular faces and a proxy fitting step that minimize the distortion error for a given partition. In order to compute an over-segmentation on textured meshes, Rouhani *et al.* in [158] extended VSA [182] by adding a photometric similarity measure that preserve image discontinuities in the texture map.

Figure 4.3 shows an illustration of an over-segmentation on 3D triangular meshes using the methods in [170, 171].

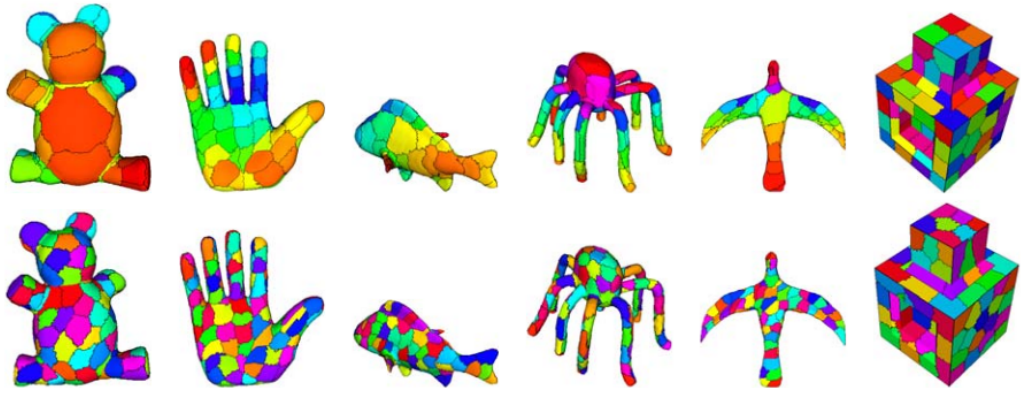


Figure 4.3: *Illustration of an over-segmentation on triangular meshes. Top: superfacets computed using the method of Wu et al. [170], Bottom: superfacets computed using Simari et al. [171]. Image from [171]*

Surprisingly, despite the huge progress in the field of deep learning applied to the 3D domain [14–16] and the ubiquity of annotated 3D large scale datasets [21, 24], to the best of our knowledge, there is no supervised over-segmentation technique, so far, that leverages deep-learning-based embeddings to compute an over-segmentation neither on 3D point clouds nor on 3D meshes. The irregular structure of 3D meshes and the disorder of point cloud compared to structured regular lattice characterizing 2D images explain the slow progress in supervised over-segmentation. Therefore, to spur research in this field, we propose to leverage deep embeddings to learn the over-segmentation task. This can be achieved using an interesting approach called *deep metric learning*. In the following, we define the deep metric learning problem then we give an overview of the most related work to ours.

4.1.2 Deep metric learning

Metric learning is about learning distance functions that are compatible with a certain similarity standard. Since the definition of a given similarity criteria is task-dependent, the success of such approaches is mainly related to the ability of aligning the learning objectives to the intended task. In practice, metric learning approaches [183] aim at learning feature representations such that it promotes close embeddings for similar data points in the feature space while penalizing close features of dissimilar data points. The increasing interest to these class of methods has been further boosted by the democratization of deep learning techniques on 2D images [38]. This success allowed for the development of an extension of traditional metric learning called *deep metric learning* where data features are encoded using deep neural networks instead of hand-crafted methods.

In the last few years, deep metric learning has proven its effectiveness across a large spectrum of recognition applications ranging from face recognition [184], medical image classification [185], visual search for product similarity [186] and person re-identification [187] among others. In all these applications, task-specific deep architectures are trained to minimise a well-designed loss function where a non linear feature representation is learned to bind to one another embeddings from similar classes while maintaining embeddings of different classes distant in the feature space. In the context of deep metric learning, the design of an adequate loss function has received a substantial attention in recent years. The desired properties of these objective functions are their fast convergence and their ability to reach a good (local) minimum during optimization. The most investigated loss functions in the literature are the *contrastive* loss [188] and the *triplet* loss [189]. In the following we give a detailed overview of each type of these objective functions.

4.1.2.1 Contrastive loss:

The contrastive loss is an euclidian-distance-based loss function that measures similarity between positive and negative pairs of samples in the dataset. Positive pairs are the set of data points belonging to the same class while the negative pairs inversely have distinct class labels. Let $\{x_{1,i}, x_{2,i}\}$ be a pair of input data points and $\{e_{1,i}, e_{2,i}\}$ the corresponding embeddings computed by a task-specific deep network. Operating on positive and negative pairs labeled respectively by $y_i = 0$ and $y_i = 1$, the contrastive loss is defined as:

$$\mathcal{L}_{contrast} = \frac{1}{N}\sum_{i=1}^{N}[(1-y_i)\|e_{1,i} - e_{2,i}\|_2^2 + (y_i)max(0, m - \|e_{1,i} - e_{2,i}\|_2)^2] \tag{4.1}$$

where m is a preset margin and N is the batch size used for training. In Figure 4.4 we show an illustration of one gradient iteration using the contrastive loss.

4.1.2.2 Triplet loss:

The triplet loss was proposed as an extension to the contrastive loss by considering a query sample called an *anchor* in addition to the positive and negative samples. This function reduce the distance between the embeddings of the anchor and the positive sample while simultaneously enlarging the distance between the anchor and the negative sample. Formally, we consider the set of input data points $\{x_i^a, x_i^p, x_i^n\}$ and their corresponding computed embeddings $\{e_i^a, e_i^p, e_i^n\}$ denoting respectively the anchor a from which

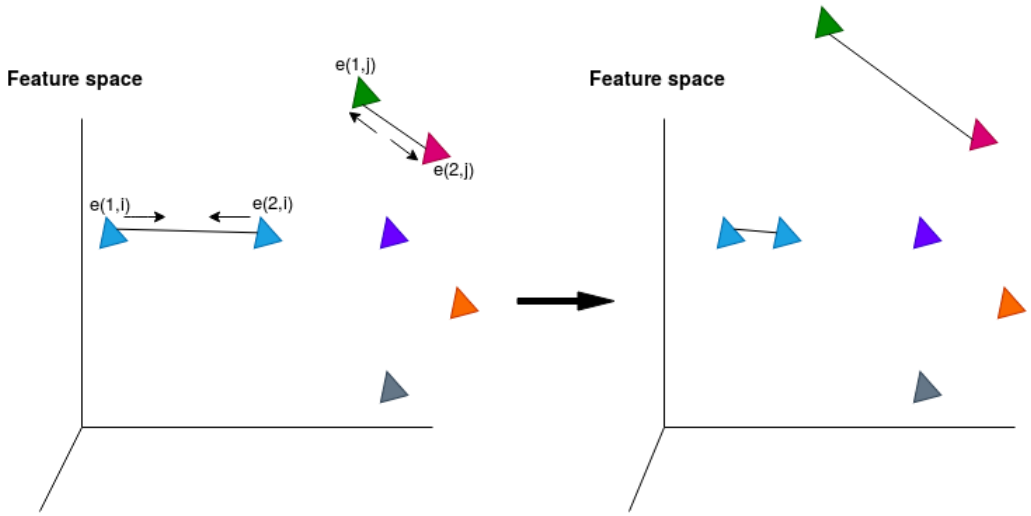


Figure 4.4: *Illustration of one gradient step during contrastive loss optimization for a positive and negative pairs: data points of the same color belongs to the same class*

a distance will be computed from a same class sample (positive p) and negative sample (n from a different class). The triplet loss can be written as:

$$\mathcal{L}_{triplet} = \frac{1}{N}\sum_{i=1}^{N} max(0, \|e_i^p - e_i^a\|_2^2 - \|e_i^n - e_i^a\|_2^2 + \lambda) \tag{4.2}$$

where N is the batch size and λ is a parameter to avoid the convergence to the trivial solution. Figure 4.5 shows an illustration of one step gradient during minimizing the triplet loss.

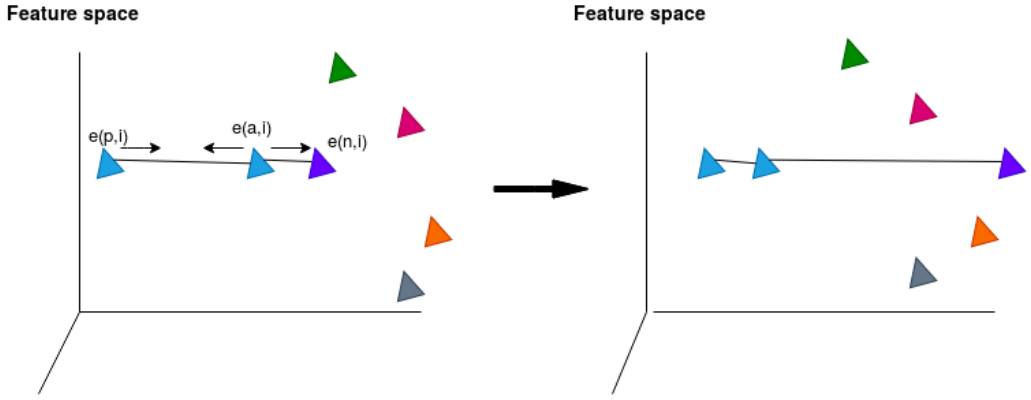


Figure 4.5: *Illustration of one gradient step during triplet loss optimization for a triplet of positive, negative and an anchor samples: data points of the same color belong to the same class*

Note that the expressiveness of this loss is hindered when $\|e_i^p - e_i^a\|_2^2 < \|e_i^n - e_i^a\|_2^2 + \lambda$. In this case, the loss will have no impact on the embeddings as the gain is equal to 0. This results in poor performance and convergence problems as a good local minimum is not reached. To this end, triplet mining [190] is commonly applied to mitigate this issue. Concretely, this technique considers only the triplets that give a positive loss. This

can be achieved using hard negative mining which consists in selecting triplets where the negative sample is strictly closer to the anchor than the positive sample. A semi-hard negative mining can be also leveraged where the negative sample is not necessarily closer to the anchor than the positive but still give positive loss.

We abstain from using this type of loss in our over-segmentation approach as providing a set of triplets from the data we are using is not straightforward. Furthermore, apart from the tricky training procedure, this loss suffers from slow convergence as well as a tendency to reach local optima as discussed by Sohn [191] since at each update, the comparison is performed with only one negative sample while ignoring the rest of them in the same batch. Instead, we propose an adaptation of the contrastive loss but different from [192], where this loss is used to improve features of 3D point clouds in a dense classification context, our task is related to over-segmentation through graph partitioning.

4.1.3 Graph theory

Graph theory is a branch of science whose purpose is to study *graphs*. A graph is a mathematical representation of a set of data elements where inner pairwise relations are linked through connections. Graph theory has been successfully used in a wide range of applications ranging from communication networks, transportation, social media and chemistry to mention a few. In particular, computer vision is one of the disciplines that heavily rely on this representation in several low-level tasks such as segmentation and tracking in addition to high-level tasks commonly referred to as recognition (classification, semantic parsing, etc.). The popularity of this representation stems from its powerful properties:

- well-grounded and mathematically proven methods
- flexibility and high level of abstraction
- versatility: can represent a wide variety of data across different disciplines of science.

In essence, data elements are called *nodes or vertices* of the graph and the links connecting them are called *edges*. In social media, the nodes of the graph can be the set of subscribed people and the edges connecting them are their friendships. In chemistry, proteins can be considered as the vertices of the graph while their interactions are modeled by the graph edges. In computer vision, pixels of 2D images, points of 3D point clouds or faces of a 3D mesh correspond to the node of the graph whereas the spatial adjacency is modeled by the graph edges. Our contribution in this chapter is inspired by mature graph-based approaches. Before introducing our method, we start by defining some of the commonly used graph taxonomies.

4.1.3.1 Definitions

In the following, we denote by $G = (V, E)$ the graph G where $V = (v_1, ..., v_n)$ is the vertex set and $E = \{e_{u,v} \mid u, v \in V\}$ is the edge set.

Definition 1. A directed graph is an ordered pair $G = (V, E)$ such that V is a non-empty set of vertices and $E \subseteq V \times V$ is the set of ordered pairs of different nodes called edges.

Definition 2. An undirected graph is a pair $G = (V, E)$ where V is the set of nodes and E the set of edges composed of the set of two-element subsets of V. The undirected graph can be derived by constraining the relation $E \subseteq V \times V$ to be symmetric.

Remark. It is frequently known that additional information can be attached to edges in the form of attributes. For instance, a *weight* function attributed to edges can be defined as $w : E \mapsto \mathbb{R}^+ \cup \{0\}$.

Definition 3. A graph $G' = (V', E')$ is called a *sub-graph* of $G = (V, E)$ if $V' \subset V$ and $E = \{E_{ij} \mid v_i \in V', \ \ v_j \in V'\}$.

Definition 4. A graph $G = (V, E)$ is called a *bipartite graph* if V can be partitioned into two subsets $V_1 \subset V$ and $V_2 \subset V$ such that $V_1 \cap V_2 = \varnothing$ and $V_1 \cup V_2 = V$, where the set of edges is defined as $E \subseteq V_1 \times V_2$.

Definition 5. A graph partition $\mathcal{S}$ is defined as a division of the vertex set V into disjoint subsets $\mathcal{S} = \{S_1, .., S_k\}$ such that $S_i \cap S_j = \varnothing$ and $E \subseteq S_i \times S_j$ for $i \neq j$ where $\bigcup_i S_i = V$.

4.1.3.2 Graph construction

A crucial key aspect for the success of graph-based methods is the design of an appropriate graph topology that faithfully models the pairwise interactions between data points. There is a wealthy literature proposing a wide variety of graph construction methods depending on the structure and topology of the underlying data. In the following we briefly explain the most popular static graph topologies. For a more in-depth study on this aspect the reader is referred to the work of Ulrike in [193].

The topology of the graph models the relationships between data points. In order to infer global structure from local information, the constructed graph needs to take into consideration the local neighborhood of each vertex of the graph. The most common graph adjacency topologies are:

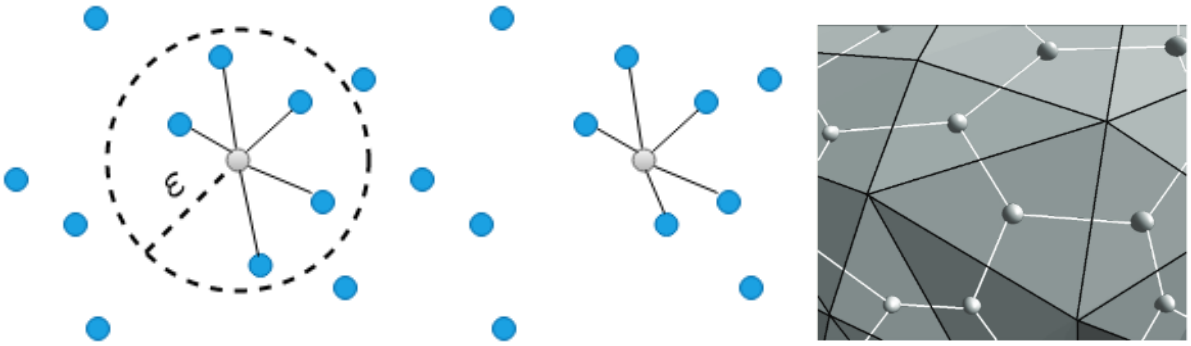
- **ε-neighborhood graph:** called also ε-graph, all vertices of this graph whose pairwise distance is smaller than ε are connected by an edge. This graph is shown in Figure 4.6 (a). Choosing a constant ε for all the nodes of the graph is not suitable to capture the local neighborhood structure. An improper thresholding of ε may results into subgraphs or disconnected components.

- **k-nearest neighbor graph:** in a k-NN graph, each vertex is connected to the set of vertices that lie within its k-nearest neighborhood vertices as depicted in Figure 4.6 (b). However the number of nearest neighbor vertices for each node in the graph varies from an object to another. k-NN graph remains more adaptive to scale and density than ε-graph.

- **Dual graph:** this graph is generally build on top of a 3D mesh. The nodes of the graph represent the mesh faces while the edges are the triangles adjacency relationships. The construction of this graph is illustrated in Figure 4.6 (c). Transforming a 3D mesh into a graph is traditionally used for mesh segmentation [194] by framing the latter task as a graph partitioning problem.

4.1.3.3 The cut pursuit algorithm

Introduced by Landrieu and Obozinski in [178], the cut pursuit algorithm is a working-set greedy strategy for minimizing functionals involving the total variation structured by a



(a) ε-graph (b) k-nn graph ($k = 5$) (c) dual graph

Figure 4.6: *Illustration of the common construction of static graphs. Image (c) from [194]*

graph $G = (V, E, w)$ where $w \in \mathbb{R}^+$ are the weights of edges E:

$$F(x) = f(x) + \sum_{(i,j)\in E} w_{(i,j)} \|x_i - x_j\| \tag{4.3}$$

where $x = (x_i)_{i\in V} \in \mathbb{R}^V$ is the variable of interest. For these functions the solution $x^\star$ of (4.3) is constant on the elements of a certain coarse partition P of V (*i.e.* $|P| \ll |V|$) due to the sparsity of its gradients. This solution can be decomposed into a smaller number of connected components of the graph G. In computational statistics and machine learning, the coarseness property of the solution can be exploited to speed-up large scale optimization problems.

As illustrated in Figure 4.7, the cut-pursuit algorithm has two main steps; *reduction* and *refinement*. Initially all the vertices of the graph are associated to the same connected component. In the reduction step, the graph G is split into constant connected

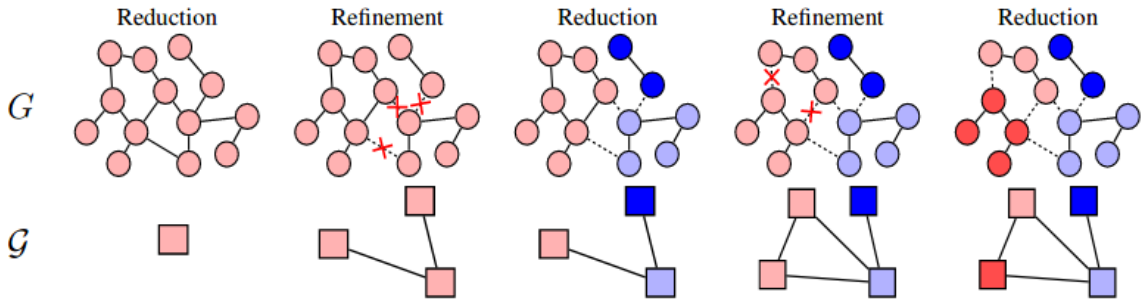


Figure 4.7: *Illustration of the different steps of the ℓ_0-cut pursuit algorithm. Image from [195]*

components and a reduced graph $\mathcal{G}$ encoding the new adjacency is computed. The reduced problem is solved under the constraint that all the vertices of the same connected component share the same value. In the refinement step, the current partition is further split such that the next reduced problem decrease F as much as possible. This is carried out by finding the steepest directional derivatives of F.

The aforedescribed cut pursuit algorithm is a versatile graph partitioning approach that can be applied to several data representations as long as they can be modeled by a graph.

However, graph partitioning methods in general remain intrinsically non-differentiable as they involve computing connected components and the objective function to be minimized is usually non-continuous and / or non-convex. This non-differentiability property inhibits the use of such interesting methods in the framework of deep learning as it is very complex even impossible to back-propagate gradients through and update the network weights accordingly during optimization. In our proposed method, we show how we bypass this limitation. In the following, we review the literature on deep learning techniques applied on 3D data.

4.1.4 Deep learning on 3D data

During the past few years, deep learning on 2D images has achieved a tremendous progress in a wide variety of computer vision tasks ranging from classification and segmentation [196] to detection and localization [197]. The key recipe for the success of these networks is a combination of convolution, non-linearity and pooling layers yielding to a robust framework which is invariant to a set of variations of the input [198]. Driven by the huge success of these 2D architectures, several attempts [199] have been proposed to transfer this knowledge to the 3D setting. The fundamental challenge that needed to be handled is how to translate deep frameworks operating on 2D images represented as structured 2D grids to three-dimensional data which are characterized by an implicit neighbourhood adjacency with an irregular support. As discussed in Chapter 3, 3D sensed data can be intrinsically represented in several forms among which the most common ones are point clouds and 3D meshes.

Following the terminology from the literature [3], 3D data representations can be broadly classified into two major categories w.r.t. its underlying *euclidean* properties. A 3D representation is said to be euclidean if it exhibits a grid data structure. Implicitly this means that this representation can be trivially defined as a function on the Euclidean space sampled on a grid. This category includes 3D data represented by a volumetric grid [15, 154, 155] or using 2D multi-view projections [200, 201]. In contrast, a non-euclidean representation lacks an euclidean vector space structure where the underlying data can be represented by a function. Directed graphs [202], unordered point clouds [14, 153] as well as 3D meshes (or manifolds) [16, 26] fall under the latter category. In this section we shed the light on the recent advances in deep learning applied to these 3D representations regardless to the task that were meant for.

4.1.4.1 Volumetric approaches

This set of methods are among the first attempts to tailor 2D deep learning on 3D data. In practice, the input data is volumetrically discretized using a regular voxel grid [15, 154] or using an octree [155] resulting into voxels of adaptative sizes so that 3D convolution becomes feasible.

Voxnet [154] construct an occupancy grid of size $32 \times 32 \times 32$ upon the input data (RGBD-D, LiDAR point clouds or 3D CAD models). The proposed network is composed of two convolutional layers, a pooling layer and two fully-connected layers. The particularity of the convolution operator lies in the 3D filter kernel used instead of common 2D filters in case of 2D images.

VoxelNet [15] illustrated in Figure 4.8 starts by partitioning the space of raw input point clouds into equally spaced voxels. Once points are grouped within voxels, a random

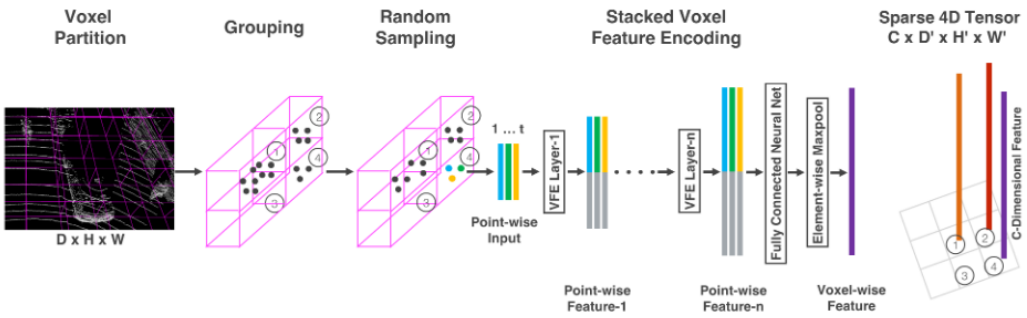


Figure 4.8: *Illustration of VoxelNet feature encoding steps. Image from [15]*

sampling is applied to reduce the number of input points while ensuring a uniform distribution inside them. Afterwards, a small embedding is computed for the set of points within each voxel using a simplified multi-layer perceptron (MLP) composed of a stack of linear layer, batch norm and non-linearity layers aggregated using a max-pooling operation. Finally those local features are passed to a region proposal network which consist of a classic Convolutional Neural Networks (CNNs) for object detection.

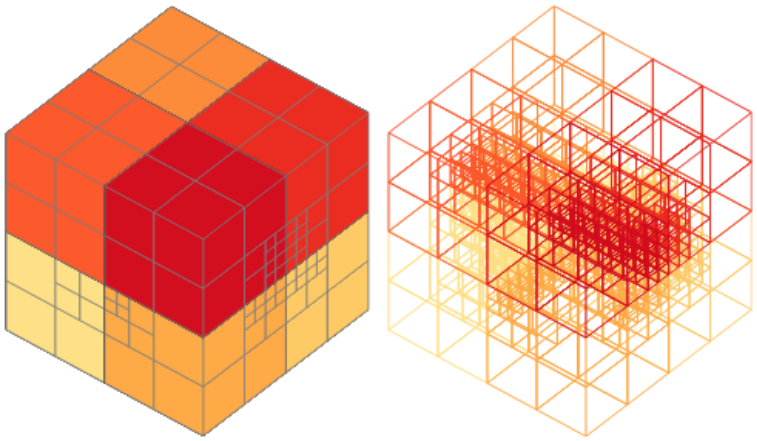


Figure 4.9: *Illustration of the hierarchical octree data structure to partition 3D data space. Image from [155]*

While the latter approach allowed to extend 2D convolution to the 3D setting, for dense 3D data the computational and memory costs grow cubically with data resolution limiting thereby the input resolution to around 30^3 voxels. To overcome this problem, OctNet [155] exploits the sparsity properties of 3D data by building hierarchical and adaptive octree that subdivide only cells containing relevant information as shown in Figure 4.9. This alleviates the memory load when performing 3D convolution. Whilst occupancy-grid based methods offer, to some extent, decent results, they are limited to single object classification and small datasets in most of the reviewed methods. Furthermore, the input resolution is substantially reduced to fit memory requirements. Consequently, the gain information from working directly in 3D is outweighed by the information loss induced because of data down-sampling.

4.1.4.2 Multi-view 2D-based methods

In this category, 3D data is represented by a set of multiple images captured from different rendered view points. The interest behind this approach is mainly its ability to leverage mature CNNs architectures in addition to large scale annotated data for training not to mention the decreased memory load when processing 2D images.

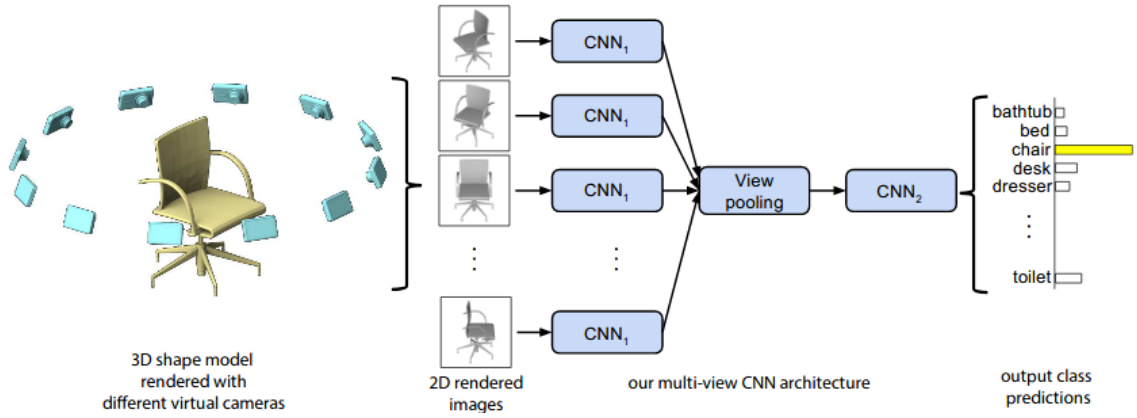


Figure 4.10: *Illustration of 3D shape recognition using multi-view convolutional networks. Image from [200]*

One of the earliest works in multi-view deep learning is the MVCNN introduced in [200]. In its simplest configuration, MVCNN consumes 12 images of a 3D object rendered from different view points each of which is fed to an independent CNN. The computed feature maps of each image are passed to a view pooling layer as shown in Figure 4.10 which will be the input to an aggregated CNN. The Convolutional Neural Networks (CNNs) architecture used in [200] is based on VGG backbone [203].

Akin to MVCNN [200], the architecture proposed by Kalogerakis *et al.* [201] takes as input a set of multiple views of a 3D object optimized for maximal surface coverage and a polygonal mesh. These views are processed independently by pre-trained Fully Convolutional Networks (FCNs) producing confidence maps of the same size as the input 512×512.

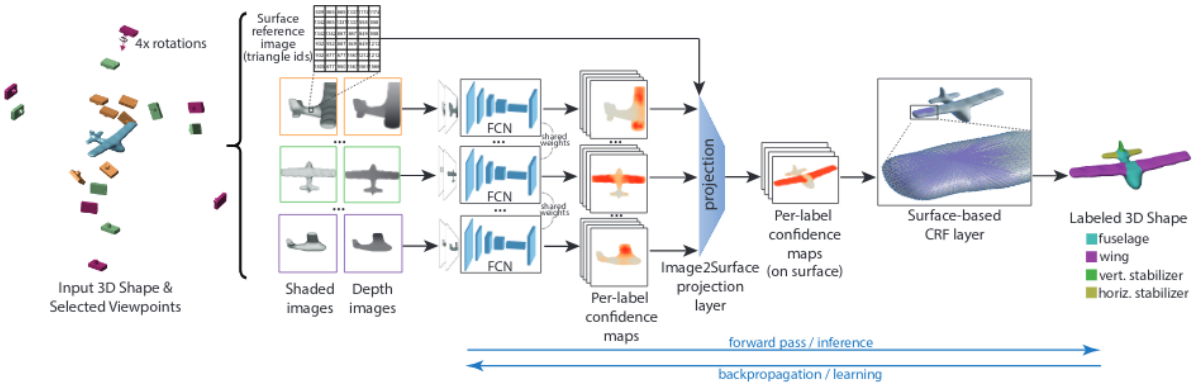


Figure 4.11: *Illustration of the network architecture for 3D shape segmentation with projective convolutional networks. Image from [201]*

These confidence maps are combined together and projected-back to the 3D surface using a a max pooling projective layer as illustrated in Figure 4.11. Finally the resulting confidence maps are converted to probabilities through a Conditional Random Field (CRF) layer defined on the surface favoring coherent labeling.

Multi-view based methods are more likely to outperform volumetric approaches in terms of efficiency as they require less computational cost. Nonetheless, this class of methods is not able to handle missing, occluded or overlapping objects especially in a mobile mapping context where a scanner is moving in a fixed scene. That is why we believe this category of methods, as it is designed, is not suited to our application.

4.1.4.3 Graph approaches

Considerable effort has been directed towards applying deep neural networks on 3D data represented by a graph. Broadly speaking, among this set of methods, we can distinguish between two major approaches w.r.t. the type of convolution used inside these networks; *spectral*-based and *spacial*-based. As the key component for the success of deep learning starts with the convolution operator, these two categories of methods define respectively the convolution in the spectral domain and the spacial domain.

Spectral-filtering-based methods were first introduced by Bruna *et al.* [204] through Spectral CNN (SCNN) a deep network operating on graphs. The intuition behind this method is to use the spectral eigen-decomposition of the graph Laplacian to define convolution in the spectral domain. This way signal patches defined in the euclidean space which correspond to features of the graph nodes are mapped to the spectral domain by projection on the graph Laplacian eigenvectors. Thus the convolution operation scales node features w.r.t. the eigenbasis. It should be noticed however, spectral filtering is a non-local operation as it involves the entire graph which comes with a huge computational burden in addition to the heavy computational cost of the graph Laplacian. To address the aforementioned shortcoming, several studies have proposed a local spectral filtering operation to alleviate the computational cost by approximating graph filters using Chebyshev polynomials Defferrard *et al.* [205] or a first-order linear approximation Kipf *et al.* [206].

While this local filtering accounts for computational efficiency, all the aforedescribed spectral methods share a common weakness. In practice, the dependency of spectral filters on the eigenbasis limits the generalization capability of the network. As demonstrated in the work of Bronstein *et al.* [3], applying a spectral filter kernel learned w.r.t. to a specific basis on another domain with different basis yields completely different results.

On the other hand, spatial-filtering-based methods are a much simpler attempt to apply deep learning on data represented by a graph. In these methods, the support for the graph convolution operation is the set of edges connecting neighboring nodes of the graph. The work of Scarselli*et al.* [207] is among the first studies applying neural networks on data structured by a graph. Graph Neural Networks (GNNs) are comprised of multiple layers through which local features are learned w.r.t. to the graph nodes. Each vertex is embedded using a Recurrent Neural Network (RNN) which repeatedly propagates features to the neighboring graph nodes until stability. Since this recurrent propagation constitutes a computational bottleneck, Li *et al.* [208] have introduced a different variant of RNN called *gated recurrent unit* (GRUs) which perform state updates more efficiently. A later work of Simonovsky *et al.* [202] extends these ideas by explicitly leveraging label edges to

perform convolution. Making use of edge labels information, by dynamically generating learnable edge weights while applying convolution, allows the network to benefit from the key properties of classic CNNs namely weight sharing and locality.

In general a graph is a versatile and powerful representation. We make use of this concept in our approach for partitioning 3D point clouds and 3D meshes.

