3D Scene Reconstruction and Completion for Autonomous Driving

Luis Guillermo Roldão Jimenez

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

THÈSE DE DOCTORAT
DE SORBONNE UNIVERSITÉ

Préparée à l'Université Pierre et Marie Curie

3D Scene Reconstruction and Completion for Autonomous Driving

Reconstruction et Complétion 3D de la Scène pour la Conduite Autonome

Soutenue par

Luis Guillermo ROLDÃO JIMENEZ

Le 05 juillet 2021

École doctorale n°130

École doctorale Informatique, Télécommunications et Électronique (Paris)

Spécialité

Informatique

Composition du jury :

Bruno VALLET DR., IGN	*Rapporteur*
Paul CHECCHIN Prof., Université Clermont Auvergne	*Rapporteur*
Jean-Emmanuel DESCHAUD CR., MINES ParisTech	*Examinateur*
Jürgen GALL Prof., University of Bonn	*Examinateur*
Adrian HILTON Prof., University of Surrey	*Examinateur*
Anne VERROUST-BLONDET CR., Inria	*Directrice de thèse*
Raoul DE CHARETTE CR., Inria	*Encadrant*
Cyril OUAZINE AKKA Research	*Invité*

UPMC SORBONNE UNIVERSITÉS | SORBONNE UNIVERSITÉ

Contents

Chapter 1

Introduction

In the last decades, research and development of autonomous vehicles has increased considerably. Nevertheless, human drivers are still required to remain behind the wheel for supervision and take over in case of failures or unexpected conditions, evidencing that complete vehicle automation is still far from accomplished. One of the main challenges to overcome lies in the perception systems in charge of sensing and understanding the vehicle surroundings. This is done by employing a suite of sensors gathering data of the outside world, further processed by computer vision algorithms that perform different tasks (Janai et al., 2020) (i.e. free space estimation (Fan et al., 2020) or obstacle detection (Lang et al., 2019)). Therefore, reliability of these algorithms plays a vital role and failure cases can lead to fatal accidents. As consequence, it is crucial for autonomous vehicles to make sense of the complex and dynamic driving scenarios encountered to operate safely. Furthermore, the ability to create a geometrical 3D model of the local surroundings is key to understand the structure of the scene for precise navigation and to deal with unexpected scenarios. This is difficult given 3D sensor limitations providing sparse and unevenly scanned data. Therefore, different classical (Oleynikova et al., 2017; Newcombe et al., 2011; Kwon et al., 2019) and learning-based (Yuan et al., 2018; Song et al., 2017; Groueix et al., 2018; Liao et al., 2018) reconstruction and completion techniques have been presented in recent years to compute a comprehensive 3D model of the scene and provide high level information useful for trajectory planning and decision making.

In the remainder of this chapter, we introduce the context that encompasses the motivation of this thesis in Section 1.1 and provide an overview of 3D computer vision in Section 1.2. Finally, the organization outline of this document can be found in Section 1.3.

1.1 Thesis context

This thesis has been carried out at the Robotics and Intelligent Transportation Systems (RITS) team from Inria Paris and funded by the research department of AKKA Technologies. The RITS team is a multidisciplinary robotics team specialized in autonomous driving, covering different research topics ranging from perception, decision making, vehicle communications and control systems. AKKA Research carries out R&D activities to develop

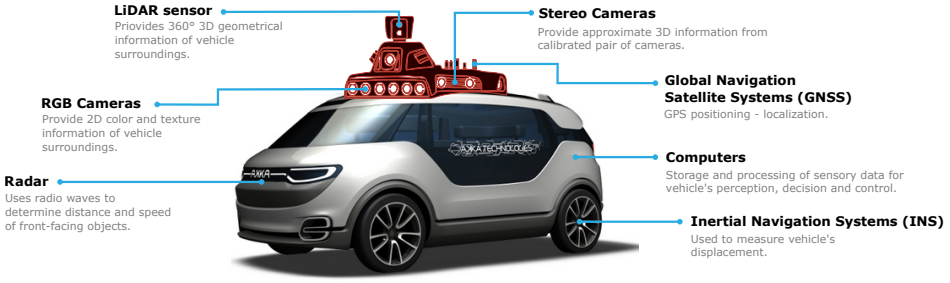


Figure 1.1: Traditional sensor setup for autonomous driving applications.

technological bricks that can be applied in an industrial context for mobility solutions.

The main goal of this PhD thesis is to study and develop algorithms for 3D scene reconstruction and completion to provide enhanced environment perception for autonomous driving. For this, we rely on 3D point clouds which can be obtained from different sets of sensors. Figure 1.1 presents a common setup of a self-driving vehicle. **RGB** and **stereo** cameras both provide dense *visual* cues of color and texture information, stereo can additionally provide approximate 3D *geometry* cues from disparity maps. **LiDAR** sensors, on the other hand, provide very accurate sparse 3D *geometry* information acquired from laser beams to calculate distance to objects through time-of-flight principle. Sophisticated LiDAR configurations are capable to generate complete 360° scans and are commonly employed in autonomous vehicle setups. Additional range sensors such as **Radar** or **ultrasounds** are commonly used for specific applications (i.e. parking maneuvers or front vehicle speed detection) as they provide more limited information in terms of accuracy or resolution. Given the inherent 3D nature of LiDAR data, its higher accuracy and large amount of information provided, we focus on its use for all the applications presented in this thesis although our algorithms could be applied to 3D point clouds obtained from any other sensor.