4.1.4.4 Unordered point sets

3D data in its simplest form is represented by the mean of an unordered set of georeferenced 3D points. Most of the prior work applying deep learning on 3D point clouds have adopted the assumption that the network input has to be structured in a regular grid with an explicit connectivity information. Both OctNet [155] and VoxelNet [15], discussed earlier, structure point clouds respectively by means of an octree and a voxel grid to preserve the euclidean properties of data before feeding it to deep networks. A different category of approaches such as SnapNet [209] have proposed to first generate virtual views from the input 3D data leading to a set of 2D images where a wide variety of mature CNNs can be easily applied to perform the desired task (semantic segmentation, object detection, etc.). Once labels or features are extracted, they are reprojected-back to their original form in 3D space. Even though these projective methods arguably handle large scale data to some extent, this gain comes at the expense of the projection reliability. In practice, prominent geometric information is lost in the back and forth projections since 2D images are no more than a rasterization of 3D scenes. Figure 4.12 illustrates the full pipeline for 3D point cloud semantic segmentation.

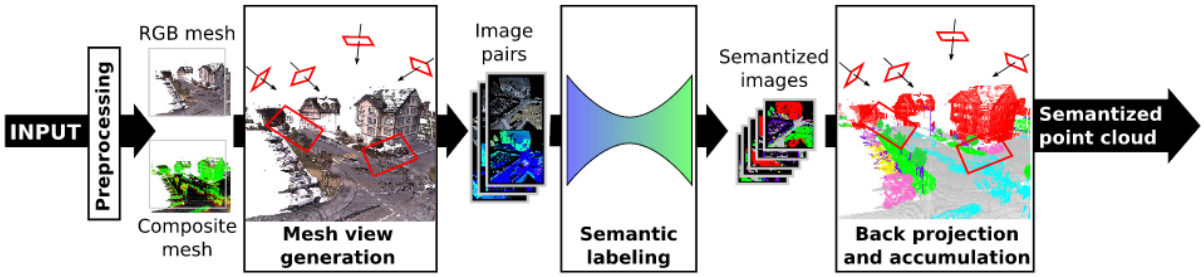


Figure 4.12: *Snapnet framework for point cloud semantic segmentation. Image from [209]*

The first pioneer work that directly make use of unordered points as input is PointNet [14]. This deep architecture differs significantly from the rest of the networks proposed in the literature by its simplicity. Instead of developing an adequate convolution operator with learnable filters, PointNet [14] have shown that using a network comprised exclusively of fully connected layers along with aggregation modules is sufficient for encoding convenient spatial information. In principle, PointNet is composed of three building blocks as illustrated in Figure 4.13; a spatial transform network (STN), a multi-layer perceptron and a symmetric function which consists of a max pooling operation.

STN module learns a canonical form of the input points and selects the most informative points characterizing the input shapes. MLP module is a succession of fully connected layers that generate a feature vector for each point independently. Finally the permutation invariant max pooling module aggregates the learned features into a global one which serve for classification purposes or segmentation after further processing. Despite the competitive results achieved by PointNet, the network as it is designed fail to

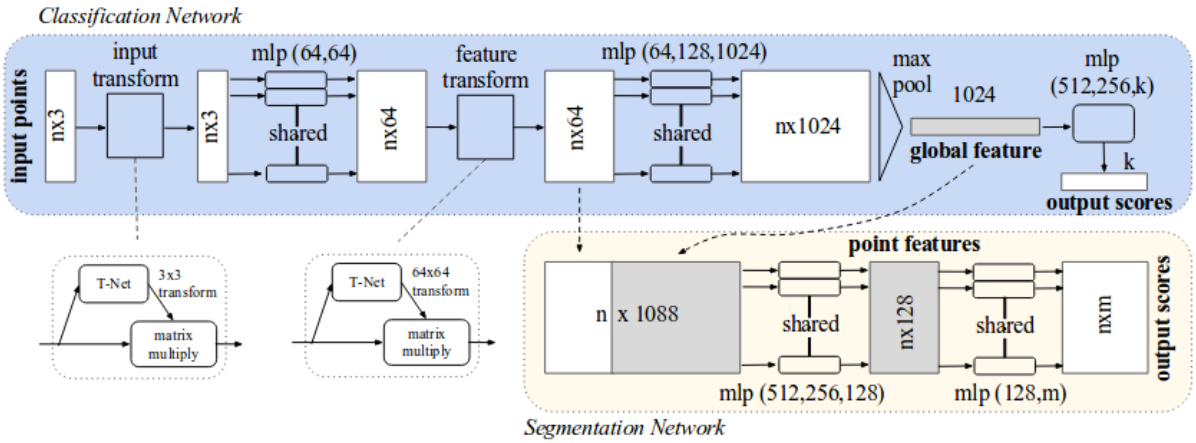


Figure 4.13: *PointNet architecture for 3D point clouds classification and segmentation. Image from [14]*

encode fine-grained patterns of each point and its local neighborhood. PointNet++ [153] mitigates this limitation by proposing a nested hierarchical portioning of points along the network. The intuition behind this idea is mainly inspired by 2D CNNs where the size of feature maps is progressively decreased along the network to learn features at various scales. First input points are partitioned into overlapping regions. Then features are learned at gradually increased neighborhood sizes such that smaller neighborhood capture fine grain features while larger ones encode global geometric shape features. Figure 4.14 illustrates the architecture proposed in [153].

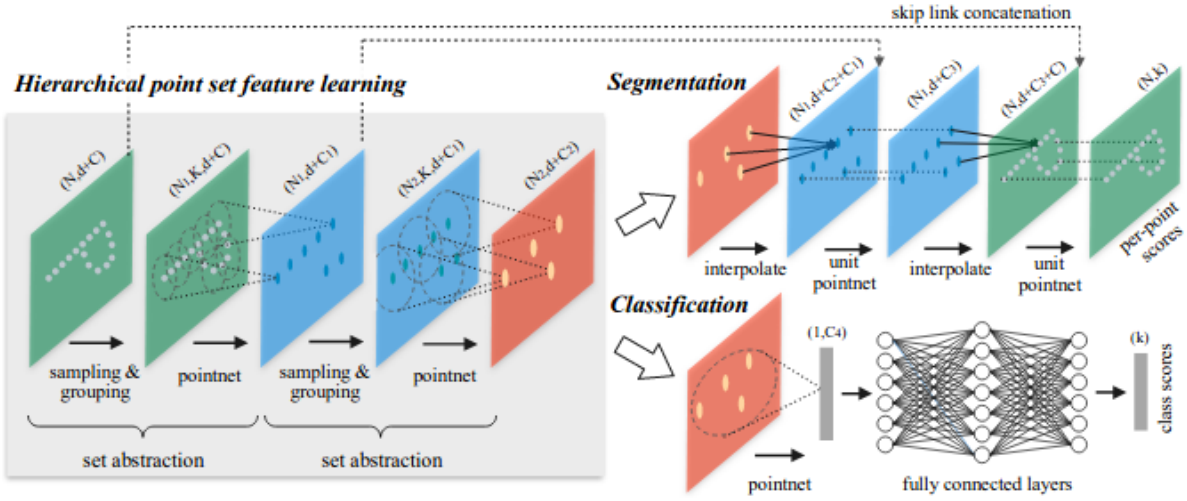


Figure 4.14: *PointNet++ hierarchical architecture for 3D point clouds classification and segmentation. Image from [153]*

Whereas PointNet++ outperforms PointNet in classification and segmentation tasks, the developed architecture is complicated and computationally not efficient compared to the gain in accuracy. Most importantly, both PointNet and PointNet++ share a common weakness which is their unability to handle large scale data. In theory, these architectures are efficient for man made shapes with few thousands of points which is not compatible

with applications requiring data at larger scales. In the context of outdoor navigation, the size of input data can easily reach millions if not billions of points especially for mobile mapping acquisition campaigns. A first interesting work that deals efficiently with scale problem while taking advantage of the new point embeddings method such as PointNet is Super-Point Graph (SPG) [17].

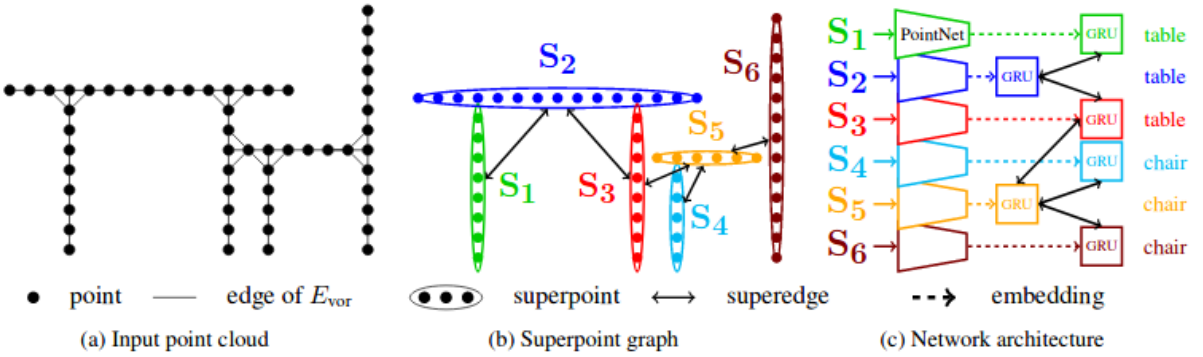


Figure 4.15: *SPG full pipeline for large scale 3D point clouds semantic segmentation. Image from [17]*

Landrieu and Simonovsky in [17] have proposed an innovative framework through which data is represented effectively, called superpoint graph. The full pipeline, as demonstrated in Figure 4.15 is composed of three major steps. First, an over-segmentation is computed over the entire point cloud resulting in a set of geometrically homogeneous segments. Second, the consequent primitives or so called superpoints are further downsampled and passed subsequently to a PointNet network for embedding. Finally to ensure that long range interactions are preserved between the computed superpoints while computing a contextual segmentation, a graph-based deep learning algorithm [202] performing graph convolution is applied on the adjacency graph having superpoints as nodes and their spatial relations as edges. Even though SPG framework has shown impressive results by improving state-of-the-art in the task of point cloud semantic segmentation, we believe that there is a large margin for improvement. As pointed out in Section 4.1.1, the over-segmentation in SPG framework is based on the work of Guinard *et al.* [157] leveraging handcrafted features for superpoints segmentation. As later steps rely essentially on the quality of the over-segmentation, we believe that this step should be also learned in a supervised deep framework to reduce potential errors.

4.1.4.5 3D Meshes

As discussed above, the success of a deep learning method applied to 3D data is mainly tied to how the convolution operator is appropriately adapted to the topology of data. 3D meshes in particular belong to the category of non-euclidean 3D representations where it is difficult to define the local convolution support on a 3D surface. Geodesic CNN [210] is among the first attempts to circumvent the lack of a support for convolution. In this work, Masci *et al.* have proposed to construct local patches in local polar coordinates as shown in Figure 4.16.

Afterwards, a geodesic convolution is performed by applying a filter kernel subject to angular coordinates at each vertex of the mesh within the constructed local patch.

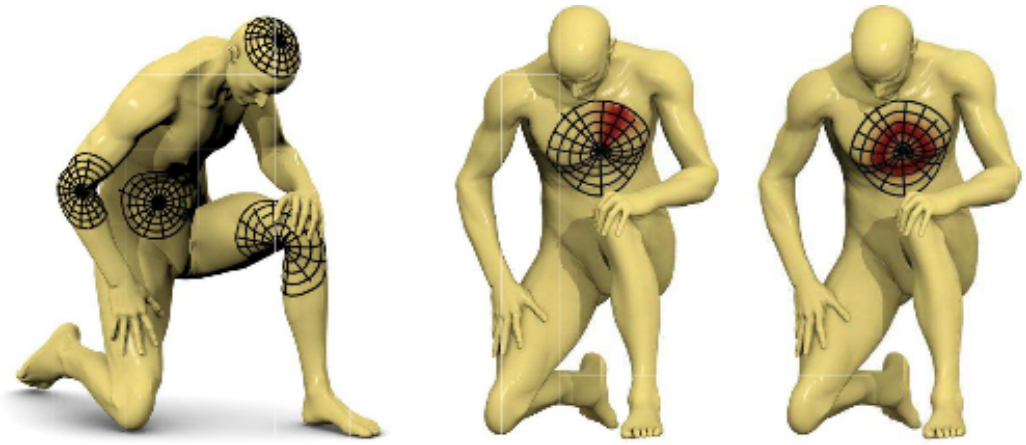


Figure 4.16: *Illustration of the local geodesic patches in polar coordinates on a manifold mesh. Image from [210]*

The proposed network is a succession of non-linearity layers, geodesic convolution layer followed by an angular max pooling in addition to a Fourier transform w.r.t. angular coordinates in order to remove arbitrary rotation ambiguity.

More recently, Huang *et al.* [26] introduced a consistently-oriented geodesic parameterization for a 3D surface associated with a high resolution signal. Called TextureNet, the network's input is a 3D textured mesh and its output is a set of learned features attached to sampled points on the surface. TextureNet deals with the lack of an unified and seamless local parameterization of the surface, by computing a four-fold rotationally symmetric field on the surface. This 4-RoSy field consists of a set of tangent directions attached to vertices that allows to consistently orient adjacent neighborhoods. A 4-RoSy convolution operation defined within this geodesic field is orientation invariant.

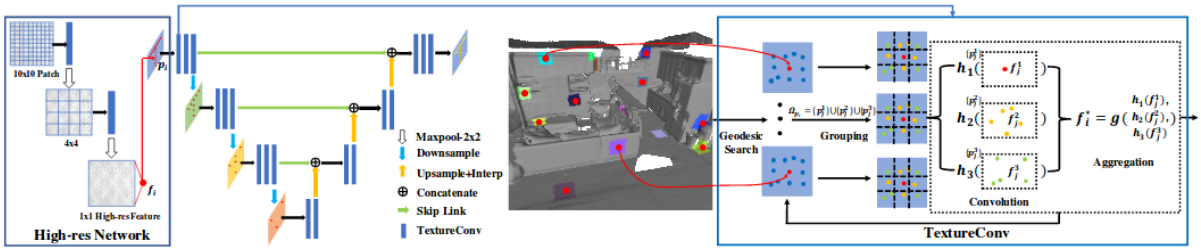


Figure 4.17: *TextureNet architecture. Image from [26]*

The proposed architecture illustrated in Figure 4.17 is a U-Net-like network [211] composed of an encoder and a decoder. In addition to pooling and non-linearity layers, the de facto core of TextureNet is the convolution layer. The latter is comprised of four blocks performing respectively a geodesic patch search followed by a texture grouping then a convolution and aggregation operations w.r.t. to the local geodesic patch.

While the aforedescribed convolution operators in [26, 210] offer interesting approaches for a consistent local surface parameterization in order to perform convolution, they remain extremely sensitive to the triangulation irregularities since they are attached to geodesic patches defined over the mesh surface not to mention the required computational cost when computed repeatedly during training.

A different approach introduced in Hanocka *et al.* [16], has proposed a network that is explicitly designed for irregular triangular meshes. The key innovation of this method is that both convolution and pooling are defined on the mesh edges instead of local surface patches. Assuming that the input mesh is manifold with possibly boundary edges, this means that each edge is shared by exactly two faces (*i.e.* having four adjacent edges or two in case of a boundary edge).

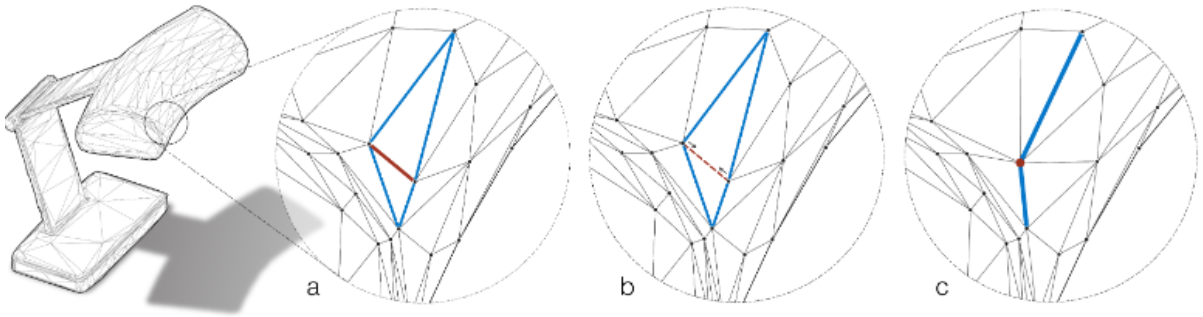


Figure 4.18: *Illustration of convolution and pooling operation on 3D meshes. Image from [16]*

For each edge, the convolution is performed w.r.t. to its 1-ring adjacent edges [2] as illustrated in Figure 4.18 (a). Such setting entails an ambiguity concerning the edge ordering. In essence, by considering a counter-clockwise ordering of vertices of the incident faces to the edge e, we can observe that there are two possible ordering namely (e_1, e_2, e_3, e_4) or (e_3, e_4, e_1, e_2) as shown in Figure 4.19. Such ambiguity hinders the capability of the convolution layer to learn invariant discriminative features. The desired properties of a convolution operator are mainly its invariance to similarity transformations including rotation, translation and scale. To fulfill these criteria, Hanocka *et al.* [16] attach to each edge relative features that are inherently invariant to rotation, translation and scale such as the dihedral angle between adjacent faces, two inner angles (α, β) and the ratios between the underlying edge and its perpendicular belonging to incident faces as illustrated in Figure 4.4.

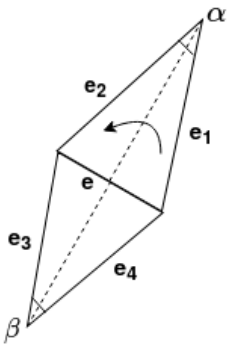


Figure 4.19: *Illustration of the hand-crafted features computed per-edge*

[2] the set of edges forming the faces incident to the underlying edge.

Afterwards symmetric functions are applied to feature edges depicting an ambiguity. We recall that a convolution is a correlation between a neighborhood and a template or kernel. Formally, according to [16], for an edge e and its 1-ring neighborhood e_1, e_2, e_3 and e_4 associated respectively with features f, f_1, f_2, f_3 and f_4 a mesh convolution with a kernel w is defined as:

$$f w_0 + \sum_{i=1}^{4} w_i f_i \quad \text{where} \quad (f_1, f_2, f_3, f_4) = (\mid e_1 - e_3 \mid, e_1 + e_3, \mid e_2 - e_4 \mid, e_2 + e_4) \tag{4.4}$$

Applying symmetric functions $\mid e_1 - e_3 \mid$ and $\mid e_2 - e_4 \mid$ on relative edge features remove the ordering ambiguity in convolution. Mesh pooling, on the other hand is performed using a task-aware edge collapse operation. In theory, once features are computed for each edge after few convolutions, the pooling consists in selecting which edges to be collapsed w.r.t. the learned task. As shown in Figure 4.18 (b) and (c), pooling consists in reducing five edges into two by first collapsing the edge with the lowest strength (the minimum ℓ_2 norm of its corresponding feature) and merging the remaining edges by averaging their corresponding features. Note that concretely the input mesh is not decimated while training, albeit the computed feature maps are the subject of pooling. This allows the network to capture key information at multiple scales akin to 2D CNNs. The performance of this approach has been demonstrated on classification and semantic segmentation tasks setting a new state-of-the-art. In our work we adapt this architecture to extract features of mesh faces.

4.1.5 Review conclusion

To conclude, this short review is far from being an extensive study as our goal is to introduce the most relevant work that inspired our approach. While the literature is full of surveys discussing 2D superpixels such as the recent work of Stutz *et al.* [163], over-segmentation of point clouds and 3D meshes is a less studied subject. Our review can be further extended to a more complete and comprehensive survey.

It should be noted that there is a rich literature regarding graph partitioning methods. We limited the definitions and algorithms presented in this section to the most related notions used in our work so that the reader do not get distracted with overwhelming irrelevant information. A pointer to further thorough studies is most of the time provided in case the reader is interested to know more about the underlying subject.

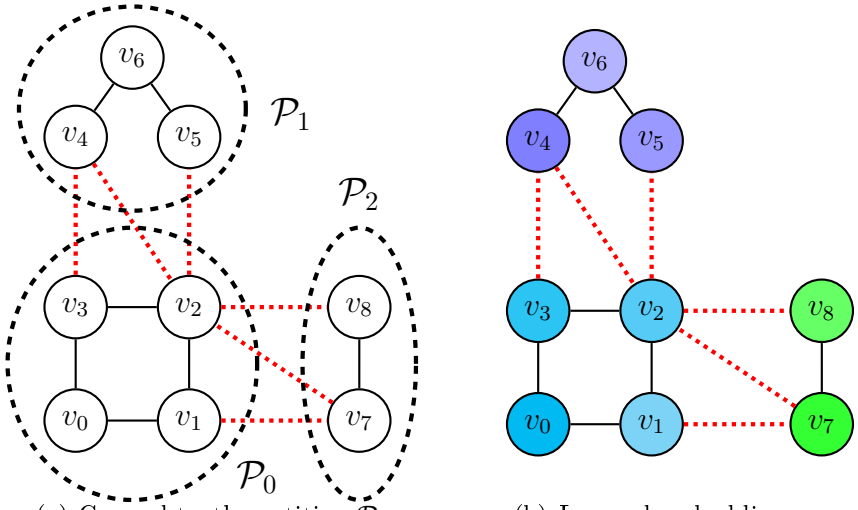
We believe deep learning on 3D point clouds in particular has witnessed a tremendous progress during the past two years. For a more thorough general study on deep learning applied to 3D data, the reader is referred to [199, 212]. All of the aforedescribed representations have demonstrated an outstanding performance especially PointNet [14], SPG [17] and meshCNN [16]. We make use of these methods to develop our over-segmentation framework for point clouds and 3D textured meshes.

4.2 Method

The goal of our method is to produce a high-quality over-segmentation of large scale 3D data represented as point clouds or 3D meshes structured by a graph in order to be used in turn to perform semantic segmentation. First, we build a graph on top of the data

points. The structure of this graph is mainly dependent on the geometry of data. Second, we partition the constructed graph supervised by the ground truth segmentation of data points. Finally, we combine our framework with a dense classification algorithm SPG [213] initiated by our over-segmentation to improve semantic segmentation results.

Concretely, our over-segmentation method is a two step approach. First, we learn vertex embeddings in such way they are homogeneous within segments and present high contrast at their borders. Second, we compute a piece-wise constant approximation of these learned embeddings with respect to an adjacency graph as illustrated in Figure 4.20.



(a) Ground truth partition $\mathcal{P}$ (b) Learned embeddings e.

Figure 4.20: *Illustration of our method: vertex embeddings are learned such that they are homogeneous within ground truth segments $\mathcal{P}$, and with high contrast at transition edges (in red).*

Formally, let V be a set of data points (3D points of a point cloud or triangles of a 3D mesh) whose adjacency structure is encoded by the graph $G = (V, E)$, with $E \subset V \times V$ the set of edges. We assume that this adjacency structure is *sparse*, in the sense that $|E| \ll |V|^2$. For a partition $\mathcal{U} = (U_1, \cdots, U_K)$ of V, we denote $E_{\text{trans}}(\mathcal{U})$ its set of transition edges, *i.e.* the set of inter-edges linking different elements of $\mathcal{U}$: $E_{\text{trans}}(\mathcal{U}) = \{(u, v) \in E \mid u \in U_i, v \in U_j, i \neq j\}$. Inversely, the set of intra-edges *i.e.* linking points within the same segment is defined as $E \setminus E_{\text{trans}}(\mathcal{U})$. First, we start by associating to each vertex v in V an embedding e_v in the m-dimensional unit sphere $\mathbb{S}_m = \{x \in \mathbb{R}^m \mid \|x\| = 1\}$. In the following section 4.2.1 we discuss how such an embedding can be computed. Second, we partition V into the constant connected component of a piecewise-constant approximation of e_v with respect to the graph G. The latter step is discussed in Section 4.2.2.

4.2.1 Learning embeddings

Our objective is to learn a vertex embedding function $\xi : V \mapsto \mathbb{S}_m$ such that $\xi(V)$ is homogeneous within the segments of the ground truth segmentation $\mathcal{P}$, and with high contrast at transition edges $E_{\text{trans}}(\mathcal{P})$. Note that as advocated by Wang *et al.* [214] constraining these embeddings to be within the m-unit sphere $\mathbb{S}_m$ not only prevents collapse during the training phase but also normalizes the distance between the embeddings.

As demonstrated by Hornik in [215], neural networks are universal approximators for arbitrary finite input measures. With the huge success of neural networks in large variety

of computer vision tasks, the function ξ would be typically a neural network operating on features of the data points corresponding to the vertices V of G. These input features can be pixel colors in 2D images, the local geometry/radiometry of points or triangles in respectively a 3D point cloud or a 3D mesh.

In 2D images, [216] have shown that taking descriptors as the activations of Convolutional Neural Networks (CNNs) layers outperforms all the hand-engineered features. Following this work [160] proposed to use a classic encoder-decoder backbone to compute features from 2D images which serves subsequently to learn task-specific superpixels. The architecture used in [160] is illustrated in Figure 4.21. The computed features are the result of concatenating features from the last layer with *XYLab* features of each pixel.

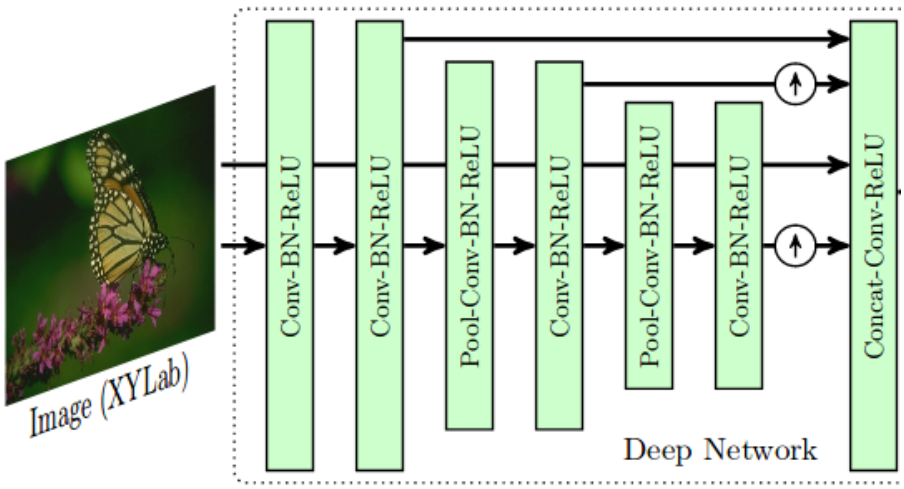


Figure 4.21: *The proposed network is composed of a stack of convolutional layers interleaved with Batch norm (BN) and non-linearity (ReLU) layers. Image from [160]*

While it seems simple to embed pixels of 2D images, in the 3D domain however, extracting local features of 3D data is not straightforward. With the recent success of global deep architectures for 3D point cloud and 3D mesh classification especially PointNet [14] and MeshCNN [16] discussed in Section 4.1.4, we propose an adaptation of these two methods respectively in Section 4.3 and Section 4.4 in order to compute dense per-point features for 3D point clouds and per-triangle features for 3D meshes.

Once these embeddings are computed, the over-segmentation is defined subsequently with respect to the adjacency graph G build upon the 3D data.

4.2.2 The generalized minimal partition problem (GMPP)

In our method, the over-segmentation is cast as a graph partitioning problem. We define these segments by computing the constant connected components in the graph G of a piecewise-constant approximation of the embeddings $e_v \in \mathbb{S}_m^V$. This approximation is the solution $f^\star$ of the following optimization problem:

$$f^\star = \underset{f \in \mathbb{R}^{V \times m}}{\arg\min} \sum_{v \in V} \|f_v - e_v\|^2 + \sum_{(u,v) \in E} w_{u,v} \left[f_u \neq f_v\right], \tag{4.5}$$

with $w \in \mathbb{R}_+^E$ the edges' weight and $[x \neq y]$ the Iverson's bracket equal to 0 if $x = y$ and 1 otherwise. To encourage splitting along high contrast areas, we define the edge weight as

$$w_{u,v} = \lambda \exp\left(\frac{-1}{\sigma}\|e_u - e_v\|^2\right), \quad \lambda, \sigma \in \mathbb{R}^+ \text{ parameters} \tag{4.6}$$

Known as the *Generalized Minimal Partition Problem* GMPP and introduced by [178], the equation (4.5) is neither continuous, differentiable, nor convex, and therefore the global minimum cannot be realistically retrieved. One way to get good approximate solution efficiently consists in using the ℓ_0-cut pursuit algorithm [178] discussed in Section 4.1.3. Note that the edges' penalty defined in equation (4.6) automatically implements ($\mathcal{P}3$) for reasonable parameterization of the problem. Thus we can define $\mathcal{S}^{(e)}$ as the segmentation given by the connected components of the approximate solution $f^\star$ of equation (4.5) for a given embedding e. In order to learn vertex embeddings and partition the graph G accordingly, an objective function, which reduce gradually the error between the ground truth segmentation $\mathcal{P}$ and the predicted one $\mathcal{S}$ during optimization, should be carefully designed.

4.2.3 Graph structured contrastive loss

As mentioned earlier, the object purity property ($\mathcal{P}1$) is the first quality of an accurate over-segmentation. A straightforward way to learn such an embedding function ξ would be to choose a metric estimating the object purity as a loss function. In the literature, ($\mathcal{P}1$) can be evaluated by measuring the agreement between a ground truth segmentation $\mathcal{P}$ and the predicted segmentation $\mathcal{S}$ using the *under-segmentation error* [163]. This error sums over each segment $S \in \mathcal{S}$ the number of vertices which are not in the *majority segment*, *i.e.* the element of $\mathcal{P}$ with the largest overlap with S. Formally, the under-segmentation error $\mathcal{L}(\mathcal{P}, \mathcal{S})$ can be defined as:

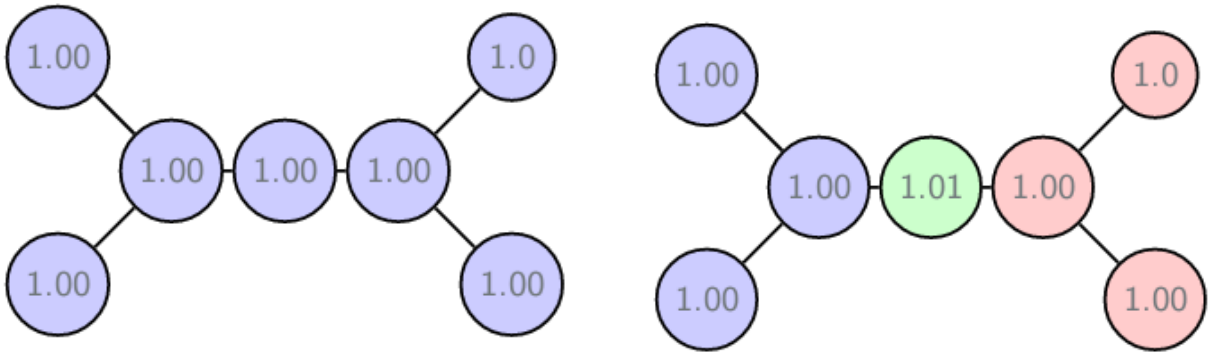
$$\mathcal{L}(\mathcal{P}, \mathcal{S}) = \frac{1}{|V|} \sum_{S \in \mathcal{S}} \min_{P \in \mathcal{P}} |S \setminus P| \ . \tag{4.7}$$

However directly back-propagating through the minimization of $\mathcal{L}(\mathcal{S}^{(\xi(V))}, \mathcal{P})$ is difficult, if not impossible for several reasons. First, the GMPP defined in (4.5) used to compute the predicted partition $\mathcal{S}$ is non-continuous and non-convex. Second, computing the connected components on a graph is inherently non-differentiable since this operator is discontinuous as a single tiny change may overhaul the entire partition as illustrated in Figure 4.22.

Instead, we note that if the border recall property ($\mathcal{P}2$) is implemented (*i.e.* predicted segments and ground truth objects share the same boundaries), then ($\mathcal{P}1$) ensues. Therefore, we propose a surrogate back-propagable loss called *the graph-structured contrastive loss* focusing on correctly detecting the borders between objects and operates on edges instead of vertices. Our loss is defined as:

$$\ell(e, \mathcal{P}) = \frac{1}{|E|}\left(\sum_{(u,v) \in E \setminus E_{\text{trans}}(\mathcal{P})} \phi\left(e_u - e_v\right) + \sum_{(u,v) \in E_{\text{trans}}(\mathcal{P})} \mu_{u,v}^{(e)} \psi\left(e_u - e_v\right)\right), \tag{4.8}$$

with ϕ (resp. ψ) a function favoring similarity (resp. contrast), and $\mu_{u,v} \in \mathbb{R}^{E_{\text{trans}}}$ a weight on transition edges, discussed later.



(a) A single connected component

(b) a small change results in 3 different connected components

Figure 4.22: *Explanation of the non-differentiability of the connected component operator*

$$\begin{cases} \phi(x) = \delta(\sqrt{\|x\|^2/\delta^2 + 1} - 1) \text{ with } \delta = 0.3, \\ \psi(x) = \max(1 - \|x\|, 0), \end{cases} \tag{4.9}$$

In the spirit of the original contrastive loss [217] discussed in Section 4.1.2, our loss encourages embeddings of vertices linked by an intra-edge to be similar, while rewarding different embeddings when linked by a transition edge. Note that our ℓ is different from the original triplet loss [214, 218], as it involves all vertices within a graph (or a sub-graph) at once, and not just an anchor and related positive/negative examples. In this way, it bypasses the problem of example picking altogether. Indeed, the positive and negative examples are directly given by the adjacency structure set by E_{trans} and $E \setminus E_{\text{trans}}$. A vertex embedding function minimizing this loss will be uniform within elements of $\mathcal{P}$ and have high contrasts at $E_{\text{trans}}(\mathcal{P})$. Consequently, $E_{\text{trans}}(\mathcal{S}^{(e)})$ should be close to $E_{\text{trans}}(\mathcal{P})$.

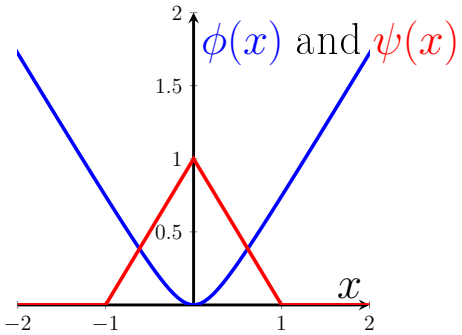


Figure 4.23: *The functions ϕ (in blue) and ψ (in red) used in the graph-structured contrastive loss.*

We chose ϕ, the function promoting intra-object homogeneity as represented in Figure 4.23. Indeed, the first term of ℓ is the (pseudo)-Huber graph-total variation [219, 220] on the intra-edges $(E \setminus E_{\text{trans}}(\mathcal{P}))$, promoting smooth homogeneity of embeddings within the same object.

ψ, the second part of ℓ is the opposite of the truncated graph-total variation [221] on the transition edges $E_{\text{trans}}(\mathcal{P})$. It penalizes similar embeddings at the border between

objects. Conscious that our embeddings are restricted to the unit sphere, we threshold this function for differences larger than 1. In other words, ψ encourages vertices linked by an inter-edge to take embeddings with an euclidean distance of 1, but does not push for a larger difference.

It should be emphasized that at this stage we do not try to learn semantic information, but rather to compute a signal on a graph such that its constant approximation respects certain properties. That is why, it may happens that objects of different classes can share the same embeddings as long as they are not adjacent.

4.2.4 Cross partition weighting

As explained earlier, if the ground truth segments and the predicted ones share the same boundaries ($\mathcal{P}2$), this will guarantee the object purity ($\mathcal{P}1$). While ($\mathcal{P}2$) does imply ($\mathcal{P}1$), tiny errors in the former can have drastic consequences on the latter. In practice, if one transition edge is missed, this can erroneously fuse two large segments covering different objects. Without an appropriate weighting strategy, ℓ will fail to take into account the effect of each edge which is susceptible to be largely varying in terms of undersegmentation error $\mathcal{L}$ (4.7). By omitting $\mu^{(e)}_{u,v}$ from the second part of equation (4.9), ℓ will only have a high accuracy in recovering transition edges while being agnostic to the influence of each $e \in E_{\text{trans}}$. In the literature, [173] proposed to incorporate the object purity by introducing a weighting scheme to the so called segmentation-aware affinity loss (SEAL). In this strategy for an edge (u, v) in a segment S of the predicted partition $\mathcal{S}$, μ implements directly the under-segmentation error (4.7):

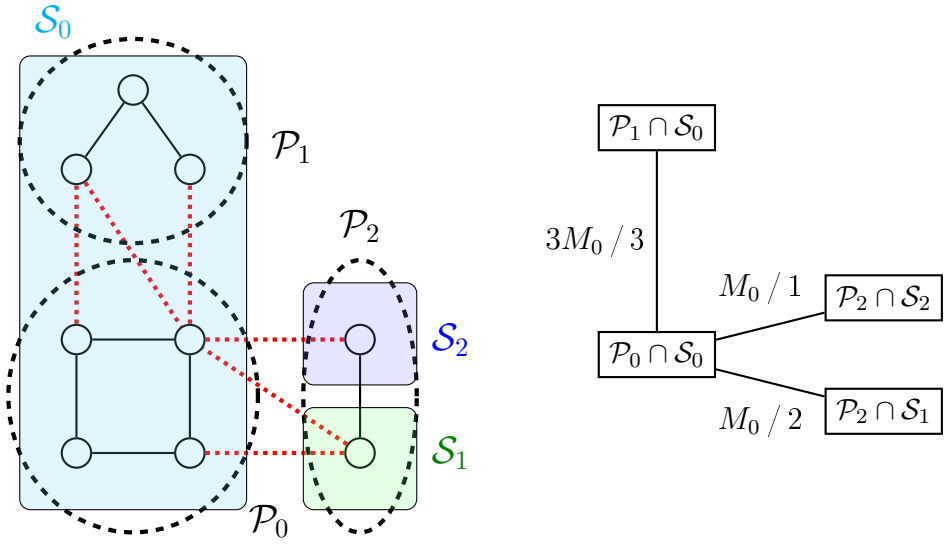
$$\mu_{u,v} = 1 + |S\ | - \min_{P \in \mathcal{P}} |S \setminus \mathcal{P}| \tag{4.10}$$

Although [173] show impressive results for superpixel segmentation, we were not able to replicate this success to superpoint/superfacet segmentation. In fact, we argue that the edge weights are evenly shared by all transition edges of a given segment regardless of their influence on the purity and the size of the interface. This excessively favors long interfaces in the loss, and inadequately handles large segments with multiple interfaces. Moreover, as soon as a segment no longer overlaps the object's border, its weight is decreased to 1 making thereby the loss unstable during training.

In order for ℓ (4.9) to better represent $\mathcal{L}$ (4.7), the edge weights $\mu \in \mathbb{R}_+^{E_{\text{trans}}(\mathcal{P})}$ should be set wisely to reflect this influence. To this end, we introduce the *cross-partition weighting strategy*. First, we build the *cross-segmentation graph* $\mathcal{G}^{(e)} = (\mathcal{C}^{(e)}, \mathcal{E}^{(e)})$, represented in Figure 4.24 and defined as the adjacency graph of the cross-partition between the ground truth object partition $\mathcal{P}$ and the predicted partition $\mathcal{S}^{(e)}$. In other words, $\mathcal{C}$ is the set of connected components of the graph G when all edges either between objects *or* between segments are removed, and the *super-edge* (*i.e.* set of edges) $(U, V) \in \mathcal{E}$ is the set of transition edges of E_{trans} between U and V in $\mathcal{C}$:

$$\begin{aligned} \mathcal{C}^{(e)} &= \left\{ P \cap S \mid P \in \mathcal{P}, S \in \mathcal{S}^{(e)} \text{ and } P \cap S \neq \varnothing \right\} \\ \mathcal{E}^{(e)} &= \left\{ (U, V) \in {\mathcal{C}^{(e)}}^2 \mid U \times V \cap E_{\text{trans}} \neq \varnothing \right\}. \end{aligned} \tag{4.11}$$

We associate the following weight $M^{(e)}_{U,V}$ to each super-edge (U, V) of $\mathcal{E}^{(e)}$ and $\mu^{(e)}_{u,v}$ to



(a) Proposed partition $\mathcal{S}$ (b) Cross-partition graph $\mathcal{G}$.

Figure 4.24: *Illustration of the proposed superedge weighting scheme. Left, we represent an erroneous proposed partition $\mathcal{S}$ with 3 segment. Right we represent the cross partition graph $\mathcal{G}$ with the edge weights $M_{U,V} / |U \times V \cap E_{trans}|$.*

each transition edge:

$$M^{(e)}_{U,V} = M_0 \min(|U|, |V|) \text{ for } (U,V) \in \mathcal{E}^{(e)} \tag{4.12}$$
$$\mu^{(e)}_{u,v} = \frac{M^{(e)}_{U,V}}{|\, U \times V \cap E_{\text{trans}} \,|} \text{ for } (u,v) \in U \times V \cap E_{\text{trans}}.$$

with M_0 a parameter of the model. These weights take into consideration not only the influence of the edges in the purity but also the shape of the interfaces. Indeed, the under-segmentation error caused by an erroneous fusion of two segments U and V is proportional to $\min(|U|, |V|)$. This is explained by the definition of $\mathcal{E}^{(e)}$ which implies that two segments U and V of $\mathcal{C}^{(e)}$ linked by a super-edge $(U,V) \in \mathcal{E}^{(e)}$ are in two different ground truth segments.

We note that setting $M_0 = |E| \,/\, |V|$ gives the same importance to the classification of transition and non-transition edges. Indeed, assuming that most edges are non-transition, we have the sum of non-transition edge weights close to $|E|$. In an over-segmentation context, M_0 must be set higher to prioritize the recovery of object borders.

The weights are at last normalized by the number of edges constituting the interface between U and V in order to distribute evenly the penalty over the number of edges constituting an interface. This prevents long borders from being over-represented in the loss. In Figure 4.25, we further explain our cross-partition weighting strategy using a concrete example of a scene composed of a door (**D**) and a wall (**W**). Two segments **L** (left) and **R** (right) overlap the door. The super-edge (**LW**, **LD**)(resp. (**RW**, **RD**)) represent the adjacency between the part of the left (resp. right) segment covering the wall and the part covering the door. With fewer trespassing points and a longer interface than (**RW**, **RD**), the weights of the edges constituting (**LW**, **LD**) are smaller.

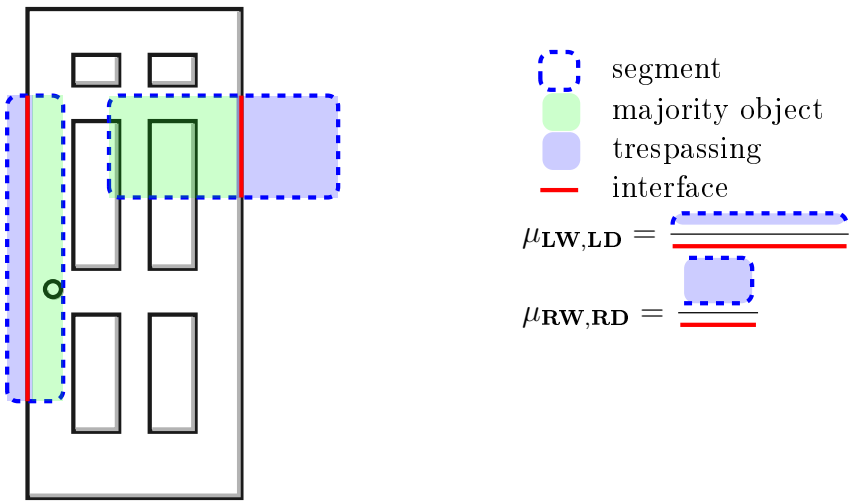


Figure 4.25: *Illustration of the cross-partition weighting strategy on a simplified scene*

4.3 Applications: 3D semantic map as point cloud

In this section, we first explain how to apply our over-segmentation method to 3D point clouds of indoor and outdoor scenes. Second, we show how the learned superpoints can be successfully combined with a superpoint-based semantic segmentation algorithm to build a 3D semantic map. We start by introducing our local point embedder LPE, a simple neural network which associates each 3D point with a low dimensional embedding that captures its local geometry and radiometry.

4.3.1 3D point cloud embedder LPE

Let C be a 3D point cloud, such that each point i is defined by its position $p_i \in \mathbb{R}^3$ and d-dimensional radiometric information $r_i \in \mathbb{R}^d$ (this can be colors if available, or intensity of LiDAR scans, or be ignored if none is available). Each point i is associated with its local features P_i and R_i, respectively comprised of the position and radiometry of its k nearest neighbors N_i in the input cloud: $P_i = \{p_j \mid j \in N_i\}, R_i = \{r_j \mid j \in N_i\}$. For ease of notation, any operator or function f applied to a set of features X is to be understood as being applied to all its elements: $f(X) = \{f(x) \mid x \in X\}$.

The goal of our network is to associate each 3D point i with a compact m-dimensional embedding e_i characterizing its features (position, color) and the geometry and radiometry of its local neighborhood. The global architecture of the proposed Local Point Embedder (LPE) is illustrated in Figure 4.26:

Our architecture is a lightweight network inspired by PointNet [14]. However, unlike PointNet, LPE does not try to extract information from the whole input point cloud, but rather encodes each point based on purely local information. Our LPE is comprised of two parts; a spatial transform network and a lightweight PointNet-like network. In the following we detail each part of the proposed architecture.

4.3.1.1 Spatial transform network:

This unit takes the positions of a target point p_i and its local k-neighborhood P_i. First the neighbors' coordinates are normalized around p_i such that the standard deviation of

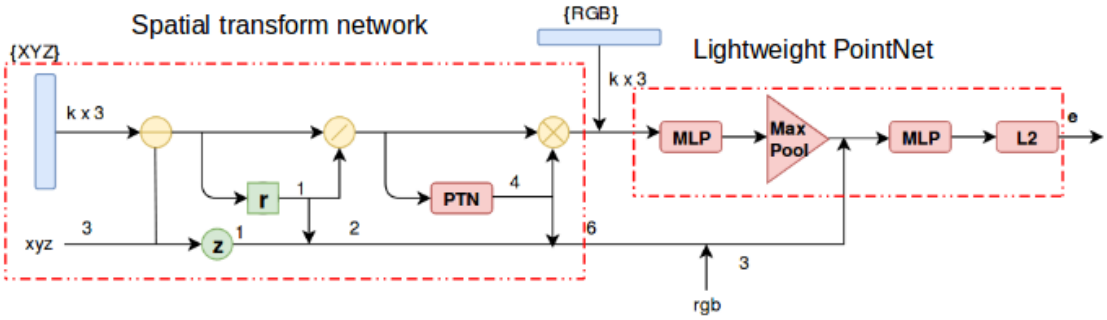


Figure 4.26: *Our LPE network used to embed each 3D point in a point cloud.*

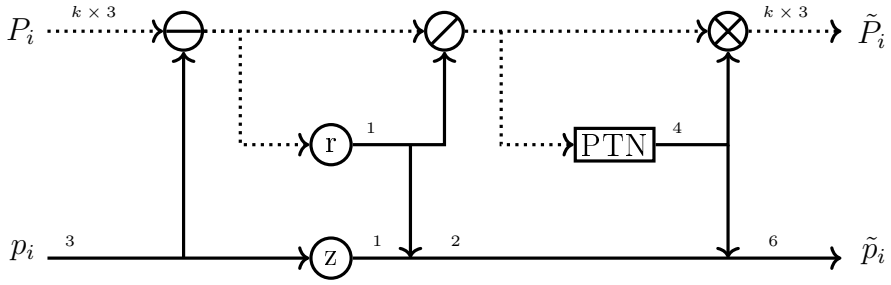


Figure 4.27: *Architecture of the spatial transform network.*

the point's position is equal to 1 (4.15). Then, this neighborhood is rotated around the z axis with a 2×2 rotation matrix computed by small PointNet network PTN (4.16). As advocated by [222], these steps aim to standardize the position of the neighborhood clouds of each point. By doing so, the next network will be able to learn position distribution. Along the normalized neighborhood position $\tilde{P}_i$, this unit also outputs geometric point-features $\tilde{p}_i$ describing the elevation $p_i^{(z)}$, the neighborhood radius, as well as its original orientation (through the 4 values of the rotation matrix: $[\Omega_{x,x}, \Omega_{x,y}, \Omega_{y,x}, \Omega_{y,y}]$)(4.17). By keeping track of the normalization operations, the embedding can remain covariant with the original neighborhood's radius, height, and original orientation, even though the points' positions have been normalized and rotated.

In Figure 4.27 we expose the details of the spatial transform part. the network takes a point's coordinate as point-input p_i and the coordinates of its neighbors as set-input P_i. The vertex r computes the radius of a point cloud (4.13), the vertex z extract the vertical coordinate of a point's position, and the vertex PTN is a small PointNet-like network (4.14) which outputs a 2×2 rotation matrix around the z axis (4.16). In this and subsequent figures, set-features (respectively point-features) are represented by a dotted line (respectively a solid line). The numbers above the lines represent the size of the channels.

$$\text{rad} = \text{std}\,(P_i) \tag{4.13}$$

$$\Omega = \text{PTN}(\tilde{P}_i) \tag{4.14}$$

$$P'_i = (P_i - p_i)/\text{rad} \tag{4.15}$$

$$\tilde{P}_i = \{p \times \Omega \mid p \in P'_i\} \tag{4.16}$$

$$\tilde{p}_i = [p_i^{(z)}, \text{rad}, \Omega] \tag{4.17}$$

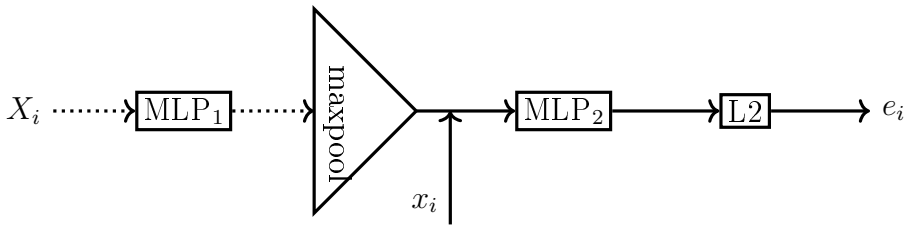


Figure 4.28: *Architecture of the second part of our LPE (4.19), which computes an embedding set-feature X_i and point-feature x_i encoding the local radiometry and the normalized geometry. The L_2 block normalizes the output on the unit sphere (4.18).*

4.3.1.2 PointNet-like embedder

The embeddings e_i are computed for each point i of C through a shared lightweight PointNet-like network (4.20). The input set-feature X_i is set as the concatenation of the neighbour's transformed position $\tilde{P}_i$ and their radiometric information R_i, while the input point-feature x_i is composed of the neighborhood geometric point-feature $\tilde{p}_i$ and the radiometry r_i of point i.

$$\mathrm{L}_2(\cdot) = \cdot / \|\cdot\| \tag{4.18}$$

$$\mathrm{LPE}(X_i, x_i) = \mathrm{L}_2\left(\mathrm{MLP}_2\left(\left[\max\left(\mathrm{MLP}_1(X_i)\right), x_i\right]\right)\right) \tag{4.19}$$

$$e_i = \mathrm{LPE}([\tilde{P}_i, R_i], [\tilde{p}_i, r_i]) \tag{4.20}$$

The embeddings computed using our LPE architecture are optimized through our graph-structured constrastive loss discussed in Section 4.2.3 supervized by the ground truth segments until they are homogeneous within objects and present high contrast at their borders.

4.3.2 Residual Point Embedder

We have tested an alternative configuration for the local point embedder LPE, in which they were stacked in layers, similarly to the classical convolutional architecture for images. We first introduce a slightly changed architecture, the Residual Point Embedder RPE, whose design is based on an LPE but takes a supplementary input e_{ini}. Instead of computing a new embedding, the RPE computes a residual (4.21) which is added to this initial embedding before normalization (4.22):

$$R(X_i, x_i) = \mathrm{MLP}_2\left(\left[\max\left(\mathrm{MLP}_1(X_i)\right), x_i\right]\right) \tag{4.21}$$

$$\mathrm{RPE}(x_i, X_i, e_{\text{ini}}) = \mathrm{L}_2\left(e_{\text{ini}} + R(X_i, x_i)\right) \tag{4.22}$$

The second change is the layers architecture. The RPEs in the first layer compute the embeddings from the local geometric and radiometric information alone, and their initial embedding is set to 0 (4.23) (such that they behave exactly like LPEs). The RPEs in subsequent layers compute new embeddings from the local radiometry and geometry as well as the embeddings computed at the previous layer of the points neighbors E_i^t (4.24). Note that for a point to be processed by a layer, all its neighbors must have been embedded by the previous layer. This allows the RPEs to have increasingly broader receptive fields, and to correct errors that might have been done by previous layers. Note

that the geometric information are only processed by the spatial transform once, cascading its values to all residual layers.

$$e_i^{(0)} = \text{RPE}^{(0)}([\tilde{P}_i, R_i], [\tilde{p}_i, r_i], 0) \tag{4.23}$$

$$e_i^{(t+1)} = \text{RPE}^{(t)}([\tilde{P}_i, E_i^{(t)}], [\tilde{p}_i, r_i, e_i^{(t)}], e_i^{(t)}) \tag{4.24}$$

Alternatively, all initial embeddings can be set to 0, which means that each layer computes a new embedding from the local position and the embeddings of the previous layers.

4.3.3 Numerical experiments

In this section, we show first our numerical experiments for the task of over-segmentation on two different datasets. Second, we combine the learned superpoints with the method of [17] to perform semantic segmentation.

4.3.3.1 Datasets

We evaluate our full pipeline of over-segmentation and semantic segmentation on two different public datasets:

- **S3DIS** [23]: this dataset is composed of 6 large-scale indoor areas from 3 different buildings. These areas show essentially office style architecture including conference rooms educational spaces and hallways. S3DIS is a large scale dataset composed of 600 million 3D RGB points acquired automatically using the Matterport scanner. Ground truth annotations are provided for 13 semantic classes of structured elements. These annotations include furniture items and commonly known classes such as door, window, ceiling, floor, chair, sofa, table, etc. Most importantly, in addition to per-point semantic labels, S3DIS offers object annotations which is a key enabler for our supervized over-segmentation method.

- **vKITTI 3D** [223]: built upon the original KITTI dataset [49], vKITTI is a virtual dataset comprised of nearly 15 million 3D points constructed by projecting video sequences into 3D space using camera parameters. Ground truth annotations are provided only for semantic labels of points with respect to 13 classes (terrain, road, sidewalk, road sign, etc.). In order to get a ground truth partition of objects for training, we compute the connected components of the semantic labels of points.

Both datasets are composed of 6 parts which allows us to use 6-fold cross-validation for evaluation. For efficiency, We subsample the two datasets using a regular grid of voxels (3cm wide for S3DIS and 5cm wide for vKITTI) as a preprocessing step before over-segmentation. In each voxel, we average the position and color of the contained points. This allows us to decrease the computation time and memory load.

4.3.3.2 Implementation details

In all our experiments, we set m the dimension of the computed embedding by LPE to 4. The architecture of LPE is a lightweight PointNet-like network operating on the $k = 20$ nearest neighborhood points of each point with no more than 15000 parameters for efficiency reasons. To prevent the creation of many small superpoints in regions of

high contrast, we modified the ℓ_0-cut pursuit algorithm [3][178] by merging components greedily with respect to the objective energy defined in (4.5), as long as they are smaller than a given threshold.

In order to build more robust networks, we added Gaussian noise of deviation 0.03 clamped at 0.1 on the normalized position and color of neighborhood clouds. We also added random rotation of the input clouds for the network to learn rotation invariance. To preserve orientation information, the clouds are rotated as a whole instead of each neighborhood. This allows the spatial transform to detect change in orientation, which can be used to detect borders.

To limit the spatial extent of the superpoints we concatenate to the points' embeddings their 3D coordinates in (4.5) multiplied by a parameter α_{spatial}, in the manner of [18]. This determines the maximum size that superpoints can reach. To automatically select a minimal superpoint size (in number of points) appropriate to the coarseness of the segmentation, we heuristically set:

$$n_{\min}^{\tilde{\lambda}} = \left[(\max \left(\frac{1}{2} n_{\min}^{(1)}, n_{\min}^{(1)} + \frac{1}{2} n_{\min}^{(1)} \log(\tilde{\lambda}) \right) \right] \tag{4.25}$$

where $n_{\min}^{(1)}$ is a dataset-specific minimum superpoints size for $\tilde{\lambda} = 1$. For example, for $n_{\min}^{(1)} = 50$, the smallest superpoint allowed for a small regularization strength $\tilde{\lambda} = 0.2$ will be 33, while it is 70 for the coarse partition obtained with $\tilde{\lambda} = 6$. While specific applications may require setting up this variable manually, this allowed us to produce the regularization paths in Figure 4.29 while only varying $\tilde{\lambda}$.

parameter	shorthand	section	S3DIS	vKITTI
Local neighborhood size	k	4.3.1	20	
# parameters	-	-	13,816	
LPE configuration	-	4.3.1	[32,128],[64,32,32,m]	
STN configuration	-	4.3.1	[16,64],[32,16,4]	
Embeddings dimension	m	4.3.1	4	
Adjacency graph	G	4.2.2	5-nn	5-nn + Delaunay
exponential edge factor	σ	4.2.2	0.5	
intra-edge factor	$\tilde{\mu}$	4.2.4	5	
spatial influence	α_{spatial}	4.3.3	0.2	0.02
smallest superpoint	$n_{\min}^{(1)}$	4.3.3	40	10
epochs	-	-	50	
decay event	-	-	20,35,45	

Table 4.2: *Configuration of the embedding network for the S3DIS and vKITTI datasets.*

We show in Table 4.2 the size of the linear layers, before and after the maxpool operation. Over 250,000 points can be embedded simultaneously on 11GB RAM in the training step, while keeping track of gradients.

4.3.3.3 Over-segmentation results

- **Evaluation metrics:** In the literature, several evaluation metrics were proposed to assess qualitatively and quantitatively an over-segmentation with respect to the afore-

[3] https://github.com/loicland/cut-pursuit

mentioned properties ($\mathcal{P}1$), ($\mathcal{P}2$), and ($\mathcal{P}3$). For point clouds, the Boundary Recall (BR) and Precision (BP) are used to evaluate the ability of the superpoints to adhere to, and not cross, object boundaries (($\mathcal{P}2$), ($\mathcal{P}3$)). However, these measures are defined with respect to *boundary pixels* [169] or points [165]. We argue that for point clouds transition occurs *between* points and not *at* points. That is why we think using edges instead of points to compute these metrics is more convenient. To this end, we define $E_{\text{trans}}^{\text{pred}}$ the set of predicted transition, *i.e.* the subset of edges of E that connect two points of C in two different superpoints. These metrics are often given with respect to a tolerance, *i.e.* the distance at which a predicted transition must take place from an actual object's border for the latter to be considered retrieved. We set this distance to 1 edge, which leads us to define $E_{\text{trans}}^{(1)}$ the set of inter-edges expanded to all directly adjacent edges in E:

$$E_{\text{trans}}^{(1)} = \{(i,j) \in E \mid \exists (i,k) \text{ or } (j,k) \in E_{\text{trans}}\}. \tag{4.26}$$

This allows us to define the boundary recall and precision with 1 edge tolerance for a set of predicted transition $E_{\text{trans}}^{\text{pred}}$:

$$BR = \frac{\mid E_{\text{trans}}^{\text{pred}} \cap E_{\text{trans}}^{(1)} \mid}{\mid E_{\text{trans}} \mid}, \; BP = \frac{\mid E_{\text{trans}}^{\text{pred}} \cap E_{\text{trans}}^{(1)} \mid}{\mid E_{\text{trans}}^{\text{pred}} \mid}. \tag{4.27}$$

In addition to BR and BP, other metrics were introduced to assess the purity of objects in an over-segmentation. The achievable segmentation accuracy (ASA) [166] is an upper-bound measure that gives the total effective segmentation area $\mathcal{S}$ with respect to a ground truth partition $\mathcal{P}$. Formally ASA is defined as:

$$ASA(\mathcal{S}, \mathcal{P}) = \frac{1}{N} \sum_{S_j \in \mathcal{S}} \max_{P_i \in \mathcal{P}} \mid S_j \cap P_j \mid \tag{4.28}$$

We introduce an analog metric to ASA called the *Oracle Overall Accuracy* (OOA) which characterizes the accuracy of the labeling that associates each superpoint S of a segmentation $\mathcal{S}$ with its majority ground-truth label. Formally, let $l \in \mathcal{K}^C$ be the semantic labels of each point within a set of classes $\mathcal{K}$, we define the OOA of a point cloud segmentation $\mathcal{S}$ as:

$$\begin{aligned} l^{\text{oracle}}(S) &= \text{mode}\{l_i \mid i \in S\} \\ OOA &= \frac{1}{\mid C \mid} \sum_{S \in \mathcal{S}} \sum_{i \in S} \left[l_i = l^{\text{oracle}}(S) \right], \end{aligned} \tag{4.29}$$

with $[x = y]$ the function equal to 1 if $x = y$ and 0 otherwise. Note that the OOA is closely related to the ASA [166], but consider the majority labels of all points within a superpixel rather than the label of the objects with most overlap. In this sense, it is a tighter upper bound to the achievable accuracy of a superpoint-based semantic classification algorithm. This metric is also more fair than the under-segmentation error [224] for other methods such as [157], or our cluster-based approach, as they do not try to retrieve objects directly, but rather regions of C with homogeneous semantic labeling.

- **Competing methods:** We denote our method by **SSP** which stands for *Supervised SuperPoints*. We compare our approach to the following methods:

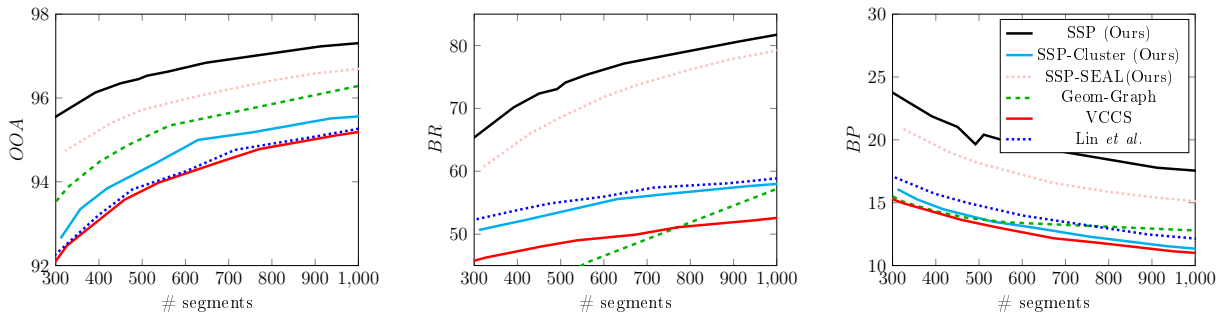
  SSP-cluster: is our adaptation of the soft clustering-based partition approach of Jampani *et al.* [160] to the 3D setting which was initially designed for 2D superpixels as discussed in Section 4.1.1.

  SSP-SEAL: uses the same supervized graph-based over-segmentation as **SSP**, albeit the cross-partition weighting scheme in equation (4.12) is replaced by the SEAL weighting strategy proposed by Jampani *et al.* in [173] in equation (4.10) as described in Section 4.2.4. Note that this is *not* equivalent to the framework of [173] discussed in Section 4.1.1, as they use a different loss and clustering algorithm.

  Geom-graph: is the graph-based method introduced by Guinard *et al.* [157] solving (4.5) on handcrafted features [225] instead of learned ones.

  VCCS: is the octree-structured cluster-based method introduced by Papon *et al.* [169]and discussed in Section 4.1.1.

  Lin *et al.* : is the adaptive resolution graph-based method introduced by [165].