Finally, **global navigation satellite systems** and **inertial navigation systems** are commonly employed for high precision localization and to track vehicle displacement. **Embedded – or remote – computers** interpret acquired information and take decisions translated into instructions for the vehicle actuators.

1.2 3D Vision

In recent years, 3D computer vision has gained major momentum thanks to the rapid development and affordability of 3D scanning technologies and the constant increase of computational capabilities. Meanwhile, a large number of datasets have been released to facilitate research in different domains

(Firman, 2016; Janai et al., 2020), including virtual and augmented reality (VR/AR), geology, medical imaging, computer graphics and robotics.

Since autonomous vehicles evolve in a 3D world, accurate and rich 3D geometry information is a crucial requirement for reliable navigation and precise localization in complex dynamic scenarios. Despite maturity of 2D computer vision algorithms and impressive performance of deep learning techniques, images cannot provide precise geometrical depth information of a scene. Even though recent learning-based methods perform depth estimation from a single monocular image (Fu et al., 2018; Garg et al., 2019), their accuracy is significantly below those of 3D LiDAR (Bhoi, 2019). Moreover, 2D images are illumination sensitive in contrast to LiDAR which actively scan the scene and overcome such limitation.

Traditionally, 3D computer vision techniques relied on primitive extraction (Fischler and Bolles, 1981; Rabbani et al., 2006), geometrical priors (Sung et al., 2015; Remil et al., 2017; Martens et al., 2017) and the use of handcrafted features with simple classifiers (Weinmann et al., 2013; Lalonde et al., 2006) for reconstruction, classification and segmentation tasks. However, these methods are usually designed for a given application and commonly present poor generalization. In recent years, the 3D vision community has been actively investigating extension of deep learning to 3D data. As a result, we have witnessed their adaptation to different data representations, including multiview-based methods using 2D CNNs (Su et al., 2015), voxel-based 3D CNNs (Maturana and Scherer, 2015; Riegler et al., 2017b) and point-based networks (Qi et al., 2017a,b).

In this thesis we follow the same pattern. In Part I, we explore classical 3D vision techniques relying on geometrical priors for 3D *reconstruction* of the scene. Moreover, we explore applications of LiDAR point clouds for autonomous vehicles that can be described in two aspects: (a) temporal aggregation of 3D scans for accurate 3D occupancy maps estimation (cf. Chapter 2); and (b), surface reconstruction from a single 3D scan (cf. Chapter 3). These works were performed during a time where the field was transitioning towards 3D deep learning. In Part II, we make the same transition and propose an in-depth and critical analysis of the novel literature on semantic scene completion (cf. Chapter 4). Finally, we propose a lightweight method to semantically complete unseen and occluded regions of entire 3D scenes (cf. Chapter 5).

1.2.1 3D Scanning technologies

Different technologies exist for the acquisition of 3D point clouds of a given scene. We now briefly describe the most popular ones.

Laser scanners. Laser range sensors perform active sensing of the environment by the use of time-of-flight principle. Light Detection and Ranging

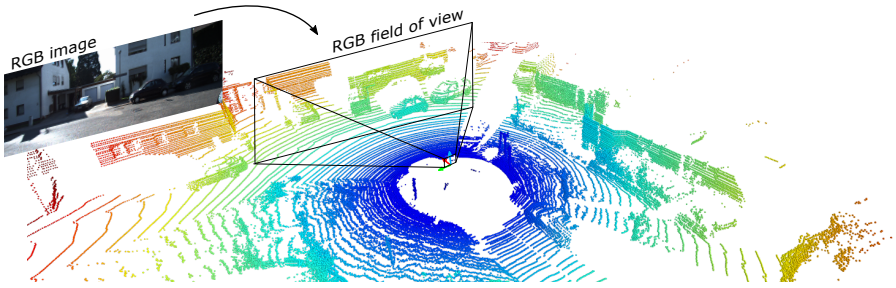


Figure 1.2: 3D point cloud acquired from Velodyne HDL-64 laser scanner. Frustum represents field of view of an RGB camera positioned close to the LiDAR sensor, RGB image shown in top left for reference. Color maps distance from sensor. Note density loss at far regions (red points) as opposed to close areas (blue points). Data from KITTI dataset (Geiger et al., 2013).

(LiDAR) is the most popular technology, which measures distance traveled by emitted pulsed light waves. These sensors provide the most accurate 3D information and are highly robust to the variation of lighting conditions, making them ideal for day and night operation. While the recent flash LiDARs have no moving parts, most of them still use rotating mirrors to direct laser beams on multiple directions and create 360° scans of the scene. LiDARs are the most common 3D sensors for autonomous driving applications, providing accuracy of about ±2cm and measuring distances up to hundreds of meters from the sensor. An example of a 3D LiDAR point cloud is shown in Figure 1.2.

Depth cameras. These cameras provide depth information for every pixel within a 2D image and are sometimes registered along with an RGB sensor, thus providing RGB-D data. Many types of depth cameras exist with different acquisition technologies (i.e. time-of-flight cameras and structured-light cameras). However, these technologies commonly struggle in outdoor scenarios and lack precision when compared to laser scanners although they are considerably cheaper. We refer to dedicated surveys for in-depth analysis of scene reconstruction from Depth or RGB-D sensors (Zollhöfer et al., 2018; Malleson et al., 2019).