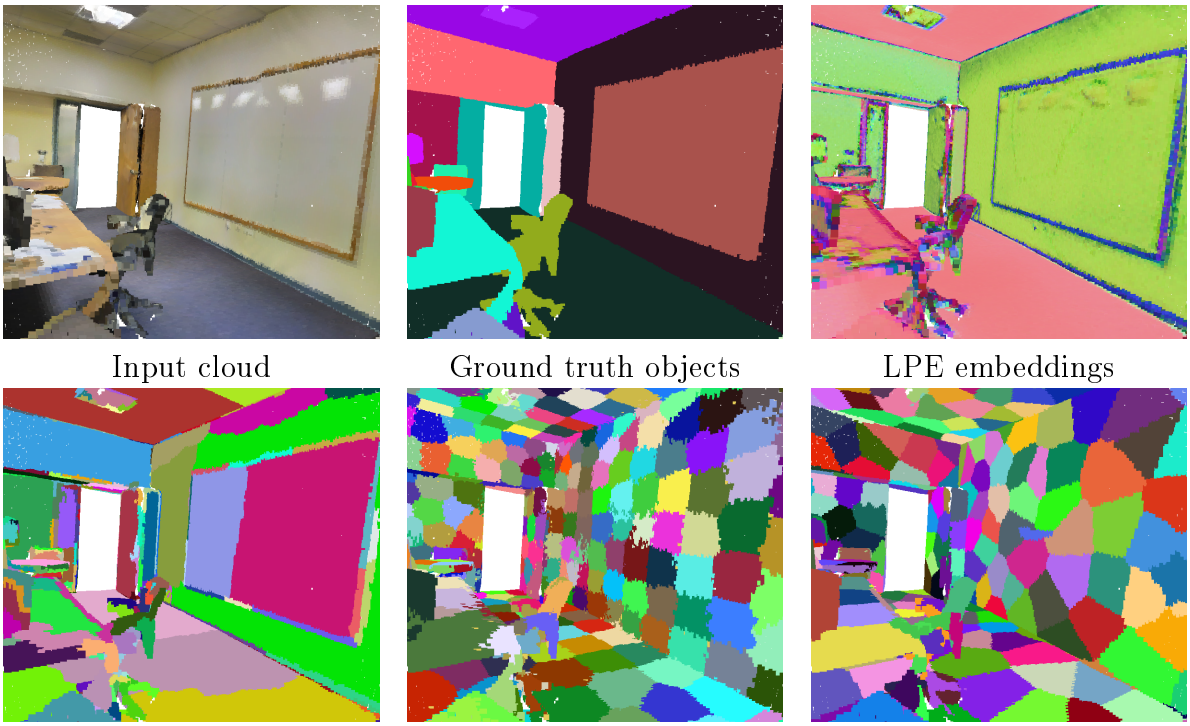


(a) OOA for S3DIS (b) BR for S3DIS (c) BP for S3DIS

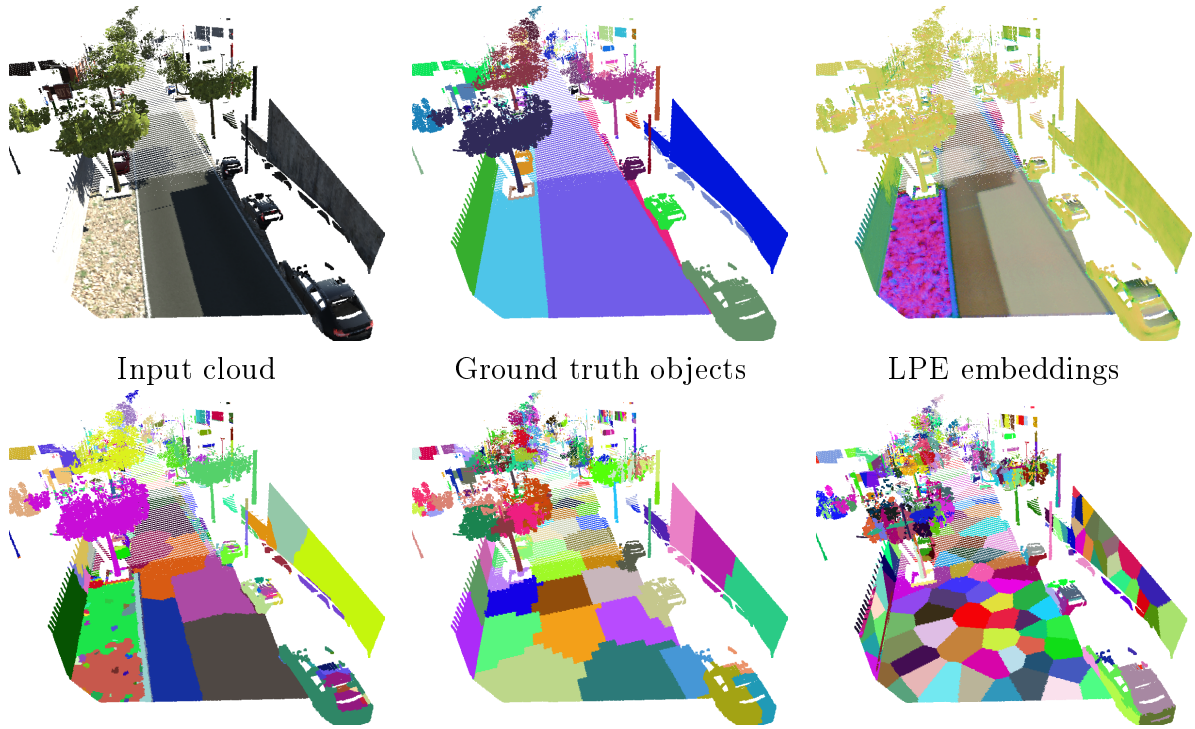
(d) OOA for vKITTI (e) BR for vKITTI (f) BP for vKITTI

Figure 4.29: *Performance of the different algorithms on the* 6*-fold S3DIS dataset (first row) and the* 6*-fold vKITTI3D dataset (second row).*

In Figure 4.29, we report the performance of our algorithm according to three segmentation metrics: OOA, BR, and BP. We observe that for the large scale S3DIS dataset (600 Mpoints), supervised methods provide considerably better results. In particular, our method **SSP** obtains better accuracy with 300 segments than the state-of-the-art method of **Lin *et al.*** with 1500 segments. The advantages for border recall and precision are even more significant. For the smaller vKITTI3D dataset (15 Mpoints), **Lin *et al.*** obtain better results than all supervised methods except our approach.



(a) S3DIS scene with 58 objects. Superpoint count : SSP 442, VCCS 436, Lin 423.



(b) vKITTI scene with 233 objects. Superpoint count: SSP 420, VCCS 422, Lin 425.

Figure 4.30: *Illustration of the oversegmentation results using our framework along with the state-of-the-art methods.*

Method	# parameters	OOA	BR	BP
Best	13,816	96.2	73.3	22.1
Prop-weights	13,816	-2.6	-12.2	+10.4
SEAL-weights	13,816	-1.3	-11.3	+3.8
2-Layers	14,688	-0.1	-0.7	-0.3
2-Residuals	14,688	+0.0	-0.2	-0.7

Table 4.3: *Impact of some of our design choice on S3DIS.* ***Best*** *is the* ***SSP*** *method with cross-partition weights.*

In Figure 4.30, we show the oversegmentation results of our method and the competing algorithms on vKITTI3D and S3DIS datasets.

We observe that our supervised partition framework produces superpoints of adaptive sizes which closely follow hard-to-segment objects such as white boards or sidewalks. We also notice that the embeddings learn to ignore certain form of intra-object geometric and radiometric variability. In particular, the lamp reflections on the white boards are almost completely ignored by the embeddings. Even more interestingly, the embeddings of trees are homogeneous despite the significant geometric variability between leafs and trunks. As a consequence, the trees are segmented into one component while the other methods produces many dubious superpoints.

4.3.3.4 Ablation study

To empirically validate our design choices for different steps of our supervized oversegmentation method, we have conducted an ablation study. we present the increase/decrease of the 3 performance metrics at 500 superpoints (linearly interpolated) of alternative methods compared to ours, on the first cross-validation fold of the S3DIS dataset. In particular we introduce **Prop-weight**, an alternative version in which the cross-partition weighting is replaced by a simple inversely-proportional weighting of the inter/intra edges. Predictably, this method gives lesser results as the edges are not weighted according to their influence in the partition. However, since the weights of the intra-edge are proportionally higher, the border precision is improved. We implemented the weights of the segmentation-aware affinity loss of [173] as well for method **SEAL-weights**, with comparable results to the **Prop-weight**. In **+TV-TV**, we replace our choice of function ϕ and ψ in the loss by respectively $|\cdot|$ and $-|\cdot|$, so that our loss is closer to the pairwise affinity loss used by [192] (but still structured by the graph). However, this approach wouldn't give meaningful partition as the intra-edge term conflicts with the constraint that the embeddings are constrained on the sphere. Removing this restriction leads the collapse of the embeddings around 0. We also tried to stack the LPE in layers, using or not a residual structure comparable to the one used in [226] to increase their receptive fields. The best results were achieved with two layers: **2-Layers** and **2-Residuals**. However, we observe that when compared with LPE of a similar number of parameters, the gains are insignificant if not null. We conclude that to embed points in order to detect borders, a small receptive field with a shallow architecture is sufficient.

4.3.3.5 Semantic segmentation resutls

- **Evaluation metrics:** In order to evaluate our semantic segmentation results, we use standard metrics from the literature. The most common metric is the *accuracy* defined as the ratio of the correctly classified elements overall elements of the dataset. When evaluating machine learning algorithms in general, four categories are usually considered. True positives (TP) defined as the set of elements that are correctly predicted to belong to a given class whereas true negatives (TN) are the set of elements that are correctly identified not belonging to a given class. Inversely, false positives (FP) are the number of negatives labeled as positives while false negatives are the number of positive examples labeled as negative. Accuracy is defined w.r.t. to these categories:

$$Accuracy = \frac{TP}{TP + FN} \tag{4.30}$$

  To circumvent the class imbalance issue which is prevalent in most of the real world datasets, we report class-wise scores in addition to the mean over all classes. Alternatively, intersection-over-union (IoU) is common metric in this task as well. IoU also referred to as the Jaccard index is a ratio between the number of elements shared between the ground truth and the prediction divided by the number of all elements across the target and the prediction. More simply the IOU can be defined as:

$$IoU = \frac{TP}{TP + FP + FN}. \tag{4.31}$$

- **Resutls:** In Table 4.4 and Table 4.5, we show how our point cloud oversegmentation framework can be successfully used by the superpoint-based semantic segmentation technique of [213][4] (**SPG**). We replace the unsupervized superpoint computation with our best-performing approach, **SSP**. We evaluate the resulting semantic segmentation using standard classification metrics: overall accuracy (OA), mean per-class accuracy (mAcc) and mean per-class intersection-over-union (mIoU). We observe a significant increase in the performance of **SPG**, beating concurrent methods on both datasets. In particular, we observe that our method allows for better retrieval of small objects which translates into much better per-class metrics, although the overall accuracy is not necessarily better than the latest state-of-the-art algorithms. The detailed per-class IoU are reported in the appendix.

We note that since the time of writing this thesis, several semantic segmentation algorithms surpassing our method have been developed as it is an extremely competitive field. We report only scores of methods that were developed before or at the same time as ours.

4.3.4 Discussion

- **Adjacency graph:** For both datasets, we find that setting the local neighborhood size to 20 was enough for embeddings to successfully detect objects' border. Combined with our lightweight structure, this results in a very low memory load overall. The adjacency graph G used for computing the graph-based partition requires more attention depending on the dataset. For the dense scans of S3DIS, the 5-nearest neighbors

[4] https://github.com/loicland/superpoint-graph

Method	OA	mAcc	mIoU
6-fold cross validation			
PointNet [14] in [223]	78.5	66.2	47.6
Engelmann *et al.* in [223]	81.1	66.4	49.7
PointNet++ [153]	81.0	67.1	54.5
Engelmann *et al.* in [192]	84.0	67.8	58.3
SPG [213]	85.5	73.0	62.1
PointCNN [227]	**88.1**	75.6	65.4
SSP + SPG (ours)	87.9	**78.3**	**68.4**
Fold 5			
PointNet [14] in [192]	-	49.0	41.1
Engelmann *et al.* in [192]	84.2	61.8	52.2
pointCNN [227]	85.9	63.9	57.3
SPG [213]	86.4	66.5	58.0
PCCN [228]	-	67.0	58.3
SSP + SPG (ours)	**87.9**	**68.2**	**61.7**

Table 4.4: *Performance of different methods for the semantic segmentation task on the S3DIS dataset. The top table is for the 6-fold cross validation, the bottom table on the fifth fold only.*

Method	OA	mAcc	mIoU
PointNet [14]	79.7	47.0	34.4
Engelmann *et al.* in [192]	79.7	57.6	35.6
Engelmann *et al.* in [223]	80.6	49.7	36.2
3P-RNN [229]	**87.8**	54.1	41.6
SSP + SPG (ours)	84.3	**67.3**	**52.0**

Table 4.5: *Performance of different methods for the semantic segmentation task on the vKITTI dataset with 6-fold cross validation.*

adjacency structure was enough to capture the connectivity of the input clouds. For the sparse scans of vKITTI, we added Delaunay edges [230] (pruned at 50 cm) such that parallel scan lines would be connected. The improvement is less significant on vKITTI, which could be due to the difficulty of constructing an adjacency graph on such a sparse acquisition. The performance is degraded further without color information, as some transitions are not predictable purely from the geometry.

- **Network configuration:** For the LPE and the PointNet structure in the spatial transform, we find that shallow and wide architectures work better than deeper networks. We emphasize that the approximated solution $f^\star$ of the optimization problem (4.5) takes values in $\mathbb{R}^{C\times m}$. However, the learned embeddings e_i are constrained in the m-unit sphere $\mathbb{S}_m$. In practice, this is a shortcoming of our approach as it could lead to sub-optimal approximate solutions. In terms of computational speed, the embeddings can be computed very efficiently in parallel on a GPU with over 3 million embeddings per second on a 1080Ti GPU. The bottleneck remains solving the graph partition problem in (4.5), which can process around $100,000$ points per second.

- **The transition edge weight M_0:** The graph-structured contrastive loss presented in Section 4.2.3 requires setting a weight M_0 determining the influence of inter-edges with respect to intra-edges. Since most edges of G are intra-edges, in practice, we define M_0 such that $M_0 = \mu c$ with $c = \mid E \mid / \mid V \mid$ the average connectivity of G. Note that c can be determined directly from the construction of the adjacency graph (it is equal to k in a k-nearest neighbor graph for example). A value of $\mu = 1$ means that the total influence in ℓ of inter-edges and intra-edges are identical. Since we are interested in oversegmentation, we set μ to 5 in all our experiments, but note that the network is not very sensitive to this parameter, as demonstrated experimentally: a value of $\mu = 3$ gives a relative performance of $(-0.2, -0.6, +1.5)$ while a value of 8 gives $(+0.1, -0.5, +1.4)$.

4.4 Applications: 3D semantic map as textured mesh

In this section, we apply our over-segmentation method to 3D textured meshes of urban scenes. We use the dataset introduced in Chapter 1 as there is no dataset that we are aware of which contains both registered geo-localized images and their corresponding 3D textured meshes of outdoor urban scenes along with their respective semantic ground truth annotations. Semantic segmentation of the textured mesh, however, is left as a future work due to time constraints. We start by presenting the embedding function ξ of the 3D textured mesh, a lightweight network inspired by the work introduced in MeshCNN [16] and U-Net [211].

4.4.1 3D textured mesh embedding

Let us consider a 3D triangular mesh $M = \{V, E, T\}$ defined by its sets of:

$$\begin{cases} \text{Vertices } V = \{v_i \mid v_i \in \mathbb{R}^3, 1 \leq i \leq n\}\,, \;\; V \subset \mathbb{R}^{3\times n} \\ \text{Edges } E = \{e_j = \{u, v\} \mid u, v \in V, 1 \leq j \leq p\}\,, \;\; E \subset V^{2\times p}, \\ \text{Triangles } T = \{t_k = \{u, v, w\} \mid \{u, v\}, \{u, w\}, \{v, w\} \in E, 1 \leq k \leq m\}\,. \end{cases}$$

where v_i is the spatial position (x, y, z) of the vertex i in the 3D space. We note that additional radiometric features can be also available such as RGB color or LiDAR reflectance attached either to the vertices V or triangles T. In the following method, we rely on 2D images for extracting radiometric features rather than the discrete photometric information attached to vertices or mesh faces.

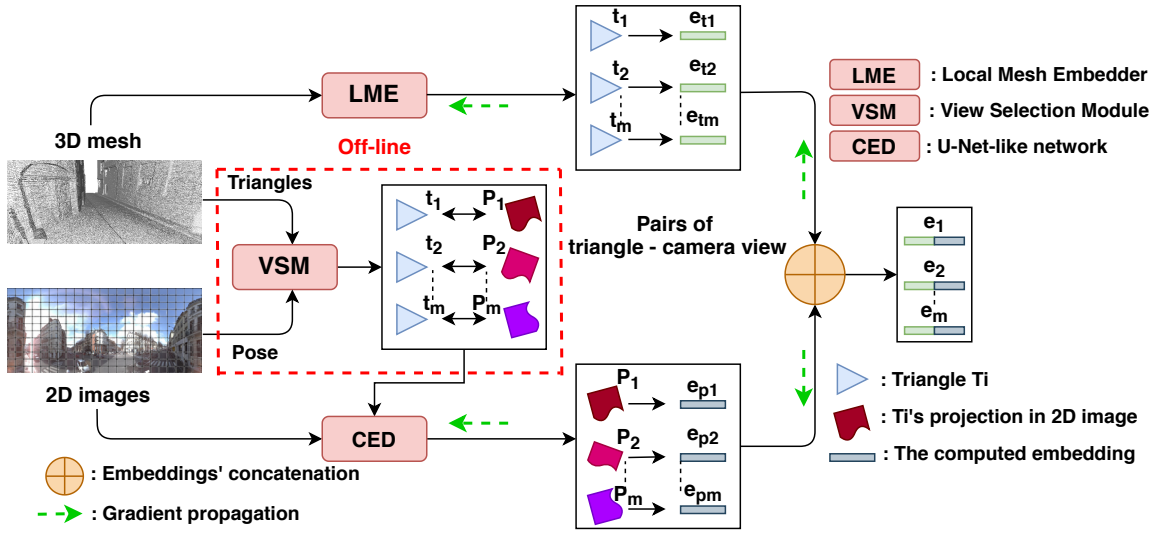


Figure 4.31: *The architecture of our Local Textured Mesh Embedding network (LTME)*

In Figure 4.31, we show the global network architecture of the function ξ used to embed 3D textured meshes locally called the *Local Textured Mesh Embedding network* (LTME). As input, our network takes a surface represented as a 3D triangular mesh M along with a set of registered 2D images C and their corresponding poses. First, we start by computing for each triangle $t \in T$ in the 3D mesh its projection fragment in the corresponding camera view $c_t \in C$. This step is performed off-line using a *view selection module* (VSM) as a pre-processing before training. Once the pairs of triangles and their corresponding camera views are determined, we compute an embedding function for each acquisition modality separately using respectively a *local mesh embedder* (LME) for the 3D mesh and a *convolutional encoder-decoder* (CED) which consists in a U-Net-like network [211] for images. Akin to the LPE architecture presented in Section 4.3.1, our goal is to associate each triangle $t \in T$ in the 3D mesh M and each pixel in the 2D corresponding image c_t with a low dimensional embedding. Therefore, the final set of the local textured mesh embeddings will be a concatenation of the mesh faces' embeddings and the pixels' embeddings corresponding to the faces' projection in each camera view. As discussed in Section 4.2.1, normalizing the embeddings to the unit sphere prevents collapse during training. Consequently, both mesh triangles' embeddings e_{lme} and image pixels' embeddings e_{ced} are constrained respectively to the d_{lme} and d_{ced} dimensional unit-spheres $\mathbb{S}_{d_{\text{lme}}}$ and $\mathbb{S}_{d_{\text{ced}}}$.

At this stage, the 3D mesh is structured by the dual graph $G_d(V_d, E_d)$ such that the nodes V_d of G_d are the set of mesh faces $t \in T$ while the edges E_d are the adjacency relations in the mesh M. The embedding function $\xi : V_d \mapsto \mathbb{S}_u \times \mathbb{S}_v$ representing our LTME is a mapping between the nodes V_d of G_d and the local computed embeddings. As illustrated in Figure 4.31, the proposed architecture is mainly composed of 3 principle modules. In the following we detail each one of them:

- **View Selection Module (VSM):** The objective of this module is to select for each triangle $t \in T$ ($|T| = m$ the number of triangles in the mesh), the optimal camera view $c \in C$ ($|C| = k$ the number of camera views) w.r.t. visibility constraints. For each triangle $t \in T$, the best camera view c_t is the one with the largest projection area of triangle t into camera c *i.e.* $c_t = \arg\max_c \{A[P_t(c)]\}$ where $A[P_t(c)]$ denotes the projection area of triangle t into the camera view c. The larger this area is, the better t fits into c. This criterion counts also for proximity as a larger projection area implicitly means that the selected view is spatially closer to the underlying face. Other geometric criteria for optimal view selection exist such as the angle between the face normal and the camera direction, albeit we abstain from using this feature as it is scale-invariant [156]. Similar to texture mapping procedure described in Chapter 3, we compute a labeling $\mathcal{L} = \{l_1, .., l_m\} \in \{1, .., k\}^m$ which associates to each face $t \in T$ the best camera view c_t. We cast our view selection as a multi-label optimization problem solved using α-expansion [175]. The optimal labeling $l^\star$ is defined as:

$$l^\star = \arg\min_{l \in \mathcal{L}} (-\sum_{t \in T} A[P_t(l_t)] + \gamma \sum_{e_{(s,t)} \in E_d} [l_s \neq l_t]) \tag{4.32}$$

As described in (4.32), this energy is a weighted sum of a data term and a regularization term. The first term evaluates the visibility quality of a triangle t having a label l_t corresponding to a camera view c as explained above. The second is a pairwise term favoring the selection of spatially regular views for adjacent triangles which are more likely to share the same label (*i.e.* camera view). In details, we penalize the set of edges E_d common to adjacent triangles $s, t \in T$ which were assigned to different labels $l_s \neq l_t$. This smoothness term consists of a Potts model where [.] are the Iverson brackets ($[x = y]$ is equal to 1 if $x = y$ and 0 otherwise). Finally γ is the weight balancing smoothness and the data fidelity.

Once $l^\star$ in equation (4.32) is computed, we are able to determine for each face in 3D space the set of 2D pixels in the corresponding camera view. This mapping from faces of the 3D mesh to pixels of the corresponding 2D views is needed once the image embeddings are computed using a convolutional encoder-decoder network.

- **Convolutional Encoder-Decoder (CED):** In order to embed pixels of a 2D image, we propose to use a convolutional encoder-decoder inspired by state-of-the-art deep convolutional network U-Net [211]. In Figure 4.32 we show the architecture of our network. CED is composed of two parts; a contracting path called encoder and an up-sampling path − the decoder. The encoder is a stack of convolutional layers interleaved with batch norm layers (BN), non-linearity layers (ReLU), in addition to pooling layers. It takes as input an image I of size $H \times W \times d$ with H, W and d are respectively the height, width and depth (*i.e.* number of channels) of I. For RGB images the depth d is equal to the dimension of the RGB space which means $d = 3$.

  The decoder part operates on the reduced feature maps resulting from the pooling layers of the encoder. It performs an up-sampling operation to restore the original size of the feature map using the indexes of pixels from the corresponding pooling layers in the encoder. This way, the unpooling layer can precisely localize the lost information in the pooling stage in the encoder. After each unpooling step a previously computed feature

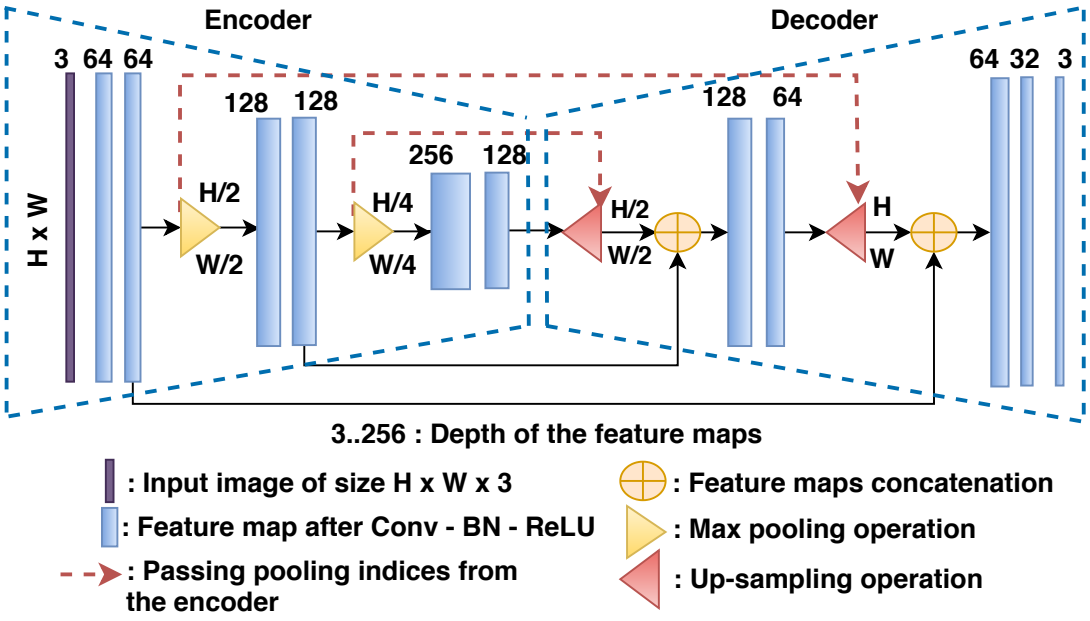


Figure 4.32: *The architecture of our Convolutional Encoder-Decoder network (CED) for camera views' embedding*

map from the encoder is concatenated with the current one in the decoder in order to combine information for more precise predictions. Lastly, a 1×1 convolutional layer allows to extract a feature map having a depth d which corresponds to the dimension of the computed embeddings for each pixel. CED embeddings are subsequently normalized within the d_{ced}-dimensional unit-sphere. In practice, we found that three dimensional pixel embeddings $e_{\text{ced}} \in \mathbb{S}_3$ are convenient for our application.

- **Local Mesh Embedder (LME):** In this module, we seek to locally embed faces of a triangular 3D mesh in a similar way as for point clouds. The proposed network is based on the work of Hanocka *et al.* called MeshCNN [16] in which CNN building blocks (*i.e.* convolution and pooling) on 2D regular images are translated to irregular meshes. LME operates on the edges E of the mesh instead of triangles T to provide a non-uniform geodesic neighborhood which is compatible with the inherent irregularity of the mesh. As explained in Section 4.1.4, we start by computing hand-crafted geometric features for each edge $e_i \in E$ where $|E| = p$. Recall that at this level the features are computed on the primal edges E of the mesh $M = \{V, E, T\}$. The dual graph $G_d(V_d, E_d)$ built on top of M is used subsequently for the supervised graph partitioning step. Figure 4.33 shows the architecture of our Local Mesh Embedding network (LME).

  For an edge $e_i \in E$, the input hand-crafted edge features are represented as a 5-dimensional vectors denoted by $hf_i \in \mathbb{R}^5$ and composed of the dihedral angle between adjacent faces, the inner two angles and the edge ratios between the length of the edge and its perpendicular lines in each adjacent face plan. The computed edge features (of initial dimension $f_{in} = 5$) are fed to a stack of edge convolution layers (EC) (4.33) (producing f_{out}-dimensional learned features) interleaved with batch normalization (BN) and non-linearity units(ReLU). In details, the edge convolution operation defined in equation (4.4) consists in the multiplication of a multi-channel tensor $\mathcal{T}(b \times f \times e \times n)$ with a kernel of size (5×1) where e is the maximum number of edges per mesh, n is the

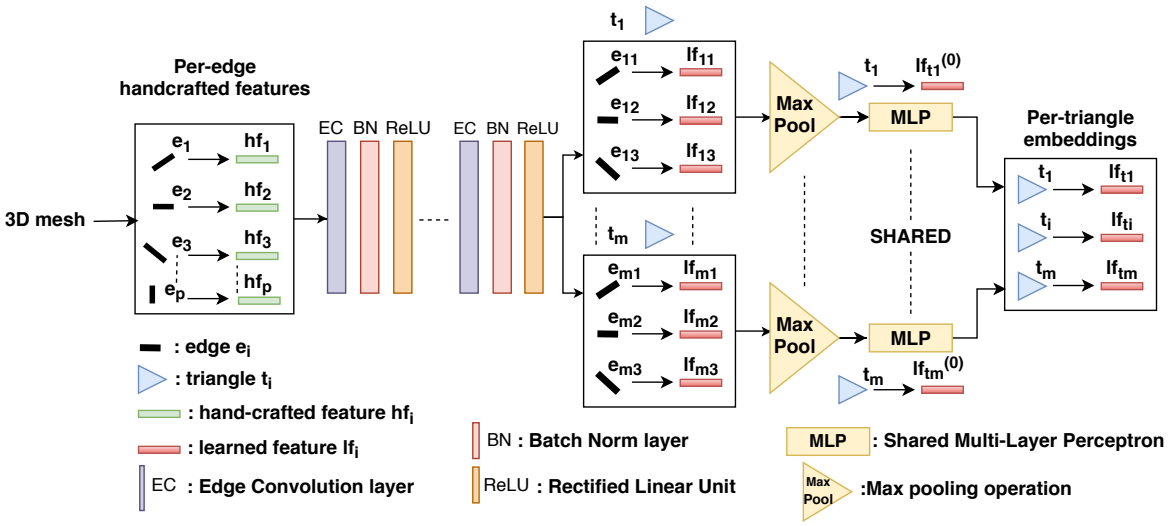


Figure 4.33: *The architecture of our Local Mesh Embedding network (LME)*

number of neighbors (by default $n = 4$ which correspond to the 1-ring neighbors as explained in Section 4.1.4), f is the number of features (initially $f = 5$) and b is the batch size. This results in deep learned features per-edge denoted lf_i which are subsequently organized with regards to the mesh triangles (*i.e.* we associate to each face $t_i \in T$ of the mesh its three learned edges' features $lf_{i1}, lf_{i2}, lf_{i3}$). Afterwards, the embedding of each face lf_{t_i} is obtained by the aggregation of its three edge deep embeddings using a max pooling operation as in (4.34) (the max operation in (4.34) returns the element-wise maximum of vectors $lf_{i1}, lf_{i2}, lf_{i3}$). Finally the computed embeddings are passed to a shared multi-layer perceptron (MLP) which allows to further learn a spatial-encoding of faces while reducing the embeddings' dimension to the desired size. Much like CED embeddings, LME embeddings are normalized within the d_{lme}-dimensional unit-sphere (4.36). In practice, LME embeddings are of dimension 3 ($e_{\text{lme}} \in \mathbb{S}_3$).

$$EC : \mathbb{R}^{f_{in} \times p} \mapsto \mathbb{R}^{f_{\text{out}} \times p} \quad (4.33) \qquad \mathrm{L}_2(\cdot) = \cdot / \|\cdot\| \quad (4.35)$$

$$lf_{t_i} = \max(lf_{i1}, lf_{i2}, lf_{i3}) \quad (4.34) \qquad e_{\text{lme}}(t_i) = \mathrm{L}_2\left(\text{MLP}\left(lf_{t_i}\right)\right) \quad (4.36)$$

Using the optimal labeling $l^\star$ (4.32) computed by the view selection module (VSM), we can determine for each face embedding $e_{\text{lme}}(t)$, in the 3D mesh its corresponding projection P_t in the feature map $e_{\text{ced}}(c_t)$ computed by CED. To determine the set of pixels that overlap the triangle's projection in the feature map, first, for efficiency reasons, we loop over the feature map pixels that are located inside the bounding box of the projected triangle. Then we perform an inside-outside test to find out the set of pixels inside the triangle's projection using the *edge function* Pineda [231].

Following the notation from Figure 4.34, for the edge defined by vertices A and B and a pixel P, the edge function is defined as the magnitude of the cross product of vectors $\overrightarrow{AP}$ and $\overrightarrow{AB}$: $E_{AB}(P) = ||\overrightarrow{AP} \times \overrightarrow{AB}||$. Following a clock-wise convention, if $E_{AB}(P) \geq 0$, $E_{BC}(P) \geq 0$ and $E_{CA}(P) \geq 0$, then we consider that the pixel P is inside the triangle's projection on the feature map, otherwise, it is outside (*e.g.* Pixel R is outside the triangle since $E_{AB}(R) < 0$).

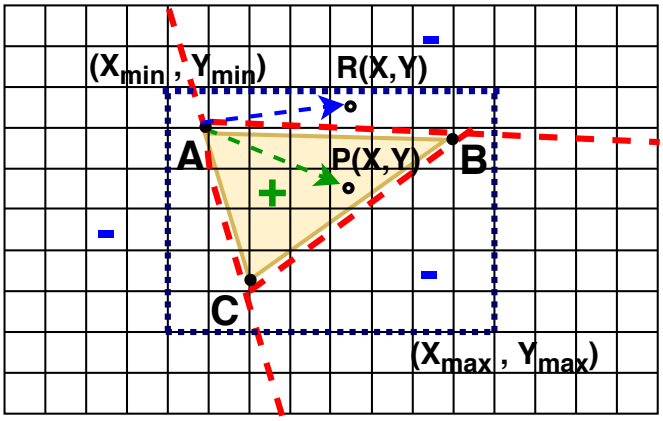


Figure 4.34: *Illustration of an inside-outside test using the edge function [231]*

Assuming that the projection area of t into the feature map $e_{\text{ced}}(c_t)$ is composed of the set of pixel embeddings $e_i^{P_t}(c_t) \subset e_{\text{ced}}(c_t)$, the local embedding corresponding to the fragment P is obtained by aggregating all the pixel embeddings $e_i^{P_t}(c_t)$ using a max pool operation (4.37) (The max operation in (4.37) gives the element-wise maximum of vectors $e_i^{P_t}(c_t)$). At last, the final embedding computed by our LTME network e_{ltme} is the concatenation of mesh faces embeddings $e_{\text{lme}}(t)$ with $t \in T$ and the corresponding projection fragments embeddings $e_{\text{ced}}^{P_t}(c_t)$ (4.38).

$$e_{\text{ced}}^{P_t}(c_t) = \max_{i \in P_t}(e_i^{P_t}(c_t)) \tag{4.37}$$

$$e_{\text{ltme}} = e_{\text{lme}}(t) \oplus e_{ced}^{P_t}(c_t) \; \forall \, t \in T \tag{4.38}$$

It should be emphasized that it might happen that a few number of faces can not be seen in any of the camera views. For those triangles we heuristically duplicate LME embeddings to ensure an homogeneous size of final embeddings. While different strategies could be used such as propagating embeddings from adjacent triangles or zero padding, experimentally we found that it has minor effects on over-segmentation results as the number of unseen faces represent no more than 3% of all the dataset.

4.4.2 Numerical experiments

In this section, we show the over-segmentation results of our method on the pLaTINUM dataset acquired and annotated during this project. Since ground truth annotations of the textured mesh is available for only semantic segmentation, we compute objects annotations as the set of connected components of semantic labels of faces in a similar way as for vKITTI dataset [223]. For these experiments, we use the LTME network described above as our embedding function ξ.

4.4.2.1 Implementation details

- **Convolutional encoder-decoder (CED):** We note that CED is first pre-trained on an equivalent outdoor urban dataset vKITTI [57] with similar number of classes as

our dataset. Due to the relatively small number of annotated images in our dataset, we are not able to train the CED network from scratch using exclusively our own data. Consequently, we fine-tune the model using our set of 2D perspective front-facing images. We resize the original images of size $2048 \times 2048 \times 3$ to $256 \times 256 \times 3$. Overall, CED encoder is composed of 5 convolutional layers with an increasing number of filters each of which is followed by a $(2, 2)$ max pooling, batch norm and ReLU layers. The convolution kernel is of size $(3, 3)$ for all the convolution layers in CED. The decoder alternates between $(2, 2)$ upsampling and convolutional layers in addition to a concatenation of feature maps of the encoder. The detailed number of convolution filters used in each convolutional layer is shown in Table 4.6.

- **Local mesh embedding network (LME):** Since our LME network inspired by meshCNN [16] is a translation of Convolutional Neural Networks (CNNs) on 2D images to 3D meshes, the commonly used pre-processing steps (*e.g.* resizing the input data, centering around the mean, etc.) were also replicated for 3D meshes. We simplify the input meshes in the training and test sets such that they have roughly a similar number of edges. Reducing the resolution of the input meshes is performed using a geometric decimation of an order of magnitude (*i.e.* an input mesh having 10^6 faces after decimation it will have 10^5 faces). As discussed above, the manifoldness property of the mesh is a key ingredient for the success of the convolution operation on edges since each edge has to be incident to at most two faces *i.e.* the number of edge neighbors is at most equal to 4. To this end, we transform any invalid, non-manifold mesh instance in the dataset, to a valid one by first locating all non-manifold edges and then removing all their incident faces before training. It should be noted that a 3D mesh, despite being manifold, may contain boundary edges (*i.e.* edges which are incident to only one triangle). Since boundary edges have only two neighboring edges, we circumvent this lack by zero padding in the convolution operation as proposed in [16]. Furthermore, We center the input edge-wise hand-crafted features by subtracting the mean edge feature of all the dataset and dividing by its standard deviation.

  The proposed LME network comprises 4 edge convolutional layers (EC) with an increasing number of filters, each of which is followed by a batch normalization layer (BN) and a non linearity (ReLU). The output of edge convolution is first max-pooled and then consumed by a multi-layer perceptron MLP having two hidden layers of size 128 and 64. The detailed configuration of LME is exposed in Table 4.6.

  To boost the robustness of the network, several data augmentation techniques can be used. We follow the same protocol advocated by Hanocka *et al.* [16]. We first note that the 5-dimensional edge-wise hand-crafted features (angles and edge ratios) used as input to the edge convolution (4.4) are designed to be invariant to the common geometric transformations *i.e.* rotation, translation and uniform scaling (the same scaling factor applied to (x, y, z) coordinates). Therefore, applying those transformations to the vertices of the mesh is most likely useless. To generate new features we shift the location of 20% of the mesh vertices by applying a random scaling (sampled from the normal distribution of mean $\mu = 0$ and standard deviation $\sigma = 1$) on its x, y, z coordinates separately.

- **Graph structured contrastive loss:** For the supervised graph partitioning step, the training is performed using our best configuration achieving the highest scores reported for point cloud over-segmentation. This means that the graph structured contrastive

loss defined in equation (4.8) is weighted using the cross partition strategy (4.12). However, unlike point clouds, we propose a slightly different weighting compatible with the mesh topology.

In the case of 3D meshes, the size and shape of triangles vary substantially w.r.t. to the geometry of the reconstructed objects. For instance, large planar objects are represented by few triangles having large areas, while non-flat objects are represented by a higher number of triangles having smaller areas. Thus, the edge weights $\mu \in \mathbb{R}_{+}^{E_{\text{trans}}(\mathcal{P})}$ in equation (4.8) should adapt to the varying topology by incorporating indicative geometric indices such as the area of triangles and the length of edges rather than simply their numbers. Formally, following the same notation as the cross partition strategy for point clouds (4.12), the adapted weights can be defined as :

$$\begin{aligned} M_{U,V}^{(e)} &= M_0 \min\left(A(U), A(V)\right) \text{ for } (U,V) \in \mathcal{E}^{(e)} \\ \mu_{u,v}^{(e)} &= \frac{M_{U,V}^{(e)}}{W_{U\times V \cap E_{\text{trans}}}} \text{ for } (u,v) \in U \times V \cap E_{\text{trans}}. \end{aligned} \tag{4.39}$$

where $A(U) = \sum_{u \in U} area(u)$ denotes the area of the segment U and $W_{U\times V\cap E_{\text{trans}}} = \sum_{e \in U\times V \cap E_{\text{trans}}} length(e)$ is the sum of transition edges' lengths. Recall that this weighting strategy is based on the cross-segmentation graph $\mathcal{G}^{(e)} = (\mathcal{C}^{(e)}, \mathcal{E}^{(e)})$ defined in equation (4.11) for which $(U,V) \in \mathcal{C}^{(e)}$ [2].

In table 4.6, we show the configuration of our LTME network. The parameters which are not mentioned in this table and required by the network, use the default values as for point cloud over-segmentation.

parameter	shorthand	Module	Details
LME # parameters	-	LME	258819
CED # parameters	-	CED	1314435
LME configuration	f	LME	[5,32,64,128,256],[256,128,64,3]
CED encoder configuration	-	CED	[64,64,128,128,256,128]
CED decoder configuration	-	CED	[128,64,64,32,3]
Batch size	b	LTME	4
LME embeddings dimension	d_{lme}	LME	3
CED embeddings dimension	v	CED	3
Regularization weight	γ	VSM	0.85
Adjacency graph	G_d	-	Dual graph
epochs	-	-	150
decay event	-	-	40,80,120

Table 4.6: *Configuration of the LTME network for pLaTINUM dataset.*

4.4.2.2 Evaluation metrics

In the literature, specific evaluation metrics have been proposed to assess qualitatively and quantitatively 3D mesh over-segmentation. In [171], Simari *et al.* have tailored two metrics from 2D superpixels to better represent the underlying topology of a 3D mesh.

Namely, for a segmentation $\mathcal{S}$ and a ground truth partition $\mathcal{P}$ the under-segmentation error (UE) and compactness (C) are defined respectively as:

$$UE(\mathcal{S},\mathcal{P}) = \frac{\sum_{P_i\in\mathcal{P}}\sum_{S_j\in\mathcal{S}} A(S_j \setminus P_i)}{\sum_i A(P_i)} \tag{4.40}$$

$$C(\mathcal{S}) = \sum_{S\in\mathcal{S}} \frac{P(S)}{\mid \mathcal{S} \mid \sqrt{A(S)}} \tag{4.41}$$

Where $A(.)$ and $P(.)$ denotes respectively the area and perimeter of a segment. Simari *et al.* in [171] argue that unlike pixels in 2D images, faces in a 3D mesh usually have non-uniform size or shape. Therefore, the areas and perimeters of faces are better suited for evaluation than merely counting their proportions. As far as we are concerned, we agree with this assessment, albeit with a different formulation. We believe that the proposed metrics for point cloud over-segmentation can also be used for evaluating mesh over-segmentation. As for point clouds, we argue that a transition occurs *between* triangles and not *at* them. We define $E^{\text{pred}}_{\text{trans}}$ the set of predicted transitions, *i.e.* the subset of edges of E_d that connect two triangles of T (or two nodes in V_d) belonging to two different superfacets, while $E^{(1)}_{\text{trans}}$ [5] is the set of ground truth transition edges. By taking a closer look to the proposed metrics in Section 4.3.3, we can notice that both BR and BP (4.27) are oblivious to the mesh topology as their unit of measure is defined w.r.t. to the number of edges of the mesh dual graph $G_d(V_d, E_d)$ being a transition or non-transition edges. Instead of computing the precision and recall in terms of number of edges, we believe that computing these two metrics w.r.t. to the edges lengths is more suitable in the case of non-uniform triangulated meshes. Formally BR and BP can be defined as follows:

$$BR = \frac{W_{E^{\text{pred}}_{\text{trans}}\cap E^{(1)}_{\text{trans}}}}{W_{E_{\text{trans}}}},\ BP = \frac{W_{E^{\text{pred}}_{\text{trans}}\cap E^{(1)}_{\text{trans}}}}{W_{E^{\text{pred}}_{\text{trans}}}},\ \text{where } W_X = \sum_{e\in X} length(e) \tag{4.42}$$

However, as shown in Chapter 3, the reconstructed surface is designed to have an almost uniform triangulation by filtering out elongated faces in the mesh. When computing the coefficient of variation of the edges lengths E_d defined as $CV = \frac{\sigma}{\mu}$ where σ is the standard deviation and μ is the edges lengths mean, we found that $CV = 0.2374$. This low level dispersion around the mean edge length confirms our claim. For those reasons, we assume that computing BR and BP in terms of number of edges and not their length is sufficient to evaluate the performance of over-segmentation methods on our dataset at least. Therefore, we stick to the current formulation to assess 3D mesh over-segmentation. The oracle overall accuracy OOA (4.29), on the other hand, bypasses the mesh structure as it considers the semantic labels of the graph nodes which are the set of mesh faces.

4.4.2.3 Competing algorithms

In order to evaluate the performance of our supervised over-segmentation method, we compare it against superfacet segmentation methods from the literature. We denote our

[5] (1): The superscript stands for the number of edges tolerated to consider the current predicted transition as correctly retrieved similar to the point cloud case in Section 4.3.3

method as **LSF** which stands for Learned Superfacets. In the following, we introduce three variants of **LSF** which serve as an ablation experiment to study separately the influence of the LME and CED modules discussed in Section 4.4.1. We have also implemented a graph-based mesh over-segmentation (**HSF-Graph** which stands for hand-crafted superfacets) relying on hand-crafted features. Finally, we use the method of Simari *et al.* [171] which is a cluster-based method (**HSF-Cluster**). This way, we have a similar set up as in the case of point clouds:

- **LSF:LME:** In this configuration the LTME architecture is comprised only of the LME network which operates on 3D meshes. We consider exclusively LME embeddings $e_{\text{lme}}(t)$. CED embeddings $e_{\text{ced}}^{P_t}(c_t)$ are ignored in this experiment.

- **LSF:LTME-L:** CED is trained on vkitti dataset [57] and fine-tuned on our dataset to perform semantic segmentation. During back-propagation, LTME weight updating in this configuration is exclusively restrained to the LME network while CED weights are frozen during training. At each graph partitioning iteration during training, we concatenate CED's pre-computed embeddings for the task of semantic segmentation with the learned LME embeddings $e_{\text{lme}}(t)$ for over-segmentation. The goal is to understand the impact of fusing embeddings learned previously on a different task (in this case semantic segmentation) with embeddings learned on the desired task (over-segmentation).

- **LSF:LTME-E:** This is the original proposed set up. In this configuration the LTME embeddings are computed as explained in equation (4.38) where we concatenate the LME embeddings $e_{\text{lme}}(t)$ with CED embeddings $e_{\text{ced}}^{P_t}(c_t)$. Both the weights of LME and CED are updated during training until the convergence of the loss.

- **HSF-Graph:** This is a graph-based over-segmentation method. We use the Generalized Minimal Partition Problem (GMPP) defined in Section 4.2.2 by the equation (4.5) to partition the 3D mesh represented by its dual graph as defined in Section 4.1.3. The nodes V_d of the graph G_d are the set of mesh faces $t \in T$ while the edges E_d are the adjacency relations in the mesh M. Each triangle is represented by a 4-dimensional feature vector characterizing its geometric adjacency inspired by Rouhani *et al.* [158]:

  Verticality: This feature is computed as a dot product between the vertical vector n_z and the normal of mesh faces t : $v = \mid n_z.n_t \mid$. The vertical objects such as building facades have low values while horizontal objects such as road and sidewalk have high values.

  Planarity: Computed for each face t_i with respect to its 1-ring neighbors [6] of size n, planarity is defined as $p = \min(\mid n_{t_i}.n_{t_j} \mid, \ldots, \mid n_{t_i}.n_{t_n} \mid)$. The value of p is high for planar facets and inversely low for non-planar facets.

  Elevation: This feature measures the height of each triangle with respect to the ground: $z = \frac{z_{t_i} - z_{min}}{z_{max} - z_{min}}$ where z_{max} (respectively z_{min}) corresponds to the maximal elevation (resp. the minimal) on the plane XY.

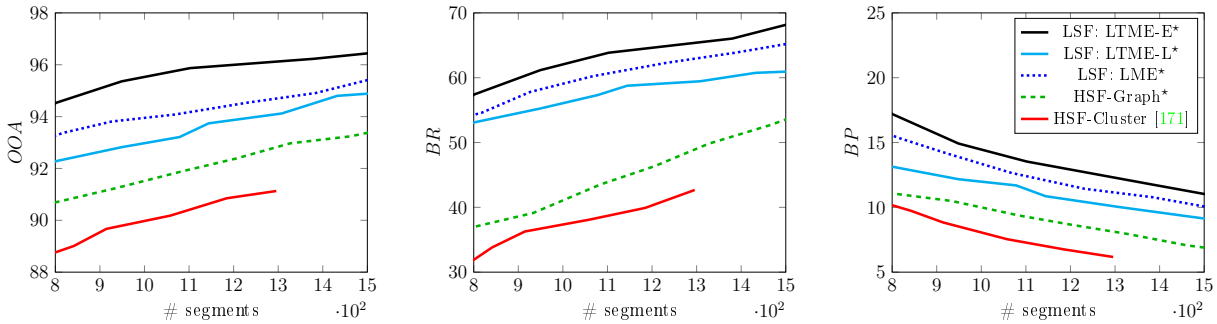
  Compactness: For each face $t_i \in T$, this feature is computed as a normalized ratio between its perimeter P_{t_i} and its area A_{t_i}. For a face t_i this ratio is defined

[6] 1-ring neighbors of a face t are the set of immediate neighboring faces to t that shares at least one of its vertices. Their number n varies depending on whether t is a border face in a non-closed manifold mesh or not.

as $r_i = \frac{P_{t_i}}{A_{t_i}}$. The compactness is defined as : $c = \frac{r_i - r_{min}}{r_{max} - r_{min}}$ where r_{max} (resp. r_{min}) are the maximal (resp. the minimal) measured compactness in the mesh. The larger the perimeter of t_i given its area, the more elongated t_i will be (less compact).

We note that other interesting hand-engineered features exist in the literature such as the average geodesic distance (AGD) [181], Shape Diameter Function (SDF) [180] among others. However since our experiments are conducted on outdoor urban scenes where the majority of objects' shapes are planar (façade, road, pavement, etc.) we only use features that are most likely boundary-indicative.

- **HSF-Cluster:** This is the clustering-based method of [171] discussed in Section 4.1.4.



(a) OOA for pLaTINUM (b) BR for pLaTINUM (c) BP for pLaTINUM

Figure 4.35: *Performance of the different over-segmentation algorithms on our dataset. We control the number of segments by varying the parameter λ in equation (4.5) between $[0.3, 0.7]$ as for point clouds. Methods tagged with a star ($\star$) were implemented by ourselves.*

In Figure 4.35 we show the performance of the described methods on our datasets for different number of superfacets in the test set. We observe that our best configuration **LSF:LTME-E** outperforms the competing methods by a significant margin. This was expected as supervised deep-learning methods have been shown to be more effective than hand-crafted based methods. Being a hand-crafted method, the cluster-based over-segmentation of [171] **HSF-Cluster** achieves lower results in terms of boundary precision and recall as well as the segmentation accuracy (ASA) than our proposed graph-based over-segmentation method **HSF-Graph**. Moreover for more than 10^3 superfacets, **HSF-Cluster** takes unreasonable time to compute the over-segmentation. This lesser performance typically arises from the sensitivity of cluster-based methods to their initialization. In addition, the reconstructed mesh from the LiDAR acquisition is characterized by a highly variable density such that the nearest objects to the scanner will have denser triangulation while the furthest exhibits a light triangulation not to mention the post processing consisting in hole closing which results most of the time in elongated triangles. This aspect is not taken into consideration by clustering-based methods as they are agnostic to adjacency and structure of the mesh. The graph-based approach **HSF-Graph** however, is able to produce compact superfacets that adhere faithfully to boundaries, to some extent, despite the limited descriptive capabilities of the chosen features for graph partitioning.

We observe that **LSF:LME** configuration performs better than **LSF:LTME-L** on all metrics. As described above, in the latter configuration, only the weights of the local mesh embedder LME are updated during training. Since, CED is already pre-trained for the task of semantic segmentation, we believe that the computed feature maps provide misleading information when fused with LME embeddings, yielding, hence, to a sort of confusion when partitioning the graph. In essence, our graph-structured contrastive loss, which is supervised by object instance annotations, is expected to produce embeddings that are homogeneous within objects and present high contrast at their borders. Meanwhile, the obtained feature maps from CED were designed for the task of semantic segmentation not instance segmentation. Due to the clutter effect in 2D images, two close parked cars viewed from behind for example, will be considered as a single semantic object which makes CED embeddings implicitly miss the border separating these two objects. **LSF:LTME-E** configuration mitigates this issue by updating the weights of CED during training w.r.t. to our graph-structured contrastive loss so that the learned embeddings in the last feature maps are able to grasp the notion of borders between objects.

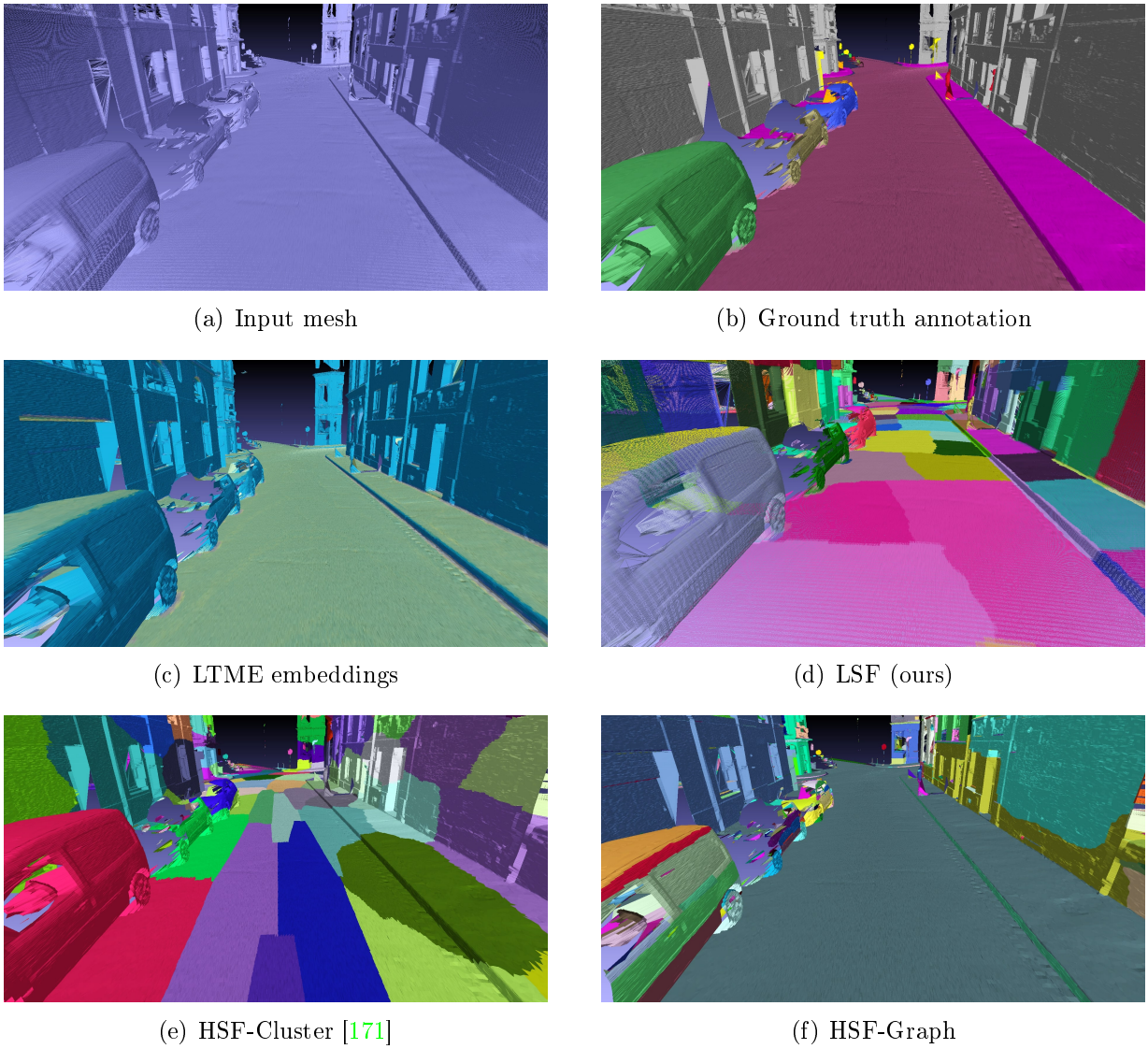
In Figure 4.36, we show illustrations of over-segmentation results of our best configuration **LSF:LTME-E** compared to evaluation methods **HSF-Graph** and **HSF-Cluster** [171] for a nearly equal number of superfacets. We observe that our method produces a better over-segmentation with compact and regular superfacets that adhere to objects' boundaries. **HSF-Cluster** present lower result than all the competing methods. We can clearly observe that the produced superfacets straddle objects having different semantic classes (building, sidewalk and road). **HSF-Graph** exhibits decent results. While planar objects such as road and sidewalk are predicted as a single large superfacet, this method is able to correctly partition the wall. It should be noticed that **HSF-Graph** produces large superfacets when features faithfully describe a simple geometry of facets in contrast to very small ones at regions where there is an ambiguity. This makes **HSF-Graph** illustration looks like it has the least number of superfacets compared to the competing methods which is not the case.

4.4.3 Discussion

As depicted in Figure 4.31, 2D images and 3D mesh are fed separately to our LTME network. The intuitive question that arises from this observation is why the network does not consume a raw textured mesh comprised of a 3D model as a triangular mesh along with its texture atlases and trained online. In practice, when implementing this approach we found that extracting pairs of triangles and their corresponding fragments in the adequate camera view is very compute-intensive. In details, we need to parse each of the texture atlases and fetch for each triangle its corresponding *texel*. Furthermore, as shown in Chapter 2, an additional color adjustment step is conducted to minimize visible seams in the final model. This will induce a bias to the original radiometric information making it a source of error when computing the embeddings of images. Most importantly, the underlying spatial adjacency of texture patches in a texture atlas is meaningless as they are basically organized that way for efficiency reasons with no regards to semantics. From Figure 4.37, we can see that the semantic adjacency in a texture atlas is not preserved.

Training CED on texture atlases will inhibit the network from leveraging the local context knowledge encoding high-level concepts related to semantic adjacency.

For this reason, we propose to go a step backward before packing the texture fragments



(a) Input mesh

(b) Ground truth annotation

(c) LTME embeddings

(d) LSF (ours)

(e) HSF-Cluster [171]

(f) HSF-Graph

Figure 4.36: *Over-segmentation illustration of a textured mesh in our pLaTINUM dataset. Superfacets count : LSF (727), HSF-Graph (713), HSF-Cluster [171] (700)*

into texture atlases and use the result of the view selection module as an off-line step to compute face-texel pairs. However, this comes with several drawbacks. For instance in a set up containing n camera views and k triangles, the search space is of dimension k^n making computation intractable. The prohibitive cost of computing pairs of faces-texels can be alleviated by using timestamps information of the camera view. As described in Chapter 2, both LiDAR and camera view are acquired strictly simultaneously. The provided timestamps allows to filter out images that do not lies within the interval of time during which a given chunk of mesh is acquired. While such filtering partially reduce computation time and memory load still it is not compatible with an online training.

Even though the view selection module has been shown to be efficient, we believe that the proposed energy formulation in (4.32) used to compute the pairs of faces and their corresponding fragments in 2D images can be substantially improved. First, we can see that (4.32) relies only on geometric information and do note leverage the available

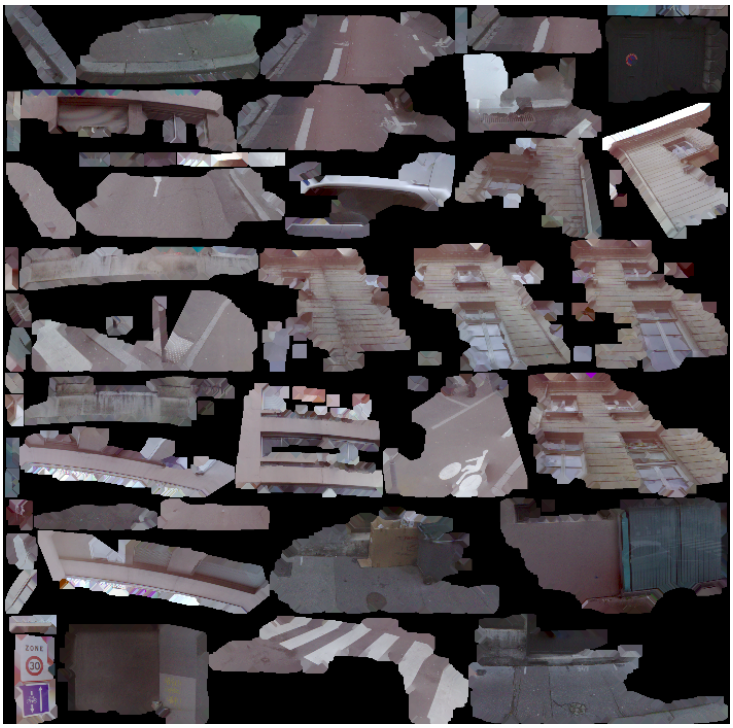


Figure 4.37: *Illustration of a set of texture patches packed into a texture atlas*

semantic ground truth annotations. At this stage, the goal of our VSM is to compute semantic faces-texels association rather than texture mapping. In the literature [156], this can be carried out by adopting a different energy formulation involving semantic labels. Moreover, [156] have challenged the assertion (advocated by the unary term in (4.32)) implying that a larger area view projection necessarily means a correct semantic association between the underlying face and corresponding texture fragment which is not always true.

We note that compared to point cloud over-segmentation, 3D mesh over-segmentation exhibits lesser results in terms of BR, BP (4.27) and OOA (4.29). We believe that this drop in scores is explained by two major factors. First as discussed earlier, LME operates on edges separately then a pooling operation aggregates edge features for each triangle. This means that the context knowledge information is taken into account only at the level of the 1-ring edges adjacency. Therefore, differently from LPE architecture where each point is embedded along with its k-nearest neighbors, LME embeds each triangle with respect to its immediate adjacent faces. Hanocka *et al.* [16] mitigates this problem by introducing a U-Net-like network as a convolutional encoder-decoder where pooling layers make the network insensitive to the change of features position in the feature map by reducing its size constantly. The pooling operation on meshes introduced in [16] is mainly an edge collapse-based decimation of the input mesh. According to [16], collapsing an edge during pooling must not result in non-manifold edges otherwise the subsequent convolution operations on edges will systematically fail as discussed in Section 4.1.4. However, in our experiments, we were not able to replicate this operation because of technical implementation problems in addition to the inherited challenging topological properties of our reconstructed meshes. As explained in Chapter 2, the reconstructed surface using our sensor-topology-based method depicts holes in the final model and is not guaranteed to

be manifold. This problem does not occur in the set up used in [16] as the meshes are manually processed to be watertight and manifold without self-intersections.

Second, lower results stem from the fact that our dataset provides relatively few ground truth annotations for mesh faces. As a matter of fact, the original 3D textured mesh is dense enough and provides more than the required number of annotated triangles for a reliable supervision during training as shown in Chapter 1. Nonetheless, the pre-processing step, which consists in reducing the size of the input mesh by 1 order of magnitude to alleviate the computational complexity, magnifies the need for more training examples. Similar results were reported in Section 4.3.3 for point clouds over-segmentation when we used the small vKITTI 3D dataset [223] instead of S3DIS dataset [23] which is much larger.

4.5 Conclusion

Throughout this chapter, we have presented an unified deep-learning-based approach for over-segmenting 3D point clouds into superpoints and 3D meshes into superfacets. To do so, state-of-the-art methods for point cloud embedding PointNet [14] and 3D mesh embedding MeshCNN [16] were tailored to extract local features of points/mesh faces. Both our supervised superpoint and superfacet methods are the first over-segmentation approaches based on features learned by deep metric learning. Our approach significantly improves state-of-the-art of 3D mesh and 3D point cloud over-segmentation. While our learned point cloud over-segmentation combined to a superoint-based method for semantic segmentation [17] have shown impressive results outperforming state-of-the-art algorithms in this task, we were not able to extend this success for 3D meshes due to time constraint. However, we have demonstrated that our learned superfacets outperform competing methods by a large margin. This is a promising indicator that our method is most likely susceptible to perform well for the task of semantic segmentation. Finally, we have released our source code for 3D point cloud over-segmentation to the community to ease reproducibility. The code materials are available in the same repository used for the initial release of SPG [17] [7].

[7] https://github.com/loicland/superpoint_graph

Chapter

5 Conclusion

Chapter content

5.1 Summary

The aim of this thesis was to develop a global geo-localized map, made of a set of 3D representations based on geometric, photometric, and semantic information in order to provide localization and navigation services for remote agents. To do so, we have conducted a comparative study to identify the best suited 3D representation of such Geographical Information System (GIS) achieving a decent trade-off between efficiency and fidelity. Based on mature and well-grounded notions in the field of computer graphics, we have constructed a high quality large-scale 3D textured mesh using mobile mapping images and LiDAR scans collected during a mapping acquisition campaign in Rouen, France.

Whilst representing the 3D map as a textured mesh offers unmatched advantages in terms of expressiveness by combining photometric and geometric properties, the lack of semantic information makes it less reliable in a context of autonomous navigation. As a first step to overcome this limitation, we have proposed the first outdoor multi-modal, multi-format benchmark encapsulating geolocalized RGB perspective and panoramic images, spherical depth and LiDAR intensity maps, 3D point clouds and textured meshes along with their respective ground truth annotations.

Our final contribution consisted in developing the first supervised over-segmentation methods operating on 3D point clouds and 3D textured meshes in order to efficiently handle large-scale acquisitions consistently. We have demonstrated the superiority of our over-segmentation approach compared to the existing methods in the literature on two public point cloud benchmarks S3DIS [23] and vKITTI3D [58] as well as on our own multi-modal dataset for 3D meshes. By plugging the learned superpoints to a semantic segmentation algorithm SPG [17] based on over-segmentation, we set a new state-of-the-art for the task

of 3D point cloud semantic segmentation. To ease the reproducibility of our work by the computer vision community and boost further research in this area, we have released our source code for all the developed algorithms presented in this thesis.

Throughout our research experience during the past three years, we have learnt to improve our time management skills in addition to the newly acquired knowledge. As the results of our work were needed by the project partners (as depicted in the project functional diagram illustrated in the introduction chapter), we were supposed to handle tight deadlines while maintaining high standards of work. These constraints have made us question and rethink our research strategies as well as our design choices several times. Finally, we ended up being fairly satisfied with the obtained results. We believe that this study can be subject to many interesting upgrades in the future. In the following section, we discuss open problems related to our work and eventual future enhancements on the quantitative and qualitative levels.