Photogrammetry and stereoscopy. Rather than a scanning technology, these are common techniques relying in processing algorithms to obtain the 3D information from two or more monocular images of a same scene sensed from different viewpoints, either through multi-view stereo or Structure-from-Motion (SfM) techniques. Similarly, we refer to dedicated surveys on SfM (Özyesil et al., 2017) and stereo vision (Sunyoto et al., 2004; Laga et al., 2020).

(a) Original shape (b) Non-uniform sampling (c) Noise and outliers (d) Missing data

Figure 1.3: Point cloud artifacts obtained during the scanning process shown in case of a 2D curve for simplicity. Adapted from: Berger et al. (2017)

1.2.2 3D Data representations

Different 3D representations exist in the literature, we now biefly present the most popular ones. We additionally present the artifacts commonly generated from the scanning process when generating raw 3D pointclouds and that we tackle with the scene reconstruction (Part I) and semantic completion (Part II) algorithms presented in this thesis.

1.2.2.1 3D Point clouds

3D point clouds (see Figure 1.4a) are the most popular and common 3D representation. It consists in an unordered set of 3D points representing a scanned scene with their respective coordinates. More formally, a point cloud is as a set of 3D points $\{\mathcal{P}_i \mid i = 1, \dots, n\}$, where each point $\mathcal{P}_i$ is a vector of its (x, y, z) spatial coordinates which can additionally contain extra feature channels such as reflection intensity, color, normal, etc.

In recent years, the constant evolution of LiDAR sensors enables to obtain rich and accurate 3D geometric information. However, the size of the input data along with sensing artifacts (Berger et al., 2017) still present important challenges for 3D reconstruction and scene understanding algorithms. These artifacts come from sensor limitations as they provide *uncertain* measurements that depend not only on distance from sensed object, but also on external conditions such as temperature of the environment, lighting conditions or physical properties of impacted materials (Dong and Chen, 2017). We now describe most important artifacts and explain how they relate to the scene reconstruction and scene completion problems. Although we present these artifacts as relative to 3D point clouds, they also affect subsequently presented representations as they commonly derive from the latter.

Non-uniform sampling. It refers to the distribution of points sampling the surface. LiDAR sensors produce scans with uneven distribution given the vertical and horizontal angular resolutions of the sensor, where density of measurements is high at close ranges and decays rapidly with the distance (refer to Figure 1.2). This produces ambiguities since some objects are partially scanned. Uneven density represents a challenge for surface recon-

struction algorithms that need to adapt *a priori* assumptions (i.e. density, noise level) to the heterogeneous density scans.

Noise and outliers. Despite their high accuracy, LiDAR sensors are not exempt of noisy measurements and outliers. Noise is commonly introduced along the line of sight of the sensor and can result from surface properties and sensor inaccuracies. Outliers in the other hand are commonly due to structural artifacts in the acquisition process. 3D reconstruction and scene understanding algorithms must be robust against these artifacts.

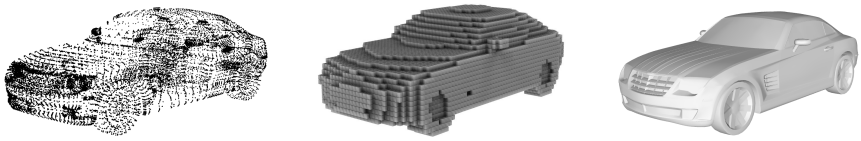
Missing data. Point clouds are sparse due to limited sensor range, non-reflective surfaces, heterogeneous sampling and occlusions as observed in Figure 1.2. For better scene understanding it is necessary to complete such missing regions. Therefore, a wide variety of scene reconstruction and completion algorithms exist in the literature to recover the missing information which are deeply studied along this thesis.

1.2.2.2 Voxel grids

3D voxel grids are a discrete volumetric representation, where each cubic cell –known as *voxel*– within the 3D grid represents a defined volumetric region of the space. Although voxel grids are commonly used to store occupancy information derived from the point cloud (aka occupancy grids) as observed in Figure 1.4b, other information can be stored such as point density, normal values or gradient field distances. The representation is useful to bring structure to the previously unstructured set of points, being convenient for many algorithms, including neighbor search or 3D CNNs (Ahmed et al., 2018). However, computational and memory requirements are high since they grow cubically with the grid resolution. This can be tackled by using hierarchical representations (detailed in Chapter 2). Given the structured data provided by voxel grids and the ease of implementation, we use this data representation along the complete thesis for the different methods presented.

1.2.2.3 Surface meshes

3D meshes enable a continuous representation of the scanned surface by a set of polygons. This is commonly done by the use of 3D vertices connected to form triangular faces representing an approximated 2D manifold embedded in 3D space (see Figure 1.4c). The surface can be obtained from a wide variety of approaches that can be distinguished between explicit and implicit (Berger et al., 2017). The continuity of this representation can be of high interest for physical modeling or detail terrain traversability applications. We will extend on this representation in Chapter 3 where we present a



(a) Point cloud (b) Voxel grid (c) Surface Mesh

Figure 1.4: Different 3D data representations. Each representation exhibits different structural and geometrical properties, making its choice application-oriented. Model from ShapeNet dataset (Chang et al., 2015).

voxel-based surface reconstruction method from heterogeneous density point clouds.