5.2 Open problems & Future work

5.2.1 pLaTINUM dataset

While the proposed dataset in Chapter 2 offers rich and diverse modalities to assess the task of outdoor scene understanding, it remains limited in size compared to competing large-scale datasets provided by industries such as Baidu [21] and Honda [20]. This can be clearly explained by the discrepancy in resources invested in producing such benchmarks. Nevertheless, we believe that we are able to enlarge the current benchmark by providing further annotations to help prototyping scene understanding methods. Since manual annotation is a tedious task, we believe that automatic image annotation methods (AIA) [79] in the literature can be leveraged to facilitate this task. Furthermore, in addition to the current available tasks, we intend to integrate a finer level of intra-building annotation to include the task of façade parsing into our benchmark. In the latter application, more detailed classes such as windows and doors belonging to the class building have to be also annotated.

A large spectrum of testing scenarios can be elaborated to deeply study the influence of each modality on the overall performance in terms of semantic segmentation. Unfortunately due to time constraints, we were not able to carry out those experiments. Some of the interesting experiments consist in adopting a multi-modal fusion-based strategy to evaluate, for instance, the effect of intensity maps and depth maps fused with RGB images. Then we compare the latter to 3D modality (either the textured mesh or the colored point cloud) to see if it is worth the effort to use pure 3D data as an alternative to 2D or 2.5D data. To the best of our knowledge, in the industrial circle, it remains not clear which of those modalities achieve the best compromise between efficiency and accuracy. For example, Tesla driver-less cars rely exclusively on camera for perception. Others such as UBER autonomous cars are equipped with LiDAR technology. Although using the dataset presented in this thesis seems to be not sufficient to draw such strong conclusions, we believe it will give at least valuable initial insights on which modality should be studied further in priority as potential future work.

5.2.2 3D map as textured mesh

More related to autonomous driving, the developed textured 3D map is far from being a ready-to-use byproduct which could be deployed in real driving conditions. A high definition map is a more complicated entity that comprises additional information within geometric, photometric and semantic layers that are beyond the scope of our study. For instance, valuable semantic classes such as road centerlines, curbs and lanes are not taken into consideration in the semantic layer.

One can argue that constructing a HD map as a 3D textured mesh can be considered as an overwhelming task since it involves surface reconstruction which is a problem arguably as hard as semantic segmentation. However, since this map is supposed to be reconstructed as an off-line independent step, we believe it should not be a problem as long as real-time constraints are not required. However, detecting changes and updating the textured mesh accordingly is by far a more complicated task. It should be noted that many industrial leaders in geo-spatial technologies have started to commercialize this solution(*i.e.* the textured mesh as a geometric layer in high definition maps). *Sanborn* [1], for instance, relies on oblique imagery and high precision aerial and mobile LiDAR to reconstruct such maps.

We do not consider our surface reconstruction method as a cutting-edge algorithm outperforming state-of-the-art methods, as much as it is a demonstration how key information provided by the scanner, which are ignored most of the time, can be conveniently leveraged to reconstruct high quality large-scale surface. However, the consequent shortcomings, especially the non reconstructed occluded objects, should be tackled explicitly. One way to address this issue is to use a fusion-based approach such as WaSURE [232] to reconstruct the surface. In this category of methods, various modalities can be merged together to faithfully reconstruct a 3D surface. Since camera and LiDAR have different geometries, objects that are occluded to the LiDAR can be visible to the camera. Therefore, we believe that using both LiDAR scans along with point clouds generated from images might be a reasonable strategy to improve the quality of the reconstruction.

5.2.3 Supervised over-segmentation

The usage of 2D superpixels in a variety of computer vision tasks has remarkably decreased these last years due to the surge of scalable deep learning methods. For 3D data, we believe that over-segmentation remains a key preprocessing step to several applications ranging from data annotation to semantic segmentation and object detection.

Our over-segmentation method is a graph-based approach which relies on the construction of a graph before partitioning. In our related work study, we have broadly discussed the existing graph typologies. We have shown in our experiments, that the topology of the reconstructed graph depends on the data structure. We believe this aspect should be studied more in-depth. Experimentally, we found that the heaviest computational step in our over-segmentation framework remains solving the Generalized Minimal Partition Problem (GMPP). A recent work of Raguet and Landrieu [195] introduced a parallel version of the cut-pursuit algorithm [178] solving functional involving the total variation more efficiently. We think integrating this parallel version will improve computational

[1] www.sanborn.com

time required by the over-segmentation.

Even though our local mesh embedding network LTME shows impressive results for the task of textured mesh over-segmentation, we think that there are many other interesting ways to embed a 3D mesh. In our work, we opted for MeshCNN [16] backbone network which was originally designed to embed faces of a triangular mesh using edge convolution and pooling. However, more versatile approaches could have been used instead such as graphs. Due to technical problems, we were not able to replicate the pooling layers introduced in MeshCNN, as after each edge collapse we have to ensure that the resulted mesh at this step remain manifold so that the next convolution operation holds. On the other hand, it has been shown that pooling is beneficial for CNNs on grid-like data as it allows for receptive field expansion and thus more generalization capability [233]. We consider that not using pooling is a limitation in our MeshCNN-based network as the extraction of edge features at different scales was not feasible due to the mesh topological constraints. The latter shortcoming, would have not been met if we had considered the 3D mesh as a graph. For instance , Simonovsky *et al.* [202] perform pooling through graph coarsening which is agnostic to mesh topology.

It should be noted that compared to the number of parameters of our LME network (around 25.10^4) , CED number of parameters is much higher (around $1,3.10^6$. We believe CED network can be substantially simplified by using a lightweight encoder/decoder with depth-wise separable convolutions as proposed in Bahl *et al.* [234] for instance.

In summary, this study have showed us that the majority of the existing research is basically oriented towards building an explicit map during an exploration phase and then acting based on that representation. However, it should be emphasized that completely different approaches are currently being explored targetting essentially autonomous *mapless* navigation [235]. In this class of methods, an agent is capable to navigate an entire city relying exclusively on visual observation using an end-to-end deep reinforcement learning network. We believe that autonomous mapless navigation is a clearly promising area of research that should be investigated further despite the difficulties that might get in the way during the real world deployement phase since the validation process involves the security of human beings.

Publications

The work presented in this thesis has successfully passed a double-blind review process and hence been presented in series of publications amongst which are papers for national and international conferences in photogrammetry, computer vision and machine learning. In the following, we list these publications:

5.3 National conferences

M. Boussaha, E. Fernandez-Moral, B. Vallet, P. Rives. On the Production of Semantic and Textured 3D Meshes of Large Scale Urban Environments from Mobile Mapping Images and LiDAR Scans. *Reconnaissance des Formes, Image, Apprentissage et Perception (RFIAP), Marne-la-Vallée, France, June 2018.*

5.4 International conferences

M. Boussaha, B. Vallet, P. Rives. Large Scale Textured Mesh Reconstruction from Mobile Mapping Images and LiDAR Scans. In: *The international Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS) IV-2, pp.49-56, Riva del Garda, Italie, June 2018.*

L. Landrieu and M. Boussaha. Point Cloud Oversegmentation With Graph-Structured Deep Metric Learning. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7440–7449, Long Beach, CA, USA, June 16-20, 2019.*

L. Landrieu and M. Boussaha. Supervized Segmentation With Graph Structured Deep Metric Learning. In: *International Conference on Machine Learning Workshops (ICML), Long Beach, CA, USA, June 10-15, 2019.*

Appendix

Illustration of the textured 3D mesh

In Figure 1, we show large scale high resolution samples from our reconstructed 3D textured mesh.

Semantic segmentation of point clouds

In Table 1, we show the detailed semantic segmentation IoU per-class scores on the dataset S3DIS [23]. We also made a video illustration which can be accessed at https://youtu.be/bKxU03tjLJ4.

Method	OA	mAc	mIoU	ceilin	floor	wall	bean	colum	windo	door	chair	table	bookca	sofa	board	clutter
A5 PointNet [14]	–	49.0	41.1	88.8	97.3	69.8	**0.1**	3.9	46.3	10.8	52.6	58.9	40.3	5.9	26.4	33.2
A5 SEGCloud [236]	–	57.4	48.9	90.1	96.1	69.9	0.0	18.4	38.4	23.1	75.9	70.4	58.4	40.9	13.0	41.6
A5 PointCNN [227]	85.9	63.9	57.3	**92.3**	**98.2**	79.4	0.0	17.6	22.8	62.1	80.6	74.4	66.7	31.7	**62.2**	**56.7**
A5 SPG [213]	86.4	66.5	58.0	89.4	96.9	78.1	0.0	**42.8**	48.9	**61.6**	84.7	75.4	69.8	**52.6**	2.1	52.2
A5 SSP + SPG (ours)	**87.9**	**68.2**	**61.7**	91.9	96.7	**80.8**	**0.0**	28.8	**60.3**	57.2	**85.5**	**76.4**	**70.5**	49.1	51.6	53.3
PointNet [14] in [237]	78.5	66.2	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	42.0	54.1	38.2	9.6	29.4	35.2
Engelmann *et al.* [237]	81.1	66.4	49.7	90.3	92.1	67.9	44.7	24.2	52.3	51.2	47.4	58.1	39.0	6.9	30.0	41.9
Engelamnn in [238]	84.0	67.8	58.3	92.1	90.4	78.5	37.8	35.7	51.2	65.4	61.6	64.0	51.6	25.6	49.9	53.7
SPG [213]	85.5	73.0	62.1	89.9	95.1	76.4	62.8	47.1	55.3	68.4	73.5	69.2	63.2	45.9	8.7	52.9
PointCNN [227]	**88.1**	75.6	65.4	**94.8**	**97.3**	75.8	**63.3**	51.7	58.4	57.2	69.1	**71.6**	61.2	39.1	52.2	58.6
SSP + SPG (ours)	87.9	**78.3**	**68.4**	91.7	95.5	**80.8**	62.2	**54.9**	**58.8**	**68.4**	**78.4**	69.2	**64.3**	**52.0**	**54.2**	**59.2**

Table 1: *Results on the S3DIS dataset on fold "Area 5" (top) and micro-averaged over all 6 folds (bottom). Intersection over union is shown split per class, with the highest value over all methods in bold.*

Models configuration for semantic segmentation

We used the open-source superpoint-graph implementation [2] without any modification beyond changing the oversegmentation step and some changes in the hyper-parameters.

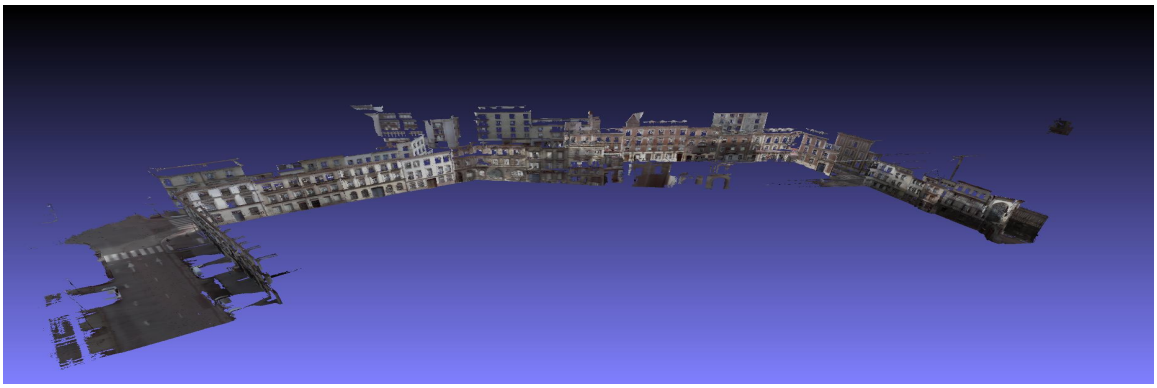
To compensate for the edges missed by the ℓ_0-cut pursuit approximation, due in part to its ignoring the spherical nature of the embeddings, we set the regularization strength $\tilde{\lambda}$ lower than 1 for vKITTI and S3DIS datasets. This help improve the accuracy and border recall. The subsequent decrease in border precision is compensated by the fact that the SPG, through its context leveraging module, can learn to propagate the semantic information to small superpoints. For the same reason, we chose a lower superpoint size for S3DIS from the segmentation experiments.

[2] github/loicland/superpoint-graph

We extended the superpoint graph subsampling threshold to 4-hops instead of 3, because our method SSP tends to produce thin components near interfaces. Since the vKITTI dataset is much smaller than S3DIS, we chose smaller networks to mitigate overfitting.

Illustrations of semantic segmentation results

In this section, we show more illustrations of our over-segmentation method combined with the semantic segmentation method of Landrieu *et al.* [17]. In Figure 2, we show a successful semantic segmentation of a complex scene from S3DIS dataset [23]. Figure 3 shows a failure case in which a white board is over-segmented in too many small superpoints. This makes their classification harder by the semantic segmentation network. In Figure 4, we see a successful semantization of an urban outdoor scene from vKITTI3D. Finally, in Figure 5, we can observe in the background, road signs with high color contrasts, which are segmented in small superpoints. This makes them very hard to classify and they are missed by the semantic segmentation algorithm.



(a) Illustration of a chunk of 350 m of textured mesh

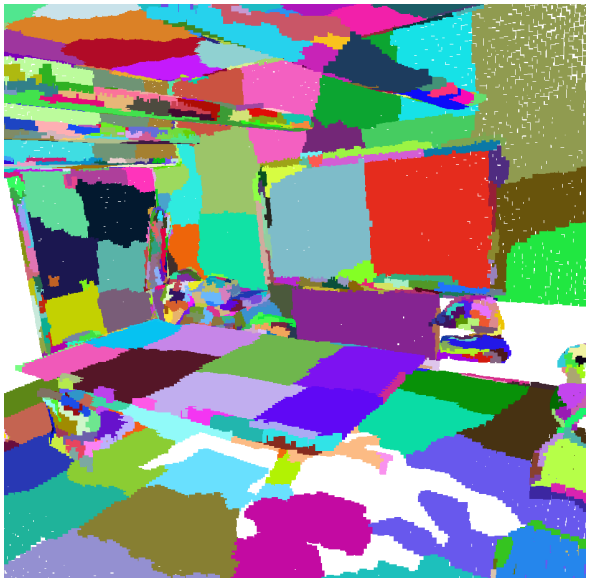
(b) high resolution zoomed region

(c) high resolution zoomed region

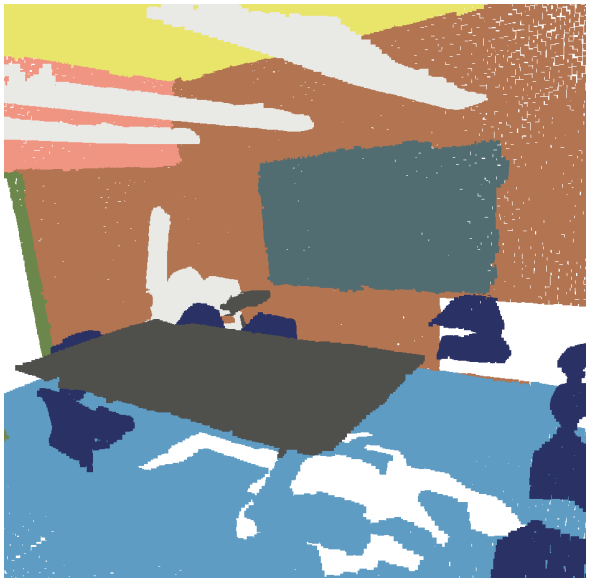
(d) high resolution zoomed region

Figure 1: *Illustration of some textured mesh samples reconstructed from our dataset.*

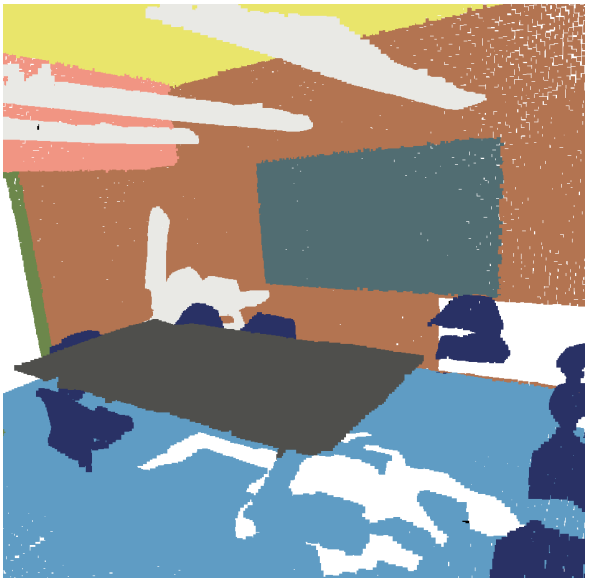
(a) Input point cloud



(b) Over-segmentation



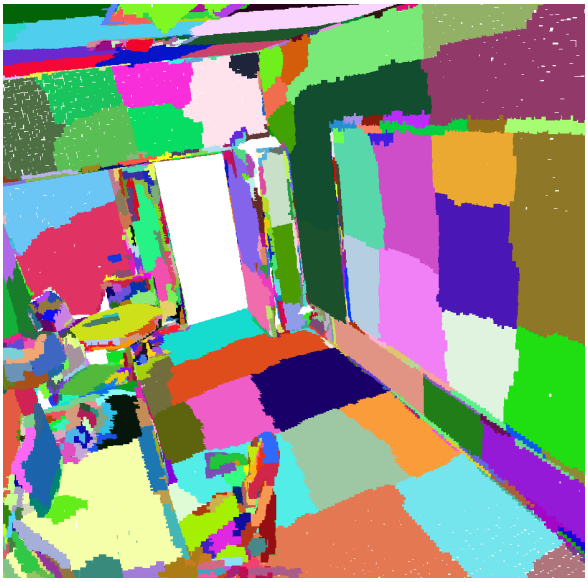
(c) Semantic segmentation



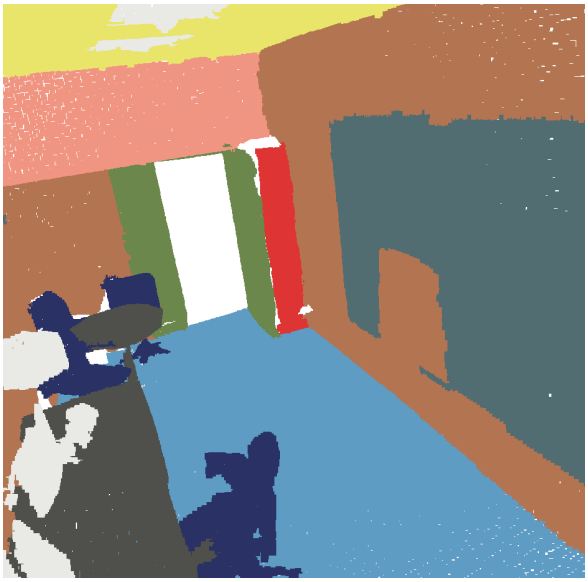
(d) Ground truth

Figure 2: *Illustration of a successful semantic segmentation of a complex scene from S3DIS dataset [23]*

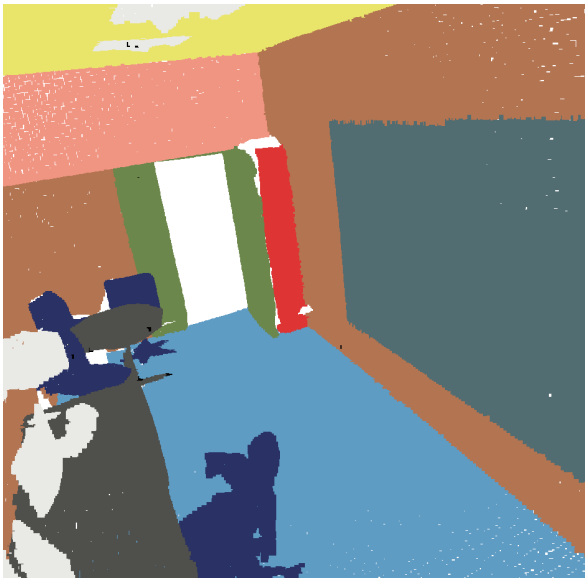
(a) Input point cloud



(b) Over-segmentation

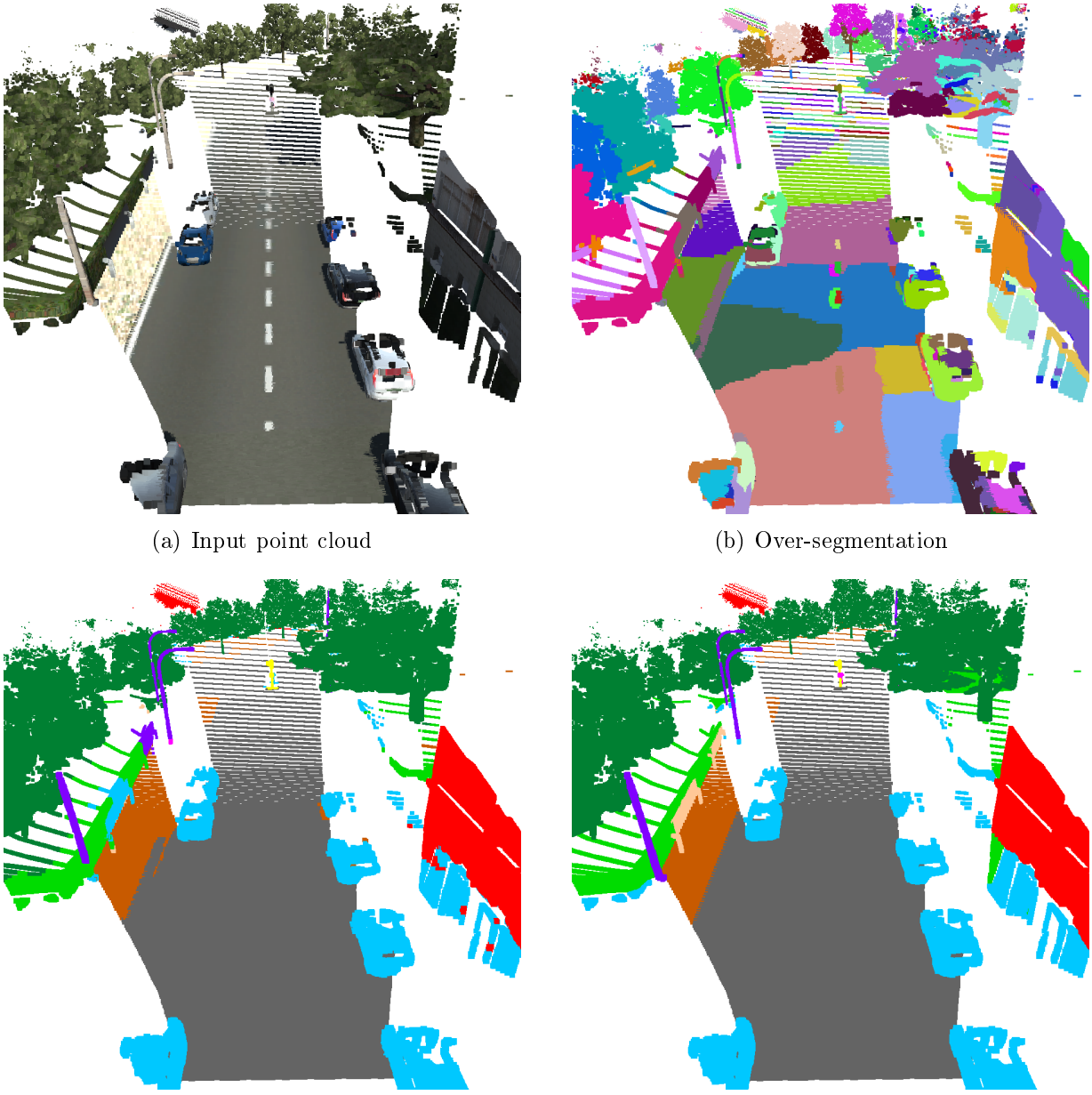


(c) Semantic segmentation



(d) Ground truth

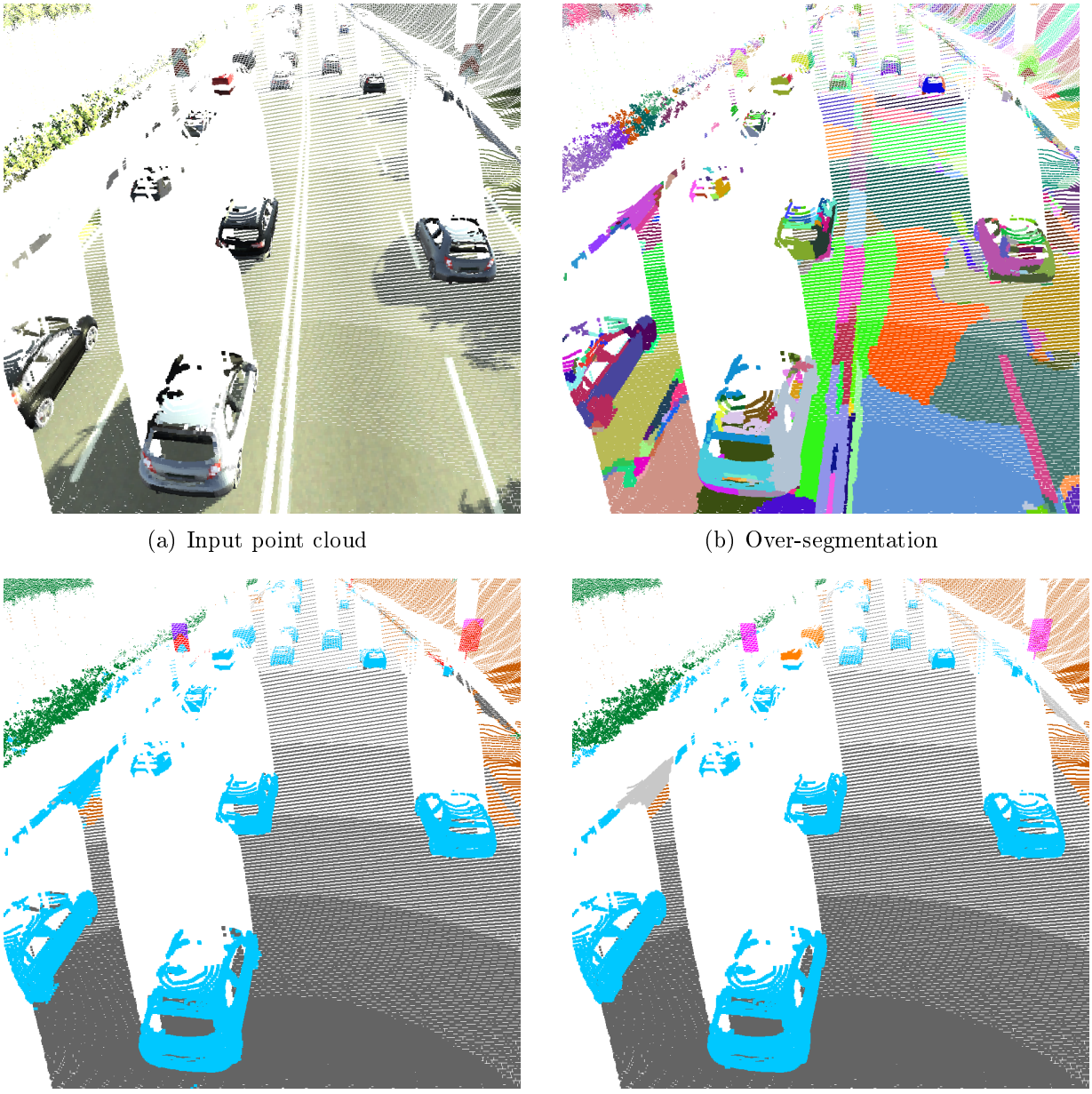
Figure 3: *Illustration of a failure semantic segmentation of a scene from S3DIS dataset [23]*



(a) Input point cloud (b) Over-segmentation

(c) Semantic segmentation (d) Ground truth

Figure 4: *Illustration of a successful semantic segmentation of a complex scene from vKITTI3D dataset [58]*



(a) Input point cloud

(b) Over-segmentation

(c) Semantic segmentation

(d) Ground truth

Figure 5: *Illustration of a failure case of semantic segmentation of a scene from vKITTI3D dataset [58]*

Bibliography

References for Chapter 1: General introduction

[1] James Max Kanter and Kalyan Veeramachaneni. "Deep feature synthesis: Towards automating data science endeavors". In: *2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, Campus des Cordeliers, Paris, France, October 19-21, 2015*. 2015, pp. 1–10. DOI: 10.1109/DSAA.2015.7344858 *Cited on page 1.*

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. 2012, pp. 1106–1114 *Cited on pages 2, 11.*

[3] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. "Geometric Deep Learning: Going beyond Euclidean data". In: *IEEE Signal Process. Mag.* 34.4 (2017), pp. 18–42. DOI: 10.1109/MSP.2017.2693418 *Cited on pages 2, 87, 90.*

[4] C Vincent Tao. "Mobile mapping technology for road network data acquisition". In: *Journal of Geospatial Engineering* 2.2 (2000), pp. 1–14 *Cited on page 6.*

[5] Aldino Rizaldy and Wildan Firdaus. "Direct georeferencing: A new standard in photogrammetry for high accuracy mapping". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 39 (2012), B1 *Cited on page 6.*

[6] Alexander Kirillov, Kaiming He, Ross B. Girshick, Carsten Rother, and Piotr Dollár. "Panoptic Segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, pp. 9404–9413. DOI: 10.1109/CVPR.2019.00963 *Cited on pages 9, 10.*

[7] Joachim Niemeyer, Jan Dirk Wegner, Clément Mallet, Franz Rottensteiner, and Uwe Soergel. "Conditional random fields for urban scene classification with full waveform LiDAR data". In: *ISPRS Conference on Photogrammetric Image Analysis*. Springer. 2011, pp. 233–244 *Cited on page 10.*

[8] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Trans. Robotics* 31.5 (2015), pp. 1147–1163. DOI: 10.1109/TRO.2015.2463671 *Cited on pages 11, 48, 50, 52.*

[9] Thomas Schops, Torsten Sattler, and Marc Pollefeys. "BAD SLAM: Bundle Adjusted Direct RGB-D SLAM". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 134–144 *Cited on pages 11, 48, 50, 52.*

[10] Jakub Porebski, Krzysztof Kogut, Paweł Markiewicz, and Paweł Skruch. "Occupancy grid for static environment perception in series automotive applications". In: *IFAC-PapersOnLine* 52.8 (2019), pp. 148–153 *Cited on pages 11, 48, 50, 52.*

[11] Christoph Brand, Martin J. Schuster, Heiko Hirschmüller, and Michael Suppa. "Stereo-vision based obstacle mapping for indoor/outdoor SLAM". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*. 2014, pp. 1846–1853. DOI: 10.1109/IROS.2014.6942805 *Cited on pages 11, 51, 52.*

[12] Jacek Zienkiewicz, Akis Tsiotsios, Andrew J. Davison, and Stefan Leutenegger. "Monocular, Real-Time Surface Reconstruction Using Dynamic Level of Detail". In: *Fourth International Conference on 3D Vision, 3DV 2016, Stanford, CA, USA, October 25-28, 2016*. 2016, pp. 37–46. DOI: 10.1109/3DV.2016.82 *Cited on pages 11, 48, 50, 52.*

[13] Gabriele Costante, Christian Forster, Jeffrey Delmerico, Paolo Valigi, and Davide Scaramuzza. "Perception-aware path planning". In: *arXiv:1605.04151* (2016) *Cited on page 11.*

[14] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *CVPR, IEEE* 1.2 (2017) *Cited on pages 11, 74, 75, 81, 87, 91, 92, 96, 98, 103, 113, 128, 135.*

[15] Yin Zhou and Oncel Tuzel. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 4490–4499. DOI: 10.1109/CVPR.2018.00472 *Cited on pages 11, 74, 81, 87, 88, 91.*

[16] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. "MeshCNN: a network with an edge". In: *ACM Trans. Graph.* 38.4 (2019), 90:1–90:12. DOI: 10.1145/3306346.3322959 *Cited on pages 11, 74, 75, 81, 87, 95, 96, 98, 114, 117, 120, 127, 128, 132.*

[17] Loic Landrieu and Martin Simonovsky. "Large-Scale Point Cloud Semantic Segmentation With Superpoint Graphs". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 4558–4567. DOI: 10.1109/CVPR.2018.00479 *Cited on pages 11, 13, 42, 75, 93, 96, 106, 128, 129, 136.*

[18] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, Sabine Süsstrunk, et al. "SLIC superpixels compared to state-of-the-art superpixel methods". In: *IEEE transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012) *Cited on pages 11, 75, 77–80, 107.*

[19] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. "TrafficPredict: Trajectory Prediction for Heterogeneous Traffic-Agents". In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. 2019, pp. 6120–6127 *Cited on pages 11, 16, 26, 28.*

[20] Abhishek Patil, Srikanth Malla, Haiming Gang, and Yi-Ting Chen. "The H3D Dataset for Full-Surround 3D Multi-Object Detection and Tracking in Crowded Urban Scenes". In: *CoRR* abs/1903.01568 (2019). arXiv: 1903.01568 *Cited on pages 11, 16, 26–28, 130.*

[21] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. "nuScenes: A multimodal dataset for autonomous driving". In: *CoRR* abs/1903.11027 (2019). arXiv: 1903.11027 *Cited on pages 11, 16, 26, 28, 81, 130.*

[22] Angel X. Chang, Angela Dai, Thomas A. Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. "Matterport3D: Learning from RGB-D Data in Indoor Environments". In: *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017*. 2017, pp. 667–676. DOI: 10.1109/3DV.2017.00081 *Cited on pages 11, 16, 27, 35, 49.*

[23] Iro Armeni, Ozan Sener, Amir Roshan Zamir, Helen Jiang, Ioannis K. Brilakis, Martin Fischer, and Silvio Savarese. "3D Semantic Parsing of Large-Scale Indoor Spaces". In: *CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016. DOI: 10.1109/CVPR.2016.170 *Cited on pages 11, 106, 128, 129, 135, 136, 138, 139.*

[24] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. "ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 2432–2443. DOI: 10.1109/CVPR.2017.261 *Cited on pages 11, 16, 27, 49, 81.*

[25] Nikolaos Zioulis, Antonis Karakottas, Dimitrios Zarpalas, and Petros Daras. "OmniDepth: Dense Depth Estimation for Indoors Spherical Panoramas". In: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VI*. 2018, pp. 453–471. DOI: 10.1007/978-3-030-01231-1_28 *Cited on pages 12, 35.*

[26] Jingwei Huang, Haotian Zhang, Li Yi, Thomas A. Funkhouser, Matthias Nießner, and Leonidas J. Guibas. "TextureNet: Consistent Local Parametrizations for Learning From High-Resolution Signals on Meshes". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, pp. 4440–4449. DOI: 10.1109/CVPR.2019.00457 *Cited on pages 12, 87, 94.*

[27] M. Boussaha, B. Vallet, and P. Rives. "Large Scale Textured Mesh Reconstruction From Mobile Mapping Images and LiDAR Scans". In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* IV-2 (2018), pp. 49–56. DOI: 10.5194/isprs-annals-IV-2-49-2018 *Cited on page 12.*

[28] Loic Landrieu and Mohamed Boussaha. "Point Cloud Oversegmentation With Graph-Structured Deep Metric Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, pp. 7440–7449. DOI: 10.1109/CVPR.2019.00762 *Cited on page 13.*

[29] Loic Landrieu and Mohamed Boussaha. "Supervized Segmentation with Graph-Structured Deep Metric Learning". In: *International Conference on Machine Learning, ICML Workshops 2019, Long Beach, CA, USA, June 10-15, 2019* (2019) *Cited on page 13.*

References for Chapter 2: A multi-modal dataset for urban scene understanding

[17] Loic Landrieu and Martin Simonovsky. "Large-Scale Point Cloud Semantic Segmentation With Superpoint Graphs". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 4558–4567. DOI: 10.1109/CVPR.2018.00479 *Cited on pages 11, 13, 42, 75, 93, 96, 106, 128, 129, 136.*

[19] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. "TrafficPredict: Trajectory Prediction for Heterogeneous Traffic-Agents". In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. 2019, pp. 6120–6127 *Cited on pages 11, 16, 26, 28.*

[20] Abhishek Patil, Srikanth Malla, Haiming Gang, and Yi-Ting Chen. "The H3D Dataset for Full-Surround 3D Multi-Object Detection and Tracking in Crowded Urban Scenes". In: *CoRR* abs/1903.01568 (2019). arXiv: 1903.01568 *Cited on pages 11, 16, 26–28, 130.*

[21] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. "nuScenes: A multimodal dataset for autonomous driving". In: *CoRR* abs/1903.11027 (2019). arXiv: 1903.11027 *Cited on pages 11, 16, 26, 28, 81, 130.*

[22] Angel X. Chang, Angela Dai, Thomas A. Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. "Matterport3D: Learning from RGB-D Data in Indoor Environments". In: *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017*. 2017, pp. 667–676. DOI: 10.1109/3DV.2017.00081 *Cited on pages 11, 16, 27, 35, 49.*

[24] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. "ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 2432–2443. DOI: 10.1109/CVPR.2017.261 *Cited on pages 11, 16, 27, 49, 81.*

[25] Nikolaos Zioulis, Antonis Karakottas, Dimitrios Zarpalas, and Petros Daras. "OmniDepth: Dense Depth Estimation for Indoors Spherical Panoramas". In: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VI*. 2018, pp. 453–471. DOI: 10.1007/978-3-030-01231-1_28 *Cited on pages 12, 35.*

[30] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, Montreal, Quebec, Canada*. 2015, pp. 91–99 *Cited on page 16.*

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90 *Cited on page 16.*

[32] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. "Pyramid Scene Parsing Network". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 6230–6239. DOI: 10.1109/CVPR.2017.660 *Cited on page 16.*

[33] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 2261–2269. DOI: 10.1109/CVPR.2017.243 *Cited on page 16.*

[34] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. "Mask R-CNN". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322 *Cited on page 16.*

[35] Yuan Wang, Tianyue Shi, Peng Yun, Lei Tai, and Ming Liu. "PointSeg: Real-Time Semantic Segmentation Based on 3D LiDAR Point Cloud". In: *CoRR* abs/1807.06288 (2018). arXiv: 1807.06288 *Cited on page 16.*

[36] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. "SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud". In: *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. 2018, pp. 1887–1893. DOI: 10.1109/ICRA.2018.8462926 *Cited on page 16.*

[37] Brian H Wang, Wei-Lun Chao, Yan Wang, Bharath Hariharan, Kilian Q Weinberger, and Mark Campbell. "LDLS: 3-D Object Segmentation Through Label Diffusion From 2-D Images". In: *IEEE Robotics and Automation Letters* 4.3 (2019), pp. 2902–2909 *Cited on page 16.*

[38] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. 2009, pp. 248–255. DOI: 10.1109/CVPRW.2009.5206848 *Cited on pages 16, 82.*

[39] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft COCO: Common Objects in Context". In: *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*. 2014, pp. 740–755. DOI: 10.1007/978-3-319-10602-1_48 *Cited on page 16.*

[40] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, pp. 3213–3223. DOI: 10.1109/CVPR.2016.350 *Cited on pages 16–20, 40.*

[41] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. "The ApolloScape Dataset for Autonomous Driving". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 954–960. DOI: 10.1109/CVPRW.2018.00141 *Cited on pages 16, 18, 20, 36, 40.*

[42] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. "Segmentation and Recognition Using Structure from Motion Point Clouds". In: *Computer Vision - ECCV 2008, 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part I*. 2008, pp. 44–57. DOI: 10.1007/978-3-540-88682-2_5 *Cited on page 16.*

[43] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. "BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling". In: *CoRR* abs/1805.04687 (2018). arXiv: 1805.04687 *Cited on pages 16, 19, 20.*

[44] Daniel Munoz, J Andrew Bagnell, Nicolas Vandapel, and Martial Hebert. "Contextual classification with functional max-margin markov networks". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 975–982 *Cited on pages 16, 21–23.*

[45] Andrés Serna, Beatriz Marcotegui, François Goulette, and Jean-Emmanuel Deschaud. "Paris-rue-Madame Database - A 3D Mobile Laser Scanner Dataset for Benchmarking Urban Detection, Segmentation and Classification Methods". In: *ICPRAM 2014 - Proceedings of the 3rd International Conference on Pattern Recognition Applications and Methods, ESEO, Angers, Loire Valley, France, 6-8 March, 2014*. 2014, pp. 819–824. DOI: 10.5220/0004934808190824 *Cited on pages 16, 21–23.*

[46] Bruno Vallet, Mathieu Brédif, Andrés Serna, Beatriz Marcotegui, and Nicolas Paparoditis. "TerraMobilita/iQmulus urban point cloud analysis benchmark". In: *Computers & Graphics* 49 (2015), pp. 126–133. DOI: 10.1016/j.cag.2015.03.004 *Cited on pages 16, 21–23, 41.*

[47] Timo Hackel, Nikolay Savinov, Lubor Ladicky, Jan Dirk Wegner, Konrad Schindler, and Marc Pollefeys. "Semantic3D.net: A new Large-scale Point Cloud Classification Benchmark". In: *CoRR* abs/1704.03847 (2017). arXiv: 1704.03847 *Cited on pages 16, 21–23.*

[48] 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: *I. J. Robotics Res.* 37.6 (2018), pp. 545–557. DOI: 10.1177/0278364918767506 *Cited on pages 16, 22, 23, 41.*

[49] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*. 2012, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074 *Cited on pages 16, 23, 25, 26, 28, 43, 44, 106.*

[50] Yukyung Choi, Namil Kim, Soonmin Hwang, Kibaek Park, Jae Shin Yoon, Kyounghwan An, and In So Kweon. "KAIST Multi-Spectral Day/Night Data Set for Autonomous and Assisted Driving". In: *IEEE Trans. Intelligent Transportation Systems* 19.3 (2018), pp. 934–948. DOI: 10.1109/TITS.2018.2791533 *Cited on pages 16, 26, 28.*

[51] Iro Armeni, Sasha Sax, Amir Roshan Zamir, and Silvio Savarese. "Joint 2D-3D-Semantic Data for Indoor Scene Understanding". In: *CoRR* abs/1702.01105 (2017). arXiv: 1702.01105 *Cited on pages 16, 27, 41.*

[52] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 2017, pp. 5000–5009. DOI: 10.1109/ICCV.2017.534 *Cited on pages 18–20.*

[53] Qishen Ha, Kohei Watanabe, Takumi Karasawa, Yoshitaka Ushiku, and Tatsuya Harada. "MFNet: Towards real-time semantic segmentation for autonomous vehicles with multi-spectral scenes". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*. 2017, pp. 5108–5115. DOI: 10.1109/IROS.2017.8206396 *Cited on page 19.*

[54] Takumi Karasawa, Kohei Watanabe, Qishen Ha, Antonio Tejero-de-Pablos, Yoshitaka Ushiku, and Tatsuya Harada. "Multispectral Object Detection for Autonomous Vehicles". In: *Proceedings of the on Thematic Workshops of ACM Multimedia 2017, Mountain View, CA, USA, October 23 - 27, 2017*. 2017, pp. 35–43. DOI: 10.1145/3126686.3126727 *Cited on pages 19, 20.*

[55] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. "CityPersons: A Diverse Dataset for Pedestrian Detection". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 4457–4465. DOI: 10.1109/CVPR.2017.474 *Cited on pages 19, 20.*

[56] Germán Ros, Laura Sellart, Joanna Materzynska, David Vázquez, and Antonio M. López. "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, pp. 3234–3243. DOI: 10.1109/CVPR.2016.352 *Cited on pages 23–25.*

[57] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. "Virtual worlds as proxy for multi-object tracking analysis". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4340–4349 *Cited on pages 23, 24, 119, 123.*

[58] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. "Exploring Spatial Context for 3D Semantic Segmentation of Point Clouds". In: *IEEE International Conference on Computer Vision, 3DRMS Workshop, ICCV*. 2017 *Cited on pages 23, 24, 129, 140, 141.*

[59] Stephan R. Richter, Zeeshan Hayder, and Vladlen Koltun. "Playing for Benchmarks". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 2017, pp. 2232–2241. DOI: 10.1109/ICCV.2017.243 *Cited on page 24.*

[60] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio López, and Vladlen Koltun. "CARLA: An Open Urban Driving Simulator". In: *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*. 2017, pp. 1–16 *Cited on pages 24, 25, 30, 43, 44.*

[61] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. "1 year, 1000 km: The Oxford RobotCar dataset". In: *I. J. Robotics Res.* 36.1 (2017), pp. 3–15. DOI: 10.1177/0278364916679498 *Cited on pages 27, 28.*

[62] Yiping Chen, Jingkang Wang, Jonathan Li, Cewu Lu, Zhipeng Luo, Han Xue, and Cheng Wang. "LiDAR-Video Driving Dataset: Learning Driving Policies Effectively". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 5870–5878. DOI: 10.1109/CVPR.2018.00615 *Cited on page 27.*

[63] Hojung Jung, Yuki Oto, Óscar Martínez Mozos, Yumi Iwashita, and Ryo Kurazume. "Multi-modal panoramic 3D outdoor datasets for place categorization". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016*. 2016, pp. 4545–4550. DOI: 10.1109/IROS.2016.7759669 *Cited on pages 27, 28.*

[64] Nicolas Paparoditis, Jean-Pierre Papelard, Bertrand Cannelle, Alexandre Devaux, Bahman Soheilian, Nicolas David, and Erwann Houzay. "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* 200.1 (2012), pp. 69–79 *Cited on page 28.*

[65] Grégoire Payen de La Garanderie, Amir Atapour Abarghouei, and Toby P. Breckon. "Eliminating the Blind Spot: Adapting 3D Object Detection and Monocular Depth Estimation to 360 Panoramic Imagery". In: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*. 2018, pp. 812–830. DOI: 10.1007/978-3-030-01261-8_48 *Cited on pages 30, 42–44.*

[66] Matthew Brown and David G. Lowe. "Automatic Panoramic Image Stitching using Invariant Features". In: *International Journal of Computer Vision* 74.1 (2007), pp. 59–73. DOI: 10.1007/s11263-006-0002-3 *Cited on page 31.*

[67] Patrick Pérez, Michel Gangnet, and Andrew Blake. "Poisson image editing". In: *ACM Trans. Graph.* 22.3 (2003), pp. 313–318. DOI: 10.1145/882262.882269 *Cited on pages 33, 64.*

[68] Daniel Scharstein and Richard Szeliski. "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms". In: *International Journal of Computer Vision* 47.1-3 (2002), pp. 7–42. DOI: 10.1023/A:1014573219977 *Cited on page 35.*

[69] Yasutaka Furukawa and Carlos Hernández. "Multi-View Stereo: A Tutorial". In: *Foundations and Trends in Computer Graphics and Vision* 9.1-2 (2015), pp. 1–148. DOI: 10.1561/0600000052 *Cited on page 35.*

[70] Austin Abrams, Christopher Hawley, and Robert Pless. "Heliometric Stereo: Shape from Sun Position". In: *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part II*. 2012, pp. 357–370. DOI: 10.1007/978-3-642-33709-3_26 *Cited on page 35.*

[71] Dan Xu, Wei Wang, Hao Tang, Hong Liu, Nicu Sebe, and Elisa Ricci. "Structured Attention Guided Convolutional Neural Fields for Monocular Depth Estimation". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 3917–3925. DOI: 10.1109/CVPR.2018.00412 *Cited on page 35.*

[72] Bo Li, Yuchao Dai, and Mingyi He. "Monocular depth estimation with hierarchical fusion of dilated CNNs and soft-weighted-sum inference". In: *Pattern Recognition* 83 (2018), pp. 328–339. DOI: 10.1016/j.patcog.2018.05.029 *Cited on page 35.*

[73] Minhyeok Heo, Jaehan Lee, Kyung-Rae Kim, Han-Ul Kim, and Chang-Su Kim. "Monocular Depth Estimation Using Whole Strip Masking and Reliability-Based Refinement". In: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IV*. 2018, pp. 39–55. DOI: 10.1007/978-3-030-01225-0_3 *Cited on page 35.*

[74] Dan Xu, Elisa Ricci, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. "Monocular Depth Estimation Using Multi-Scale Continuous CRFs as Sequential Deep Networks". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 41.6 (2019), pp. 1426–1440. DOI: 10.1109/TPAMI.2018.2839602 *Cited on page 35.*

[75] Iro Armeni, Ozan Sener, Amir Roshan Zamir, Helen Jiang, Ioannis K. Brilakis, Martin Fischer, and Silvio Savarese. "3D Semantic Parsing of Large-Scale Indoor Spaces". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, pp. 1534–1543. DOI: 10.1109/CVPR.2016.170 *Cited on page 35.*

[76] Namdar Homayounfar, Wei-Chiu Ma, Shrinidhi Kowshika Lakshmikanth, and Raquel Urtasun. "Hierarchical Recurrent Attention Networks for Structured Online Maps". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 3417–3426. DOI: 10.1109/CVPR.2018.00360 *Cited on page 37.*

[77] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. "PointPillars: Fast encoders for object detection from point clouds". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12697–12705 *Cited on page 37.*

[78] Haiyan Guan, Jonathan Li, Yongtao Yu, Michael A. Chapman, and Cheng Wang. "Automated Road Information Extraction From Mobile Laser Scanning Data". In: *IEEE Trans. Intelligent Transportation Systems* 16.1 (2015), pp. 194–205. DOI: 10.1109/TITS.2014.2328589 *Cited on page 38.*

[79] Qimin Cheng, Qian Zhang, Peng Fu, Conghuan Tu, and Sen Li. "A survey and analysis on automatic image annotation". In: *Pattern Recognition* 79 (2018), pp. 242–259. DOI: 10.1016/j.patcog.2018.02.017 *Cited on pages 39, 130.*

[80] Pongsate Tangseng, Zhipeng Wu, and Kota Yamaguchi. "Looking at Outfit to Parse Clothing". In: (2017). arXiv: 1703.01386v1 [cs.CV] *Cited on page 40.*

[81] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua, and Sabine Süsstrunk. "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 34.11 (2012), pp. 2274–2282. DOI: 10.1109/TPAMI.2012.120 *Cited on page 40.*

[82] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. "Unsupervised Monocular Depth Estimation with Left-Right Consistency". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 6602–6611. DOI: 10.1109/CVPR.2017.699 *Cited on page 42.*

[83] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 2017, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244 *Cited on page 43.*

[84] David Eigen, Christian Puhrsch, and Rob Fergus. "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network". In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. 2014, pp. 2366–2374 *Cited on page 43.*

References for Chapter 3: Large scale textured mesh reconstruction

[8] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Trans. Robotics* 31.5 (2015), pp. 1147–1163. DOI: 10.1109/TRO.2015.2463671 *Cited on pages 11, 48, 50, 52.*

[9] Thomas Schops, Torsten Sattler, and Marc Pollefeys. "BAD SLAM: Bundle Adjusted Direct RGB-D SLAM". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 134–144 *Cited on pages 11, 48, 50, 52.*

[10] Jakub Porebski, Krzysztof Kogut, Paweł Markiewicz, and Paweł Skruch. "Occupancy grid for static environment perception in series automotive applications". In: *IFAC-PapersOnLine* 52.8 (2019), pp. 148–153 *Cited on pages 11, 48, 50, 52.*

[11] Christoph Brand, Martin J. Schuster, Heiko Hirschmüller, and Michael Suppa. "Stereo-vision based obstacle mapping for indoor/outdoor SLAM". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*. 2014, pp. 1846–1853. DOI: 10.1109/IROS.2014.6942805 *Cited on pages 11, 51, 52.*

[12] Jacek Zienkiewicz, Akis Tsiotsios, Andrew J. Davison, and Stefan Leutenegger. "Monocular, Real-Time Surface Reconstruction Using Dynamic Level of Detail". In: *Fourth International Conference on 3D Vision, 3DV 2016, Stanford, CA, USA, October 25-28, 2016*. 2016, pp. 37–46. DOI: 10.1109/3DV.2016.82 *Cited on pages 11, 48, 50, 52.*

[22] Angel X. Chang, Angela Dai, Thomas A. Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. "Matterport3D: Learning from RGB-D Data in Indoor Environments". In: *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017*. 2017, pp. 667–676. DOI: 10.1109/3DV.2017.00081 *Cited on pages 11, 16, 27, 35, 49.*

[24] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. "ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 2432–2443. DOI: 10.1109/CVPR.2017.261 *Cited on pages 11, 16, 27, 49, 81.*

[67] Patrick Pérez, Michel Gangnet, and Andrew Blake. "Poisson image editing". In: *ACM Trans. Graph.* 22.3 (2003), pp. 313–318. DOI: 10.1145/882262.882269 *Cited on pages 33, 64.*

[85] Michael Zollhöfer, Patrick Stotko, Andreas Görlitz, Christian Theobalt, Matthias Nießner, Reinhard Klein, and Andreas Kolb. "State of the Art on 3D Reconstruction with RGB-D Cameras". In: *Comput. Graph. Forum* 37.2 (2018), pp. 625–652. DOI: 10.1111/cgf.13386 *Cited on page 48.*

[86] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age". In: *IEEE Transactions on robotics* 32.6 (2016), pp. 1309–1332 *Cited on page 48.*

[87] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sebastien Glaser. "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving". In: *IEEE Trans. Intelligent Vehicles* 2.3 (2017), pp. 194–220. DOI: 10.1109/TIV.2017.2749181 *Cited on page 48.*

[88] Alexandre Chapoulie, Patrick Rives, and David Filliat. "Topological segmentation of indoors/outdoors sequences of spherical views". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*. IEEE, 2012, pp. 4288–4295. DOI: 10.1109/IROS.2012.6385962 *Cited on page 48.*

[89] Alexandre Chapoulie, Patrick Rives, and David Filliat. "Appearance-based segmentation of indoors/outdoors sequences of spherical views". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*. IEEE, 2013, pp. 1946–1951. DOI: 10.1109/IROS.2013.6696614 *Cited on page 48.*

[90] H. Jacky Chang, C. S. George Lee, Y. Charlie Hu, and Yung-Hsiang Lu. "Multi-robot SLAM with topological/metric maps". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 29 - November 2, 2007, Sheraton Hotel and Marina, San Diego, California, USA*. IEEE, 2007, pp. 1467–1472. DOI: 10.1109/IROS.2007.4399142 *Cited on page 48.*

[91] S. B. Nickerson, Piotr Jasiobedzki, David Wilkes, Michael R. M. Jenkin, Evangelos E. Milios, John K. Tsotsos, Allan D. Jepson, and O. N. Bains. "The ARK project: Autonomous mobile robots for known industrial environments". In: *Robotics and Autonomous Systems* 25.1-2 (1998), pp. 83–104. DOI: 10.1016/S0921-8890(98)00032-3 *Cited on pages 48, 50, 52.*

[92] Andrew J. Davison. "Real-Time Simultaneous Localisation and Mapping with a Single Camera". In: *9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France*. 2003, pp. 1403–1410. DOI: 10.1109/ICCV.2003.1238654 *Cited on pages 48, 50, 52.*

[93] Paul M. Newman and Kin Leong Ho. "SLAM-Loop Closing with Visually Salient Features". In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005, April 18-22, 2005, Barcelona, Spain*. 2005, pp. 635–642. DOI: 10.1109/ROBOT.2005.1570189 *Cited on pages 48, 50, 52.*

[94] Yan Lu and Dezhen Song. "Visual Navigation Using Heterogeneous Landmarks and Unsupervised Geometric Constraints". In: *IEEE Trans. Robotics* 31.3 (2015), pp. 736–749. DOI: 10.1109/TRO.2015.2424032 *Cited on pages 48, 50, 52.*

[95] Andreas Nüchter. *3D Robotic Mapping - The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom*. Vol. 52. Springer Tracts in Advanced Robotics. Springer, 2009. ISBN: 978-3-540-89883-2. DOI: 10.1007/978-3-540-89884-9 *Cited on pages 48, 50, 52.*

[96] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. "RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments". In: *Experimental Robotics - The 12th International Symposium on Experimental Robotics, ISER 2010, December 18-21, 2010, New Delhi and Agra, India*. 2010, pp. 477–491. DOI: 10.1007/978-3-642-28572-1_33 *Cited on pages 48, 50, 52.*

[97] Richard A. Newcombe, Steven Lovegrove, and Andrew J. Davison. "DTAM: Dense tracking and mapping in real-time". In: *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*. 2011, pp. 2320–2327. DOI: 10.1109/ICCV.2011.6126513 *Cited on pages 48, 50, 52.*

[98] Matia Pizzoli, Christian Forster, and Davide Scaramuzza. "REMODE: Probabilistic, monocular dense reconstruction in real time". In: *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*. 2014, pp. 2609–2616. DOI: 10.1109/ICRA.2014.6907233 *Cited on pages 48, 50, 52.*

[99] Thomas Whelan, Stefan Leutenegger, Renato F. Salas-Moreno, Ben Glocker, and Andrew J. Davison. "ElasticFusion: Dense SLAM Without A Pose Graph". In: *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*. 2015. DOI: 10.15607/RSS.2015.XI.001 *Cited on pages 48, 50, 52.*

[100] A Elfes. *Occupancy grids: a probabilistic framework for robot perception and navigation [Ph. D. thesis]*. 1989 *Cited on pages 48, 50, 52.*

[101] Nathaniel Fairfield, George Kantor, and David Wettergreen. "Real-Time SLAM with Octree Evidence Grids for Exploration in Underwater Tunnels". In: *J. Field Robotics* 24.1-2 (2007), pp. 03–21. DOI: 10.1002/rob.20165 *Cited on pages 48, 50, 52.*

[102] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew W. Fitzgibbon. "KinectFusion: Real-time dense surface mapping and tracking". In: *10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011, Basel, Switzerland, October 26-29, 2011*. 2011, pp. 127–136. DOI: 10.1109/ISMAR.2011.6092378 *Cited on pages 48, 50, 52.*

[103] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. "Real-time 3D reconstruction at scale using voxel hashing". In: *ACM Trans. Graph.* 32.6 (2013), 169:1–169:11. DOI: 10.1145/2508363.2508374 *Cited on pages 48, 50, 52.*

[104] Olaf Kähler, Victor Adrian Prisacariu, Julien P. C. Valentin, and David W. Murray. "Hierarchical Voxel Block Hashing for Efficient Integration of Depth Images". In: *IEEE Robotics and Automation Letters* 1.1 (2016), pp. 192–197. DOI: 10.1109/LRA.2015.2512958 *Cited on pages 48, 50, 52.*

[105] Amaël Delaunoy and Emmanuel Prados. "Gradient Flows for Optimizing Triangular Mesh-based Surfaces: Applications to 3D Reconstruction Problems Dealing with Visibility". In: *International Journal of Computer Vision* 95.2 (2011), pp. 100–123. DOI: 10.1007/s11263-010-0408-9 *Cited on pages 48, 50, 52.*

[106] Hoang-Hiep Vu, Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. "High Accuracy and Visibility-Consistent Dense Multiview Stereo". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 34.5 (2012), pp. 889–901. DOI: 10.1109/TPAMI.2011.172 *Cited on pages 48, 50, 52.*

[107] Amaël Delaunoy and Marc Pollefeys. "Photometric Bundle Adjustment for Dense Multi-view 3D Modeling". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. 2014, pp. 1486–1493. DOI: 10.1109/CVPR.2014.193 *Cited on pages 48, 50, 52.*

[108] Thomas Whelan, Stefan Leutenegger, Renato F. Salas-Moreno, Ben Glocker, and Andrew J. Davison. "ElasticFusion: Dense SLAM Without A Pose Graph". In: *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*. 2015. DOI: 10.15607/RSS.2015.XI.001 *Cited on pages 48, 50, 52.*

[109] Andrea Romanoni, Daniele Fiorenti, and Matteo Matteucci. "Mesh-based 3D textured urban mapping". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*. 2017, pp. 3460–3466. DOI: 10.1109/IROS.2017.8206186 *Cited on pages 48–50, 52, 58, 66.*

[110] Matthew Berger, Joshua A. Levine, Luis Gustavo Nonato, Gabriel Taubin, and Cláudio T. Silva. "A benchmark for surface reconstruction". In: *ACM Trans. Graph.* 32.2 (2013), 20:1–20:17. DOI: 10.1145/2451236.2451246 *Cited on pages 48, 54.*

[111] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gaël Guennebaud, Joshua A. Levine, Andrei Sharf, and Cláudio T. Silva. "A Survey of Surface Reconstruction from Point Clouds". In: *Comput. Graph. Forum* 36.1 (2017), pp. 301–329. DOI: 10.1111/cgf.12802 *Cited on pages 48, 54, 55, 58.*

[112] Peter Ondrúška, Pushmeet Kohli, and Shahram Izadi. "Mobilefusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones". In: *IEEE transactions on visualization and computer graphics* 21.11 (2015), pp. 1251–1258 *Cited on page 48.*

[113] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration". In: *ACM Transactions on Graphics (ToG)* 36.3 (2017), p. 24 *Cited on page 48.*

[114] Przemyslaw Musialski, Peter Wonka, Daniel G. Aliaga, Michael Wimmer, Luc Van Gool, and Werner Purgathofer. "A Survey of Urban Reconstruction". In: *Eurographics 2012 - State of the Art Reports, Cagliari, Italy, May 13-18, 2012*. 2012, pp. 1–28. DOI: 10.2312/conf/EG2012/stars/001-028 *Cited on pages 48, 52.*

[115] Michael Kaess. "Simultaneous localization and mapping with infinite planes". In: *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*. 2015, pp. 4605–4611. DOI: 10.1109/ICRA.2015.7139837 *Cited on pages 51, 52.*

[116] Yasutaka Furukawa, Carlos Hernández, et al. "Multi-view stereo: A tutorial". In: *Foundations and Trends® in Computer Graphics and Vision* 9.1-2 (2015), pp. 1–148 *Cited on page 53.*

[117] Jan-Michael Frahm, Pierre Fite Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, and Svetlana Lazebnik. "Building Rome on a Cloudless Day". In: *Computer Vision - ECCV 2010, 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV*. 2010, pp. 368–381. DOI: 10.1007/978-3-642-15561-1_27 *Cited on page 53.*

[118] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. "Towards Internet-scale multi-view stereo". In: *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*. 2010, pp. 1434–1441. DOI: 10.1109/CVPR.2010.5539802 *Cited on page 53.*

[119] Rongjun Qin, Jiaojiao Tian, and Peter Reinartz. "3D change detection–approaches and applications". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 122 (2016), pp. 41–56 *Cited on page 53.*

[120] Michael M. Kazhdan, Matthew Bolitho, and Hugues Hoppe. "Poisson surface reconstruction". In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing, Cagliari, Sardinia, Italy, June 26-28, 2006*. 2006, pp. 61–70. DOI: 10.2312/SGP/SGP06/061-070 *Cited on pages 54, 55, 66–68.*

[121] Michael M. Kazhdan and Hugues Hoppe. "Screened poisson surface reconstruction". In: *ACM Trans. Graph.* 32.3 (2013), 29:1–29:13. DOI: 10.1145/2487228.2487237 *Cited on pages 54, 55, 66–68, 71.*

[122] Jonathan C. Carr, Richard K. Beatson, Jon B. Cherrie, Tim J. Mitchell, W. Richard Fright, Bruce C. McCallum, and Tim R. Evans. "Reconstruction and representation of 3D objects with radial basis functions". In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001, Los Angeles, California, USA, August 12-17, 2001*. 2001, pp. 67–76. DOI: 10.1145/383259.383266 *Cited on pages 54, 55.*

[123] David Levin. "Mesh-independent surface interpolation". In: *Geometric modeling for scientific visualization*. Springer, 2004, pp. 37–49 *Cited on pages 54–56.*

[124] Nina Amenta and Yong Joo Kil. "Defining point-set surfaces". In: *ACM Trans. Graph.* 23.3 (2004), pp. 264–270. DOI: 10.1145/1015706.1015713 *Cited on pages 54–56.*

[125] Gaël Guennebaud and Markus H. Gross. "Algebraic point set surfaces". In: *ACM Trans. Graph.* 26.3 (2007), p. 23. DOI: 10.1145/1276377.1276406 *Cited on pages 54–57.*

[126] Yutaka Ohtake, Alexander G. Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. "Multi-level partition of unity implicits". In: *ACM Trans. Graph.* 22.3 (2003), pp. 463–470. DOI: 10.1145/882262.882293 *Cited on pages 54, 56, 57.*

[127] Chun-Xia Xiao. "Multi-Level Partition of Unity Algebraic Point Set Surfaces". In: *J. Comput. Sci. Technol.* 26.2 (2011), pp. 229–238. DOI: 10.1007/s11390-011-9429-2 *Cited on pages 54, 56, 57.*