1.3 Thesis structure

This thesis is organized in two parts. Part I explores traditional methods useful for **3D scene reconstruction** which we define as the process of recovering the 3D model of a scene given one or multiple scans. We particularly study the use of discrete occupancy grids and continuous surface meshes.

In Chapter 2 we tackle aggregation of multiple measurements in an occupancy grid representation (Roldão et al., 2018) which can be seen as intermediate step for scene reconstruction. Different from the literature, we consider ray-path observations to resolve ambiguities in partially occupied cells and consider the density of observations to weight occupancy updates. Experiments show that our method can reduce occupancy state inaccuracies in partially occupied cells. Our method is of interest for localization and creation of grid based HD maps (Kiran et al., 2018).

In Chapter 3, we shift to a continuous surface reconstruction method from a single 3D scan (Roldão et al., 2019). To accommodate the LiDAR heterogeneous density we propose an adaptive neighborhood strategy to perform local approximations of the surface at different levels. The latter are used to compute a truncated signed distance field. The final surface is obtained by applying a meshification algorithm over the gradient field. Our method is able to deal with sparse density scans, achieving an accurate reconstruction of the local surroundings and completing missing regions in small areas by interpolation.

Despite the potential of 3D scene reconstruction algorithms to complete small regions of missing data through interpolation, completion of large areas generated by occlusions or limited field-of-view proves more challenging and can only be tackled by learning priors with data-driven methods. Moreover, 3D scene reconstruction only comprises geometric understanding of the scene without considering the semantic meaning of surrounding objects which is

key to leverage interaction with the real world. Therefore, in Part II of the thesis we switch to deep learning methods to perform **3D semantic scene completion (SSC)** which jointly performs scene completion and semantic segmentation of entire 3D scenes.

In Chapter 4 we provide an in-depth and critical analysis of the literature which was motivated by little consensus and large number of recently published approaches (Roldão et al., 2021). We hence study the most popular datasets for the task, the wide variety of architectures employed, and performance achieved in different scenarios.

Furthermore, in Chapter 5 we propose our lightweight multiscale semantic completion network – LMSCNet –, which performs multiscale semantic completion at different resolutions enabling faster inference times at the coarsest subdivision (Roldão et al., 2020). We achieve a lightweight approach by employing a 2D backbone architecture that encodes one of the spatial dimensions as a feature dimension. 3D segmentation heads at multiple scales are in charge of retrieving the 3D semantically completed scene at different resolutions. Our method achieved state-of-the-art performance at the time of submission on the popular SemanticKITTI dataset (Behley et al., 2019), while requiring less computation and memory resources.

Finally, chapter 6 summarizes our contributions and gives an outlook on future works.

Part I

3D Scene Reconstruction

In the first part of this thesis we use traditional computer vision techniques useful for 3D reconstruction of the scene by using geometrical and physical priors. This is done either for multi-frame aggregation through the use of occupancy grids or single frame environment perception by using surface reconstruction algorithms.

Chapter 2

Statistical Update of Occupancy Grid Maps

The contributions of this chapter were made public in an Arxiv research report (Roldão et al., 2018) and a workshop conference paper (Kiran et al., 2018):

Roldão, L., de Charette, R., and Verroust-Blondet, A. (2018). A statistical update of grid representations from range sensors. *ArXiv 2018*.

Kiran, B. R., Roldão, L., Irastorza, B., Verastegui, R., Süss, S., Yogamani, S., Talpaert, V., Lepoutre, A., and Trehard, G. (2018). Real-time dynamic object detection for autonomous driving using prior 3D-maps. In *ECCV Workshop 2018*.

Our aim in this chapter is to explore the use of 3D occupancy grid maps for multi-frame environment mapping applications.

Note that this chapter is quite different from the rest as it concentrates on aggregation from multiple scans without applying interpolation or completion techniques to complete missing data. Although this is not strictly considered as *reconstruction*, it can be studied as a sub-part of the reconstruction problem.

Contents

2.1 Introduction

Occupancy grids, introduced by Moravec and Elfes (1985), present a discrete and compact representation of the environment for mobile robots. This is done by *tessellation* of the space into fixed-size cells that describe a probabilistic estimate of its occupancy (Thrun et al., 2005). Different from continuous point-based representations, occupancy grids are immediately usable to navigate within dynamic environments, since they segment the space into free, occupied and unknown cells.

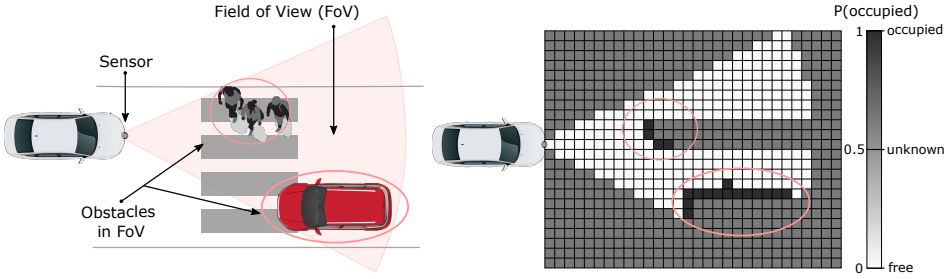
Occupancy grids were initially implemented in 2D to generate a projected map of the environment (Moravec and Elfes, 1985). The representation was further extended in Bares et al. (1989) and Hebert et al. (1989), with additional height values assigned to each cell to represent non-flat surfaces (2.5D). Multiple height values can also be stored (Triebel et al., 2006) to model vertical structures (i.e. underpasses). In the last decade, thanks to increasing computational capabilities, three-dimensional grids have been proposed (Hornung et al., 2013) and are considered for the application presented in this chapter. Furthermore, we focus on the update of occupancy grids from multiple LiDAR scans by considering Bayesian probability (Elfes, 1989) which accounts for inaccuracies present in the sensor measurements and vehicle pose estimations (refer to Section 2.2.3).

In the literature, it is common to update the complete state of a grid cell from a single measurement at a given instant. However, range sensors provide only a partial observation as beams only traverse a fraction of the cell. This is specially problematic for partially occupied cells since contradictory observations might be acquired depending on where the cell is traversed. To reduce inconsistencies, we propose an inverse sensor model that considers the ray path information within cells. Experimental evaluations are performed on both synthetic and real data by using the CARLA simulator (Dosovitskiy et al., 2017) and the KITTI dataset (Geiger et al., 2013), respectively. We compare our results with the popular 3D OctoMap (Wurm et al., 2010; Hornung et al., 2013). Our contributions can be summarized as follows:

- We propose a statistical Inverse Sensor Model (ISM) which considers the complete set of measurements acquired from range sensors and accounts for both the ray-path information and the density of observations.

- Our approach can be directly applied to Bayesian fusion of occupancy grids in both linear and log-odds notation for optimized applications.

- In both synthetic and real data, our method outperforms the OctoMap baseline providing a more detailed update of the occupancy maps and reducing inaccuracies.



(a) Real 3D scene with sensor FoV (b) Corresponding 2D occupancy grid

Figure 2.1: 2D projected occupancy grid (b), obtained from a real 3D scene (a). Colormap represents probability of occupancy according to sensor measurements. Unknown areas correspond to cells outside FoV or occluded.

2.2 Related work

The occupancy grid serves as an updatable environment model of the world, where the occupied space covered by different objects is estimated through a perception model evaluated for each measurement provided by range sensors. This model needs to account for sensor limitations, uncertainties and errors as described in Section 1.2.2.1. Once the model has been created, the grid is useful to determine free and occupied space as observed in Figure 2.1. Furthermore, the grid needs to be updated from multiple observations acquired form the sensors at different times and positions.

As explained in Dia (2020), three main steps are presented in the literature for probabilistic occupancy update of the cells in the grid map by exploiting sensor measurements. We highlight an additional step which considers the effect of long-term sensor fusion on grid-based maps:

1. A **sensor model** is used to account for measurement uncertainty. This is done by modeling the uncertainty associated to sensor outputs in different scenarios to approximate a mathematical model for sensor measurements, and it varies according to the type of sensor employed (Section 2.2.1).

2. The **inverse sensor model** represents the conditional occupancy probability of each cell given the measurements. For this, it is commonly assumed that each cell is represented by an independent binary random variable (Moravec, 1988) (Section 2.2.2).

3. Fusion of sensor measurements is performed by using **Bayesian fusion** (Bayes, 1763). This is used to (a) fuse observations coming from either a single or various different sensors, or (b) fuse observations from different instants given a known sensor pose (Section 2.2.3).

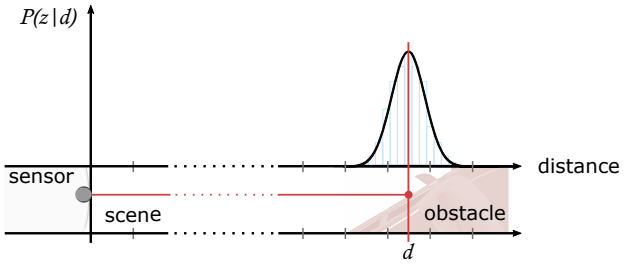


Figure 2.2: Model for a nearest-target range sensor. The Probability Density Function $P(z|d)$ represents uncertainty by a normal distribution centered at impacted distance (d), the standard deviation models the sensor precision.

4. Different **update policies** can be considered to tackle occupancy state overconfidence of a cell updated from many observations (Yguel et al., 2007), especially in unbounded domains such as the log-odd space shown in Equation 2.5. This is important to enable occupancy updates of changing cells in dynamic scenarios (Section 2.2.4).

Before presenting our method in Section 2.3, we provide formal definitions and taxonomy that will be used along this chapter. Some of our definitions are based on the recent theses of Rakotovao (2017) and Dia (2020).

2.2.1 Sensor model

The **sensor model** translates the uncertainty of measurements into a probabilistic distribution which provides the likelihood of getting a specific sensor measurement. This depends on the sensor employed, its physical configuration, limitations and precision. More formally, let z be a measurement obtained from a range sensor at distance d. The sensor model is represented by the **Probability Density Function (PDF)** given by $P(z|d)$ which depends on the distance $f(d)$. If we assume a perfect sensor with a nearest-target behavior, i.e. a single return is caused by nearest obstacle inside FoV (e.g. LiDAR), the PDF can be expressed as:

$$P(z|d) = \begin{cases} 1 & \text{if } z = f(d), \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

Given the absence of perfect sensors, a probability distribution can be obtained by repeating the measurement process several times to obtain the precision of the sensor (see Figure 2.2). Additional parameters can be added to the model such as the distance from the sensor which should impact precision (Konolige, 1997). For instance, an analytical sensor model that considers the distance traveled by the ray through a decay rate is proposed in Schaefer et al. (2017). Furthermore, Bennewitz et al. (2009) considers the

angle of incidence between the sensor and impacted surface to account for erroneous reading generated by poor-reflecting surfaces.