[128] Z.-Q. Cheng, Y.-Z. Wang, B. Li, Kai Xu, Gang Dang, and S.-Y. Jin. "A Survey of Methods for Moving Least Squares Surfaces". In: *Proceedings of the Eurographics / IEEE VGTC Workshop on Volume Graphics 2008, Los Angeles, California, USA, 2008*. 2008, pp. 9–23. DOI: 10.2312/VG/VG-PBG08/009-023 *Cited on page 55.*

[129] L Grammatikopoulos, I Kalisperakis, G Karras, and E Petsa. "Automatic multi-view texture mapping of 3D surface projections". In: *Proceedings of the 2nd ISPRS International Workshop 3D-ARCH*. 2007, pp. 1–6 *Cited on page 57.*

[130] Marco Callieri, Paolo Cignoni, Massimiliano Corsini, and Roberto Scopigno. "Masked photo blending: Mapping dense photographic data set on high-resolution sampled 3D models". In: *Computers & Graphics* 32.4 (2008), pp. 464–473 *Cited on page 57.*

[131] Victor S. Lempitsky and Denis V. Ivanov. "Seamless Mosaicing of Image-Based Texture Maps". In: *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*. 2007. DOI: 10.1109/CVPR.2007.383078 *Cited on pages 58, 64.*

[132] Ran Gal, Yonatan Wexler, Eyal Ofek, Hugues Hoppe, and Daniel Cohen-Or. "Seamless Montage for Texturing Models". In: *Comput. Graph. Forum* 29.2 (2010), pp. 479–486. DOI: 10.1111/j.1467-8659.2009.01617.x *Cited on page 58.*

[133] Ignacio Garcia-Dorado, Ilke Demir, and Daniel G. Aliaga. "Automatic urban modeling using volumetric reconstruction with surface graph cuts". In: *Computers & Graphics* 37.7 (2013), pp. 896–910. DOI: 10.1016/j.cag.2013.07.003 *Cited on page 58.*

[134] Cédric Allène, Jean-Philippe Pons, and Renaud Keriven. "Seamless image-based texture atlases using multi-band blending". In: *19th International Conference on Pattern Recognition (ICPR 2008), December 8-11, 2008, Tampa, Florida, USA*. 2008, pp. 1–4. DOI: 10.1109/ICPR.2008.4761913 *Cited on page 58.*

[135] Luis Roldão, Raoul de Charette, and Anne Verroust-Blondet. "3D Surface Reconstruction from Voxel-based Lidar Data". In: *arXiv preprint arXiv:1906.10515* (2019) *Cited on page 58.*

[136] Zimo Li, Prakruti C. Gogia, and Michael Kaess. "Dense Surface Reconstruction from Monocular Vision and LiDAR". In: *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*. 2019, pp. 6905–6911. DOI: 10.1109/ICRA.2019.8793729 *Cited on pages 58, 72.*

[137] Xavier Brun, Jean-Emmanuel Deschaud, and François Goulette. "On-the-way city mobile mapping using laser range scanner and fisheye camera". In: 2007 *Cited on pages 58, 59.*

[138] Matthew Carlberg, James Andrews, Peiran Gao, and Avideh Zakhor. "Fast surface reconstruction and segmentation with ground-based and airborne lidar range data". In: *UC Berkeley, Tech. Rep.* Citeseer. 2008 *Cited on pages 58, 59.*

[139] William E. Lorensen and Harvey E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm". In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1987, Anaheim, California, USA, July 27-31, 1987*. 1987, pp. 163–169. DOI: 10.1145/37401.37422 *Cited on page 59.*

[140] Bruno Vallet, Mathieu Brédif, Andrés Serna, Beatriz Marcotegui, and Nicolas Paparoditis. "TerraMobilita/iQmulus urban point cloud analysis benchmark". In: *Computers & Graphics* 49 (2015), pp. 126–133. DOI: 10.1016/j.cag.2015.03.004 *Cited on page 59.*

[141] Peter Liepa. "Filling Holes in Meshes". In: *First Eurographics Symposium on Geometry Processing, Aachen, Germany, June 23-25, 2003*. 2003, pp. 200–205. DOI: 10.2312/SGP/SGP03/200-206 *Cited on page 61.*

[142] Nicolas Paparoditis, Jean-Pierre Papelard, Bertrand Cannelle, Alexandre Devaux, Bahman Soheilian, Nicolas David, and Erwann Houzay. "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* 200.1 (2012), pp. 69–79 *Cited on pages 61, 71.*

[143] Peter Lindstrom and Greg Turk. "Fast and memory efficient polygonal simplification". In: *Visualization '98, Proceedings, October 18-23, 1998, Research Triangle Park, North Carolina, USA*. 1998, pp. 279–286. DOI: 10.1109/VISUAL.1998.745314 *Cited on page 62.*

[144] Peter Lindstrom and Greg Turk. "Evaluation of Memoryless Simplification". In: *IEEE Trans. Vis. Comput. Graph.* 5.2 (1999), pp. 98–115. DOI: 10.1109/2945.773803 *Cited on page 62.*

[145] Michael Waechter, Nils Moehrle, and Michael Goesele. "Let There Be Color! Large-Scale Texturing of 3D Reconstructions". English. In: *ECCV 2014*. Ed. by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars. Vol. 8693. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 836–850. ISBN: 978-3-319-10601-4. DOI: 10.1007/978-3-319-10602-1_54 *Cited on pages 62–64, 69, 70.*

[146] Yuri Boykov, Olga Veksler, and Ramin Zabih. "Fast Approximate Energy Minimization via Graph Cuts". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 23.11 (2001), pp. 1222–1239. DOI: 10.1109/34.969114 *Cited on page 64.*

[147] Fausto Bernardini, Joshua Mittleman, Holly E. Rushmeier, Cláudio T. Silva, and Gabriel Taubin. "The Ball-Pivoting Algorithm for Surface Reconstruction". In: *IEEE Trans. Vis. Comput. Graph.* 5.4 (1999), pp. 349–359. DOI: 10.1109/2945.817351 *Cited on pages 67, 68.*

[148] Yanping Fu, Qingan Yan, Long Yang, Jie Liao, and Chunxia Xiao. "Texture Mapping for 3D Reconstruction With RGB-D Sensor". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 4645–4653. DOI: 10.1109/CVPR.2018.00488 *Cited on page 69.*

[149] Bastian Goldlücke, Mathieu Aubry, Kalin Kolev, and Daniel Cremers. "A Super-Resolution Framework for High-Accuracy Multiview Reconstruction". In: *International Journal of Computer Vision* 106.2 (2014), pp. 172–191. DOI: 10.1007/s11263-013-0654-8 *Cited on page 71.*

References for Chapter 4: Supervised over-segmentation for 3D semantic mapping

[3] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. "Geometric Deep Learning: Going beyond Euclidean data". In: *IEEE Signal Process. Mag.* 34.4 (2017), pp. 18–42. DOI: 10.1109/MSP.2017.2693418 *Cited on pages 2, 87, 90.*

[14] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *CVPR, IEEE* 1.2 (2017) *Cited on pages 11, 74, 75, 81, 87, 91, 92, 96, 98, 103, 113, 128, 135.*

[15] Yin Zhou and Oncel Tuzel. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 4490–4499. DOI: 10.1109/CVPR.2018.00472 *Cited on pages 11, 74, 81, 87, 88, 91.*

[16] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. "MeshCNN: a network with an edge". In: *ACM Trans. Graph.* 38.4 (2019), 90:1–90:12. DOI: 10.1145/3306346.3322959 *Cited on pages 11, 74, 75, 81, 87, 95, 96, 98, 114, 117, 120, 127, 128, 132.*

[17] Loic Landrieu and Martin Simonovsky. "Large-Scale Point Cloud Semantic Segmentation With Superpoint Graphs". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 4558–4567. DOI: 10.1109/CVPR.2018.00479 *Cited on pages 11, 13, 42, 75, 93, 96, 106, 128, 129, 136.*

[18] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, Sabine Süsstrunk, et al. "SLIC superpixels compared to state-of-the-art superpixel methods". In: *IEEE transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012) *Cited on pages 11, 75, 77–80, 107.*

[21] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. "nuScenes: A multimodal dataset for autonomous driving". In: *CoRR* abs/1903.11027 (2019). arXiv: 1903.11027 *Cited on pages 11, 16, 26, 28, 81, 130.*

[23] Iro Armeni, Ozan Sener, Amir Roshan Zamir, Helen Jiang, Ioannis K. Brilakis, Martin Fischer, and Silvio Savarese. "3D Semantic Parsing of Large-Scale Indoor Spaces". In: *CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016. DOI: 10.1109/CVPR.2016.170 *Cited on pages 11, 106, 128, 129, 135, 136, 138, 139.*

[24] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. "ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 2432–2443. DOI: 10.1109/CVPR.2017.261 *Cited on pages 11, 16, 27, 49, 81.*

[26] Jingwei Huang, Haotian Zhang, Li Yi, Thomas A. Funkhouser, Matthias Nießner, and Leonidas J. Guibas. "TextureNet: Consistent Local Parametrizations for Learning From High-Resolution Signals on Meshes". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, pp. 4440–4449. DOI: 10.1109/CVPR.2019.00457 *Cited on pages 12, 87, 94.*

[38] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. 2009, pp. 248–255. DOI: 10.1109/CVPRW.2009.5206848 *Cited on pages 16, 82.*

[49] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*. 2012, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074 *Cited on pages 16, 23, 25, 26, 28, 43, 44, 106.*

[57] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. "Virtual worlds as proxy for multi-object tracking analysis". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4340–4349 *Cited on pages 23, 24, 119, 123.*

[150] Zhaowei Ma, Chang Wang, Yifeng Niu, Xiangke Wang, and Lincheng Shen. "A saliency-based reinforcement learning approach for a UAV to avoid flying obstacles". In: *Robotics and Autonomous Systems* 100 (2018), pp. 108–118 *Cited on page 74.*

[151] Ignacio Alzugaray and Alberto Sanfeliu. "Learning the hidden human knowledge of UAV pilots when navigating in a cluttered environment for improving path planning". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016*. 2016, pp. 1589–1594. DOI: 10.1109/IROS.2016.7759257 *Cited on page 74.*

[152] Ioannis Kostavelis and Antonios Gasteratos. "Semantic mapping for mobile robotics tasks: A survey". In: *Robotics and Autonomous Systems* 66 (2015), pp. 86–103. DOI: 10.1016/j.robot.2014.12.006 *Cited on page 74.*

[153] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *NIPS*. 2017 *Cited on pages 74, 87, 92, 113.*

[154] Daniel Maturana and Sebastian A. Scherer. "VoxNet: A 3D Convolutional Neural Network for real-time object recognition". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*. 2015, pp. 922–928. DOI: 10.1109/IROS.2015.7353481 *Cited on pages 74, 87.*

[155] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. "OctNet: Learning Deep 3D Representations at High Resolutions". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 6620–6629. DOI: 10.1109/CVPR.2017.701 *Cited on pages 74, 87, 88, 91.*

[156] Hayko Riemenschneider, András Bódis-Szomorú, Julien Weissenberg, and Luc Van Gool. "Learning Where to Classify in Multi-view Semantic Segmentation". In: *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*. 2014, pp. 516–532. DOI: 10.1007/978-3-319-10602-1_34 *Cited on pages 75, 116, 127.*

[157] Stéphane Guinard and Loic Landrieu. "Weakly supervised segmentation-aided classification of urban scenes from 3D LiDAR point clouds". In: *ISPRS Workshop*. 2017 *Cited on pages 75, 77, 79, 93, 108, 109.*

[158] Mohammad Rouhani, Florent Lafarge, and Pierre Alliez. "Semantic segmentation of 3D textured meshes for urban scene analysis". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 123 (2017), pp. 124–139 *Cited on pages 75, 81, 123.*

[159] Wei-Chih Tu, Ming-Yu Liu, Varun Jampani, Deqing Sun, Shao-Yi Chien, Ming-Hsuan Yang, and Jan Kautz. "Learning Superpixels With Segmentation-Aware Affinity Loss". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 568–576. DOI: 10.1109/CVPR.2018.00066 *Cited on pages 75, 79.*

[160] Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. "Superpixel Sampling Networks". In: *ECCV*. 2018. DOI: 10.1007/978-3-030-01234-2_22 *Cited on pages 75, 77, 79, 98, 109.*

[161] Xiaofeng Ren and Jitendra Malik. "Learning a Classification Model for Segmentation". In: *9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France*. 2003, pp. 10–17. DOI: 10.1109/ICCV.2003.1238308 *Cited on pages 76, 78.*

[162] Olga Veksler, Yuri Boykov, and Paria Mehrani. "Superpixels and Supervoxels in an Energy Optimization Framework". In: *ECCV*. 2010. DOI: 10.1007/978-3-642-15555-0_16 *Cited on pages 76–78.*

[163] David Stutz, Alexander Hermans, and Bastian Leibe. "Superpixels: An evaluation of the state-of-the-art". In: *Computer Vision and Image Understanding* 166 (2018). DOI: 10.1016/j.cviu.2017.03.007 *Cited on pages 76, 96, 99.*

[164] Yizhak Ben-Shabat, Tamar Avraham, Michael Lindenbaum, and Anath Fischer. "Graph based over-segmentation methods for 3D point clouds". In: *Computer Vision and Image Understanding* (2018). ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2018.06.004 *Cited on pages 76, 77, 79.*

[165] Yangbin Lin, Cheng Wang, Dawei Zhai, Wei Li, and Jonathan Li. "Toward better boundary preserved supervoxel segmentation for 3D point clouds". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 143 (2018). ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2018.05.004 *Cited on pages 76, 77, 80, 108, 109.*

[166] Ming-Yu Liu, Oncel Tuzel, Srikumar Ramalingam, and Rama Chellappa. "Entropy rate superpixel segmentation". In: *CVPR*. IEEE. 2011 *Cited on pages 77, 78, 108.*

[167] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan A. Essa. "Efficient hierarchical graph-based video segmentation". In: *CVPR*. 2010. DOI: 10.1109/CVPR.2010.5539893 *Cited on pages 77, 78.*

[168] David Weikersdorfer, David Gossow, and Michael Beetz. "Depth-adaptive superpixels". In: *Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012, Tsukuba, Japan, November 11-15, 2012*. 2012, pp. 2087–2090 *Cited on pages 77, 78.*

[169] Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Wörgötter. "Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds". In: *CVPR*. 2013. DOI: 10.1109/CVPR.2013.264 *Cited on pages 77, 79, 80, 108, 109.*

[170] Zizhao Wu, Ruyang Shou, Yunhai Wang, and Xinguo Liu. "Interactive shape co-segmentation via label propagation". In: *Computers & Graphics* 38 (2014), pp. 248–254 *Cited on pages 77, 80, 81.*

[171] Patricio D. Simari, Giulia Picciau, and Leila De Floriani. "Fast and Scalable Mesh Superfacets". In: *Comput. Graph. Forum* 33.7 (2014), pp. 181–190. DOI: 10.1111/cgf.12486 *Cited on pages 77, 80, 81, 121–126.*

[172] Giulia Picciau, Patricio D. Simari, Federico Iuricich, and Leila De Floriani. "Supertetras: A Superpixel Analog for Tetrahedral Mesh Oversegmentation". In: *Image Analysis and Processing - ICIAP 2015 - 18th International Conference, Genoa, Italy, September 7-11, 2015, Proceedings, Part I*. 2015, pp. 375–386. DOI: 10.1007/978-3-319-23231-7_34 *Cited on pages 77, 81.*

[173] Wei-Chih Tu1 Ming-Yu Liu, Varun Jampani2 Deqing Sun Shao-Yi, Chien1 Ming-Hsuan Yang, and Jan Kautz. "Learning superpixels with segmentation-aware affinity loss". In: *CVPR*. IEEE. 2018 *Cited on pages 77, 101, 109, 111.*

[174] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. "Efficient Graph-Based Image Segmentation". In: *International Journal of Computer Vision* 59.2 (2004). DOI: 10.1023/B:VISI.0000022288.19776.77 *Cited on page 78.*

[175] Yuri Boykov and Vladimir Kolmogorov. "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 26.9 (2004), pp. 1124–1137. DOI: 10.1109/TPAMI.2004.60 *Cited on pages 78, 116.*

[176] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. "A survey on deep learning techniques for image and video semantic segmentation". In: *Applied Soft Computing* 70 (2018), pp. 41–65 *Cited on page 79.*

[177] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, and Daniel Cremers. "Clustering with Deep Learning: Taxonomy and New Methods". In: *CoRR* abs/1801.07648 (2018). arXiv: 1801.07648 *Cited on page 79.*

[178] Loic Landrieu and Guillaume Obozinski. "Cut Pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs". In: *SIAM Journal on Imaging Sciences* 10.4 (2017) *Cited on pages 79, 85, 99, 107, 131.*

[179] S. Song, H. Lee, and S. Jo. "Boundary-enhanced supervoxel segmentation for sparse outdoor LiDAR data". In: *Electronics Letters* 50.25 (2014). ISSN: 0013-5194. DOI: 10.1049/el.2014.3249 *Cited on page 79.*

[180] L Shapira, A Shamir, and D Cohen-Or. *Consistent partitioning of meshes*. Tech. rep. Tech. rep., The interdisciplinary Center, 2005 *Cited on pages 80, 124.*

[181] Ron Kimmel. "Fast marching methods on triangulated domains". In: *Proc. Nat. Acad. Sci.* 95 (1998), pp. 8431–8435 *Cited on pages 80, 124.*

[182] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. "Variational shape approximation". In: *ACM Trans. Graph.* 23.3 (2004), pp. 905–914. DOI: 10.1145/1015706.1015817 *Cited on page 81.*

[183] Brian Kulis. "Metric Learning: A Survey". In: *Foundations and Trends in Machine Learning* 5.4 (2013), pp. 287–364. DOI: 10.1561/2200000019 *Cited on page 82.*

[184] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. "SphereFace: Deep Hypersphere Embedding for Face Recognition". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 6738–6746. DOI: 10.1109/CVPR.2017.713 *Cited on page 82.*

[185] Alfonso Medela, Artzai Picon, Cristina López Saratxaga, Oihana Belar, Virginia Cabezón, Riccardo Cicchi, Roberto Bilbao, and Ben Glover. "Few Shot Learning in Histopathological Images: Reducing the Need of Labeled Data on Biological Datasets". In: *16th IEEE International Symposium on Biomedical Imaging, ISBI 2019, Venice, Italy, April 8-11, 2019*. 2019, pp. 1860–1864. DOI: 10.1109/ISBI.2019.8759182 *Cited on page 82.*

[186] Sean Bell and Kavita Bala. "Learning visual similarity for product design with convolutional neural networks". In: *ACM Trans. Graph.* 34.4 (2015), 98:1–98:10. DOI: 10.1145/2766959 *Cited on page 82.*

[187] Pu Chen, Xinyi Xu, and Cheng Deng. "Deep View-Aware Metric Learning for Person Re-Identification". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. 2018, pp. 620–626. DOI: 10.24963/ijcai.2018/86 *Cited on page 82.*

[188] Sumit Chopra, Raia Hadsell, and Yann LeCun. "Learning a Similarity Metric Discriminatively, with Application to Face Verification". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*. 2005, pp. 539–546. DOI: 10.1109/CVPR.2005.202 *Cited on page 82.*

[189] Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. "Distance Metric Learning for Large Margin Nearest Neighbor Classification". In: *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*. 2005, pp. 1473–1480 *Cited on page 82.*

[190] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 2015, pp. 815–823. DOI: 10.1109/CVPR.2015.7298682 *Cited on page 83.*

[191] Kihyuk Sohn. "Improved Deep Metric Learning with Multi-class N-pair Loss Objective". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2016, pp. 1849–1857 *Cited on page 84.*

[192] Francis Engelmann, Theodora Kontogianni, Jonas Schult, and Bastian Leibe. "Know What Your Neighbors Do: 3D Semantic Segmentation of Point Clouds". In: *GMDL Workshop, ECCV*. 2018 *Cited on pages 84, 111, 113.*

[193] Ulrike von Luxburg. "A tutorial on spectral clustering". In: *Statistics and Computing* 17.4 (2007), pp. 395–416. DOI: 10.1007/s11222-007-9033-z *Cited on page 85.*

[194] Ariel Shamir. "A survey on Mesh Segmentation Techniques". In: *Comput. Graph. Forum* 27.6 (2008), pp. 1539–1556. DOI: 10.1111/j.1467-8659.2007.01103.x *Cited on pages 85, 86.*

[195] Hugo Raguet and Loic Landrieu. "Parallel Cut Pursuit For Minimization of the Graph Total Variation". In: *CoRR* abs/1905.02316 (2019). arXiv: 1905.02316 *Cited on pages 86, 131.*

[196] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015 *Cited on page 87.*

[197] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91 *Cited on page 87.*

[198] Yann LeCun. "Learning Invariant Feature Hierarchies". In: *Computer Vision - ECCV 2012. Workshops and Demonstrations - Florence, Italy, October 7-13, 2012, Proceedings, Part I*. 2012, pp. 496–505. DOI: 10.1007/978-3-642-33863-2_51 *Cited on page 87.*

[199] Eman Ahmed, Alexandre Saint, Abd El Rahman Shabayek, Kseniya Cherenkova, Rig Das, Gleb Gusev, Djamila Aouada, and Björn E. Ottersten. "Deep Learning Advances on Different 3D Data Representations: A Survey". In: *CoRR* abs/1808.01462 (2018). arXiv: 1808.01462 *Cited on pages 87, 96.*

[200] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. "Multi-view Convolutional Neural Networks for 3D Shape Recognition". In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. 2015, pp. 945–953. DOI: 10.1109/ICCV.2015.114 *Cited on pages 87, 89.*

[201] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. "3D Shape Segmentation with Projective Convolutional Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 6630–6639. DOI: 10.1109/CVPR.2017.702 *Cited on pages 87, 89.*

[202] Martin Simonovsky and Nikos Komodakis. "Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 29–38. DOI: 10.1109/CVPR.2017.11 *Cited on pages 87, 90, 93, 132.*

[203] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015 *Cited on page 89.*

[204] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. "Spectral Networks and Locally Connected Networks on Graphs". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014 *Cited on page 90.*

[205] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2016, pp. 3837–3845 *Cited on page 90.*

[206] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017 *Cited on page 90.*

[207] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. "The Graph Neural Network Model". In: *IEEE Trans. Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605 *Cited on page 90.*

[208] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. "Gated Graph Sequence Neural Networks". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. 2016 *Cited on page 90.*

[209] Alexandre Boulch, Joris Guerry, Bertrand Le Saux, and Nicolas Audebert. "SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks". In: *Computers & Graphics* 71 (2018), pp. 189–198. DOI: 10.1016/j.cag.2017.11.010 *Cited on page 91.*

[210] Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. "Geodesic Convolutional Neural Networks on Riemannian Manifolds". In: *2015 IEEE International Conference on Computer Vision Workshop, ICCV Workshops 2015, Santiago, Chile, December 7-13, 2015*. 2015, pp. 832–840. DOI: 10.1109/ICCVW.2015.112 *Cited on pages 93, 94.*

[211] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*. 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28 *Cited on pages 94, 114–116.*

[212] Anastasia Ioannidou, Elisavet Chatzilari, Spiros Nikolopoulos, and Ioannis Kompatsiaris. "Deep Learning Advances in Computer Vision with 3D Data: A Survey". In: *ACM Comput. Surv.* 50.2 (2017), 20:1–20:38. DOI: 10.1145/3042064 *Cited on page 96.*

[213] Loic Landrieu and Martin Simonovsky. "Large-scale point cloud semantic segmentation with superpoint graphs". In: *CVPR*. IEEE. 2018 *Cited on pages 97, 112, 113, 135.*

[214] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. "Learning fine-grained image similarity with deep ranking". In: *CVPR*. 2014 *Cited on pages 97, 100.*

[215] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2 (1991), pp. 251–257. DOI: 10.1016/0893-6080(91)90009-T *Cited on page 97.*

[216] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2014, Columbus, OH, USA, June 23-28, 2014*. 2014, pp. 512–519. DOI: 10.1109/CVPRW.2014.131 *Cited on page 98.*

[217] Sumit Chopra, Raia Hadsell, and Yann LeCun. "Learning a similarity metric discriminatively, with application to face verification". In: *CVPR*. Vol. 1. IEEE. 2005 *Cited on page 100.*

[218] Elad Hoffer and Nir Ailon. "Deep metric learning using triplet network". In: *International Workshop on Similarity-Based Pattern Recognition*. Springer. 2015 *Cited on page 100.*

[219] Peter J Huber et al. "Robust regression: asymptotics, conjectures and Monte Carlo". In: *The Annals of Statistics* 1.5 (1973) *Cited on page 100.*

[220] Pierre Charbonnier, Laure Blanc-Féraud, Gilles Aubert, and Michel Barlaud. "Deterministic edge-preserving regularization in computed imaging". In: *IEEE Transactions on Image Processing* 6.2 (1997) *Cited on page 100.*

[221] Tong Zhang et al. "Some sharp performance bounds for least squares regression with l1 regularization". In: *The Annals of Statistics* 37.5A (2009) *Cited on page 100.*

[222] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial transformer networks". In: *Advances in Neural Information Processing Systems*. 2015 *Cited on page 104.*

[223] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. "Exploring Spatial Context for 3D Semantic Segmentation of Point Clouds". In: *ICCV Workshops*. 2017. DOI: 10.1109/ICCVW.2017.90 *Cited on pages 106, 113, 119, 128.*

[224] Alex Levinshtein, Adrian Stere, Kiriakos N Kutulakos, David J Fleet, Sven J Dickinson, and Kaleem Siddiqi. "Turbopixels: Fast superpixels using geometric flows". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.12 (2009) *Cited on page 108.*

[225] Jerome Demantke, Clément Mallet, Nicolas David, and Bruno Vallet. "Dimensionality based scale selection in 3D lidar point clouds". In: *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci* 38.5 (2011), W12 *Cited on page 109.*

[226] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *CVPR*. 2016 *Cited on page 111.*

[227] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. "PointCNN". In: *arXiv preprint arXiv:1801.07791* (2018) *Cited on pages 113, 135.*

[228] Shenlong Wang, Simon Suo, Wei-Chiu Ma3 Andrei Pokrovsky, and Raquel Urtasun. "Deep parametric continuous convolutional neural networks". In: *CVPR*. 2018 *Cited on page 113.*

[229] Xiaoqing Ye, Jiamao Li, Hexiao Huang, Liang Du, and Xiaolin Zhang. "3D Recurrent Neural Networks with Context Fusion for Point Cloud Semantic Segmentation". In: *ECCV*. 2018. DOI: 10.1007/978-3-030-01234-2_25 *Cited on page 113.*

[230] Boris Delaunay et al. "Sur la sphere vide". In: *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* 7.793-800 (1934), pp. 1–2 *Cited on page 114.*

[231] Juan Pineda. "A parallel algorithm for polygon rasterization". In: *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1988, Atlanta, Georgia, USA, August 1-5, 1988*. Ed. by Richard J. Beach. ACM, 1988, pp. 17–20. DOI: 10.1145/54852.378457 *Cited on pages 118, 119.*

References for Chapter 5: Conclusion

[16] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. "MeshCNN: a network with an edge". In: *ACM Trans. Graph.* 38.4 (2019), 90:1–90:12. DOI: 10.1145/3306346.3322959 *Cited on pages 11, 74, 75, 81, 87, 95, 96, 98, 114, 117, 120, 127, 128, 132.*

[17] Loic Landrieu and Martin Simonovsky. "Large-Scale Point Cloud Semantic Segmentation With Superpoint Graphs". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 4558–4567. DOI: 10.1109/CVPR.2018.00479 *Cited on pages 11, 13, 42, 75, 93, 96, 106, 128, 129, 136.*

[20] Abhishek Patil, Srikanth Malla, Haiming Gang, and Yi-Ting Chen. "The H3D Dataset for Full-Surround 3D Multi-Object Detection and Tracking in Crowded Urban Scenes". In: *CoRR* abs/1903.01568 (2019). arXiv: 1903.01568 *Cited on pages 11, 16, 26–28, 130.*

[21] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. "nuScenes: A multimodal dataset for autonomous driving". In: *CoRR* abs/1903.11027 (2019). arXiv: 1903.11027 *Cited on pages 11, 16, 26, 28, 81, 130.*

[23] Iro Armeni, Ozan Sener, Amir Roshan Zamir, Helen Jiang, Ioannis K. Brilakis, Martin Fischer, and Silvio Savarese. "3D Semantic Parsing of Large-Scale Indoor Spaces". In: *CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016. DOI: 10.1109/CVPR.2016.170 *Cited on pages 11, 106, 128, 129, 135, 136, 138, 139.*

[58] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. "Exploring Spatial Context for 3D Semantic Segmentation of Point Clouds". In: *IEEE International Conference on Computer Vision, 3DRMS Workshop, ICCV*. 2017 *Cited on pages 23, 24, 129, 140, 141.*

[79] Qimin Cheng, Qian Zhang, Peng Fu, Conghuan Tu, and Sen Li. "A survey and analysis on automatic image annotation". In: *Pattern Recognition* 79 (2018), pp. 242–259. DOI: 10.1016/j.patcog.2018.02.017 *Cited on pages 39, 130.*

[178] Loic Landrieu and Guillaume Obozinski. "Cut Pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs". In: *SIAM Journal on Imaging Sciences* 10.4 (2017) *Cited on pages 79, 85, 99, 107, 131.*

[195] Hugo Raguet and Loic Landrieu. "Parallel Cut Pursuit For Minimization of the Graph Total Variation". In: *CoRR* abs/1905.02316 (2019). arXiv: 1905.02316 *Cited on pages 86, 131.*

[202] Martin Simonovsky and Nikos Komodakis. "Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 29–38. DOI: 10.1109/CVPR.2017.11 *Cited on pages 87, 90, 93, 132.*

[232] Laurent Caraffa, Mathieu Brédif, and Bruno Vallet. "3D Watertight Mesh Generation with Uncertainties from Ubiquitous Data". In: *Computer Vision - ACCV 2016 - 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part IV*. 2016, pp. 377–391. DOI: 10.1007/978-3-319-54190-7_23 *Cited on page 131.*

[233] Fisher Yu and Vladlen Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. 2016 *Cited on page 132.*

[234] Gaetan Bahl, Lionel Daniel, Matthieu Moretti, and Florent Lafarge. "Low-Power Neural Networks for Semantic Segmentation of Satellite Images". In: *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*. 2019, pp. 2469–2476. DOI: 10.1109/ICCVW.2019.00302 *Cited on page 132.*

[235] Piotr Mirowski, Matthew Koichi Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, Koray Kavukcuoglu, Andrew Zisserman, and Raia Hadsell. "Learning to Navigate in Cities Without a Map". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 2424–2435 *Cited on page 132.*

References for Chapter 5.4: Appendix

[14] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *CVPR, IEEE* 1.2 (2017) *Cited on pages 11, 74, 75, 81, 87, 91, 92, 96, 98, 103, 113, 128, 135.*

[17] Loic Landrieu and Martin Simonovsky. "Large-Scale Point Cloud Semantic Segmentation With Superpoint Graphs". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 4558–4567. DOI: 10.1109/CVPR.2018.00479 *Cited on pages 11, 13, 42, 75, 93, 96, 106, 128, 129, 136.*

[23] Iro Armeni, Ozan Sener, Amir Roshan Zamir, Helen Jiang, Ioannis K. Brilakis, Martin Fischer, and Silvio Savarese. "3D Semantic Parsing of Large-Scale Indoor Spaces". In: *CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016. DOI: 10.1109/CVPR.2016.170 *Cited on pages 11, 106, 128, 129, 135, 136, 138, 139.*

[58] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. "Exploring Spatial Context for 3D Semantic Segmentation of Point Clouds". In: *IEEE International Conference on Computer Vision, 3DRMS Workshop, ICCV*. 2017 *Cited on pages 23, 24, 129, 140, 141.*

[213] Loic Landrieu and Martin Simonovsky. "Large-scale point cloud semantic segmentation with superpoint graphs". In: *CVPR*. IEEE. 2018 *Cited on pages 97, 112, 113, 135.*

[227] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. "PointCNN". In: *arXiv preprint arXiv:1801.07791* (2018) *Cited on pages 113, 135.*

[236] Lyne P Tchapmi, Christopher B Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. "SEGCloud: Semantic Segmentation of 3D Point Clouds". In: *International Conference on 3D Vision* (2017) *Cited on page 135.*

[237] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. "Exploring Spatial Context for 3D Semantic Segmentation of Point Clouds". In: *ICCV, 3DRMS Workshop*. 2017 *Cited on page 135.*

[238] Francis Engelmann, Theodora Kontogianni, Jonas Schult, and Bastian Leibe. "Know What Your Neighbors Do: 3D Semantic Segmentation of Point Clouds". In: *arXiv preprint arXiv:1810.01151* (2018) *Cited on page 135.*