2.2.2 Inverse sensor model

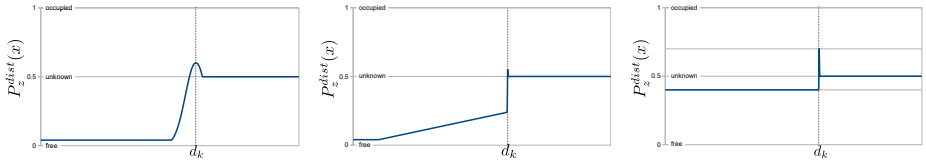
Consider a grid of defined dimensions denoted as $\mathcal{G}$, containing N cells (c) such as $\mathcal{G} = \{c_i\}, i = 1, \ldots, N$, where c_i represents the i-th cell within $\mathcal{G}$. Consider also k measurements returned from one or many sensors at known poses defined as $z_1, \ldots, z_k$. The occupancy grid, which we refer to as $\mathcal{O}$ represents a function that maps these measurements $z_1, \ldots, z_k$ into the set of occupancy probabilities of all cells c_i within $\mathcal{G}$. More formally:

$$\mathcal{O}\{z_1, \ldots, z_k\} = \{P(c_i|z_1 \wedge \ldots \wedge z_k), \forall c_i \in \mathcal{G}\}, \tag{2.2}$$

where $P(c_i|z_1 \wedge \ldots \wedge z_k) = 1 - P(\neg c_i|z_1 \wedge \ldots \wedge z_k) \in [0, 1]$ represents the occupancy probability of the i-th cell according to the set of measurements.

The **Inverse Sensor Model (ISM)** is a function which enables to compute the occupancy probability of each cell given a single measurement returned by the sensor. This function considers the sensor model defined in Section 2.2.1 and translates it into the occupancy probability of each cell concerned within the measurement. Notice that a single measurement not only provides information for the impacted cell at distance d but also for all cells that have been traversed within the ray path. In the literature, it is commonly assumed that all cells traversed by a range observation can be considered as free while the impacted cell can be considered as occupied. Additionally, all cells lying behind the impacted cell remain uncertain. Furthermore, uncertainties accounted by the sensor model can be integrated within this consideration. However, we highlight that for range sensors occupancy state updates of an entire cell are based on partial observations as each ray covers an infinitesimal portion of the cell.

The first ISM was proposed in (Elfes, 1989) by considering Bayesian probability. We refer to (Dia, 2020) for a complete mathematical derivation of the model. The method accounts for all possible grid configurations (2^N) and evaluates all possible locations of the obstacle given by the measurement and its uncertainty. However, the main limitation of such approach lies in its exponential complexity, hindering practical implementations if real-time performance is required, specially for three-dimensional grids. An extension of the method with linear complexity was introduced in Pathak et al. (2007) by assuming all cell states as conditionally independent given a sensor observation. This induces conflicts that may lead to inconsistent maps, even for noise-free sensors. More recently, the techniques proposed by Rakotovao (2017) and Kaufman et al. (2016) relax this assumption and maintain linear complexity for nearest-target behavior sensors.



(a) Gaussian (Homm et al., 2010) (b) Piece-wise linear (Weiss et al., 2007) (c) Three-valued (Wurm et al., 2010)

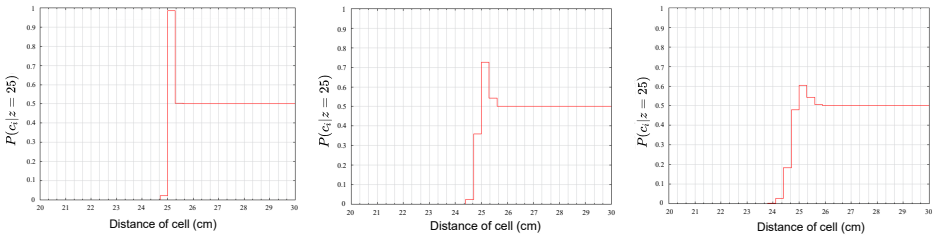
Figure 2.3: Examples of analytical ISMs to approximate Bayesian ISM (Elfes, 1989). All models consider a probability distribution based on a range measurement obtained at distance d_k. The complexity of the model directly impacts computation time of cell probabilities. Source: Rakotovao et al. (2015).

Other methods present analytic approaches to directly represent the inverse sensor model by a continuous function or a discrete approximation of the Bayesian method (refer to Figure 2.3) as follows:

$$P(c_i|z) \approx P_z^{dist}(d_i), \tag{2.3}$$

swhere $P_z^{dist}(d_i)$ is a continuous function defined over distance d_i from the sensor. This enables to directly evaluate the inverse sensor model for a given cell without considering all grid configurations, leading to a $O(1)$ complexity and allowing real-time implementations (Dia, 2020). For instance, some works propose to model the ISM by a Gaussian distribution if the sensor can be modeled by the same distribution as shown in Figure 2.3a (Payeur et al., 1998; Gartshore et al., 2002; Homm et al., 2010; Einhorn et al., 2011; Adarve et al., 2012). This has been done for both stereo cameras (Li and Ruichek, 2013; Nguyen et al., 2012) and laser scanners (Weiss et al., 2007). Another option consists in obtaining a piece-wise linear approximation which approaches the Gaussian distribution as seen in Fig 2.3b (Weiss et al., 2007). Additionally, a simpler representation is presented in Wurm et al. (2010) and Hornung et al. (2013) by considering only three possible values for the ISM, reflecting empty, occupied and unknown regions respectively and enabling fast computation (as shown in Figure 2.3c).

Notice that the complexity of the ISM directly impacts computation efficiency. The occupancy grid presents two main parameters: its spatial dimensions and its resolution. While the first represents the amount of space covered by the grid, the second refers to the number of cells within such space. Consider a particular sensor with precision σ, if the size of the cells within the grid – denoted as ω – is considerably larger than the sensor precision (– low grid resolution –), formally $\omega \gg \sigma$, simpler sensor models can be preferred to save computation, as they approach to three-valued



(a) $\sigma = 0.1\ cm$ (b) $\sigma = 0.2\ cm$ (c) $\sigma = 0.3\ cm$

Figure 2.4: Relationship between the grid resolution and the precision of the sensor for the ISM. Figures show analytical Gaussian approximation ISM (Figure 2.3a) for a range return at 25 cm. All cases consider cell size $\omega = 0.3\ cm$ and variable sensor precision σ. Notice that the complexity of the model decreases with the sensor precision. Furthermore if $\omega \gg \sigma$, the implementation of the ISM resembles a three-valued model (Wurm et al., 2010) shown in Figure 2.3c. Source: Dia (2020).

model presented in (Wurm et al., 2010). This can be easily appreciated in Figure 2.4. A study of the adequate resolution of an occupancy grid according to the sensor precision has been performed in Dia et al. (2017).

Given the ease of implementation and computation speed provided by analytical inverse sensor models, we advocate for these methods through our contributions presented in this chapter. However, despite the wide variety of ISMs proposed, we highlight that none of them considers the length traversed by the ray observations within the grid cells, which should directly impact occupancy probability assignation. To the best of our knowledge, we are the firsts to propose a method based in such observation. Our proposed ISM is described in Section 2.3.

2.2.3 Bayesian fusion

Once the inverse sensor model is defined, an additional model is needed to enable the fusion of measurements from multiple sources and from different moments when performing the incremental update of the occupancy grid. For this, sensor poses for all measurements at every instant need to be known. Thereof, we assume that we have a reasonably precise pose of the robotic platform (namely, the autonomous vehicle) along this chapter. In practice, poses can be estimated by integrating ego-vehicle motion from inertial navigation systems fused with GNSS sensors (Leonard et al., 2009) or applying registration techniques commonly used for Simultaneous localization and Mapping (aka SLAM) (Bresson et al., 2017).

In the literature, the update of occupancy grids from several measurements is often estimated by Bayesian methods (Elfes, 1989; Berger, 1988).

Different sensor measurements are usually assumed as independent to simplify the calculations. Even when this is not strictly accurate, it simplifies considerably the mapping algorithm without significantly increasing the error. To create the map, each cell's occupancy probability is recursively updated by performing a ray-casting operation that tracks each ray from the sensor origin to the impact point. Commonly, cells that have been traversed by a ray —*misses*— are considered as free, while cells where an impact occurs —*hits*— are considered as occupied.

In this line of work, most approaches insert the different sensor measurements acquired over time by applying a static state binary Bayes filter as introduced in Elfes (1989), where the occupancy probability of a cell c is defined by the following recursive equation:

$$P(c|z_{1:t}) = \left[1 + \frac{1 - P(c|z_t)}{P(c|z_t)} \frac{1 - P(c|z_{1:t-1})}{P(c|z_{1:t-1})} \frac{P(c)}{1 - P(c)}\right]^{-1}, \tag{2.4}$$

where $z_{1:t-1}$ corresponds to all previous measurements acquired; z_t expresses the current observation at time t, and $P(c)$ represents the prior knowledge about the map. Values are assigned to $P(c|z_t)$ as a function of the ISM considered (cf. Section 2.2.2), which specifies the probability of a cell c being occupied based on a single sensor measurement z_t. We refer to Thrun et al. (2005) for the entire derivation of this equation. In the literature, it is commonly assumed a uniform prior probability, meaning $P(c) = 0.5$. By making this assumption and applying a symmetric inverse sensor model – i.e. free and occupied observations have the same weight on the ISM – the probability update can be simplified to counting traversals and hits of sensor measurements as proposed in Kelly et al. (2006). To output a binary representation (i.e. occupied or free), probabilities are thresholded at 0.5, representing everything below as free and the rest as occupied.

Equation 2.4 is commonly implemented in logarithmic form by using the *log-odds* notation $l(x)$, defined as $l(x) = log\left(\frac{P(x)}{1-P(x)}\right)$, leading to:

$$l(c|z_{1:t}) = l(c|z_t) + l(c|z_{1:t-1}). \tag{2.5}$$

The log-odds notation is computationally advantageous since additions are computed faster than multiplications and also avoids numerical instabilities when probabilities are close to 0 and 1. Note that in case of precomputed sensor models, logarithms do not have to be calculated during the update step.

Differently, a forward sensor model proposed in Thrun (2003) refuses the sensor measurements independence hypothesis commonly assumed by Bayesian methods. This is done by finding the grid configuration that better explains best causes of a complete set of measurements, commonly by applying Expectation Maximization (EM) algorithms (Dempster et al., 1977).

2.2.4 Update policies

Update policies are commonly considered after studying the effect of long term measurements fusion on the occupancy grid (Yguel et al., 2007). When updates are performed by using solely the Bayesian update defined in Equation 2.4, the occupancy grid representation suffers from two major drawbacks: (1) it is overconfident, and (2) the occupancy state of adjacent cells is highly irregular – adjacent cells present heterogeneous probability values even if they show the same occupancy state –. This becomes evident when considering updates in the log-ratio space as defined in Equation 2.5, which leads to unbounded sums along the fusion updates.

1. The **overconfidence** can be easily observed by considering a symmetric sensor model. In such case, the number of observations required to change the occupancy state of a cell – i.e. from free to occupied –, will be equivalent to the number of past measurements that have defined this state. This is specially harmful in dynamic environments where occupancy states of cells can rapidly change.

2. On the other hand, **irregularity** in occupancy probabilities of adjacent cells is caused by the unbounded nature of Equation 2.5. In particular, as the viewpoint of the sensor observations change, heterogeneous occupancy probabilities along neighboring cells are obtained, preventing to implement map compression techniques through the use of hierarchical structures (cf. Section 2.2.5).

In order to solve these problems, different update policies have been proposed in the literature:

Max policy. Proposed in Payeur et al. (1997), it considers the maximum occupancy probability between the current and the previous measurement: $l(c|z_{1:t}) = \max(l(c|z_t), l(c|z_{1:t-1}))$. However, this is inappropriate for dynamic scenarios since occupied cells can never be updated as free.

Clamping policy. It restricts the unbounded probabilities in the log-ratio space $l(c|z_{1:t})$ by using clamping values l_{min} and l_{max} (Kraetzschmar et al., 2004). This technique avoids overconfident cells, as clamping probabilities enable changing occupancy state without the previously needed equivalent number of observations. Furthermore, it deals with grid irregularity as non-conflictive cells – cells without contradictory observations – commonly reach clamping values. This is convenient for representation compression as neighboring cells can be grouped into a single larger cell if they are depicted by same occupancy probability.

$$l(c|z_{1:t}) = \max(\min(l(c|z_t) + l(c|z_{1:t-1}), l_{max}), l_{min}). \tag{2.6}$$

Exponential forgetting policy. Presented in Yguel et al. (2007), it receives its name from the consideration that a past observation is exponentially less important for the current occupancy state estimation. It is defined by $l(c|z_{1:t}) = l(c|z_{1:t-1})(1-\zeta)+l(c|z_t)\zeta$, where ζ is a hyper-parameter which determines the forgetting ratio. The policy also enables to deal with overconfidence and irregularity but can make occupancy of occluded cells rapidly converge to an unknown state.

2.2.5 Hierarchical data structures

By definition, a discrete representation inhibits a fine-grained reconstruction. Discretization inaccuracies can be reduced by using a smaller cell size (Milstein, 2008; Dia et al., 2017), but this increases the computation and memory needs, particularly for 3D representations where the storage usage increases exponentially with the resolution of the map (smaller cell size).

Recursive structures such as quadtrees (Kraetzschmar et al., 2004) and octrees (Payeur et al., 1997, 1998; Hornung et al., 2013) are commonly used to face this problem, leading to a $\mathcal{O}(\log N)$ complexity for insertions and queries. They allow to make recursive subdivisions of the space until reaching a defined leaf cell of minimum size. The advantage of such structures is that groups of cells with the same state can be grouped and represented by a parent cell, being more computationally efficient.

2.3 Method

In this chapter we introduce an inverse sensor model based on the ray path information and sensor density. We start by signaling potential problems and flaws present in current methods. Our two main contributions are presented in Sections 2.3.2 and 2.3.1 and the resulting ISM is shown in Section 2.3.3. Finally, experimental results of our method and comparison with State of the Art approach are presented in Section 2.4.

For other methods in the literature, it is commonly considered that within a single scan, the state of each cell is binary (i.e. –free– or –occupied– respectively). Occupancy updates are commonly done by performing a *ray-casting* operation that tracks each ray from the sensor origin to the impact coordinates as observed in Figure 2.5. Hence, a cell is set as occupied if an impact occurred within, and free if it has been traversed by any ray. Furthermore, for high-density range sensors, it is common to group all measurements within a single time-frame for a unique update of the occupancy grid state (Wurm et al., 2010), or group measurements with same patterns into a single one to optimize the map updating process (Kwon et al., 2019). The problem of such approach is that the complete state of a cell is updated from a single partial observation, neglecting the contribution of multiple measurements and their validity as explained in Figure 2.5.

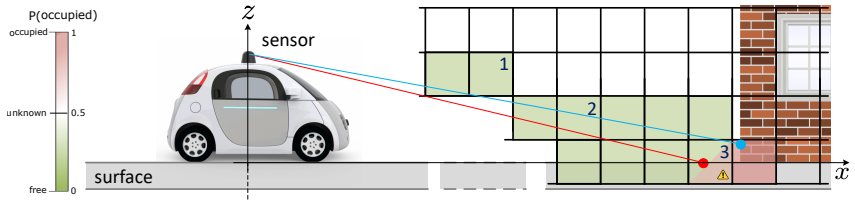


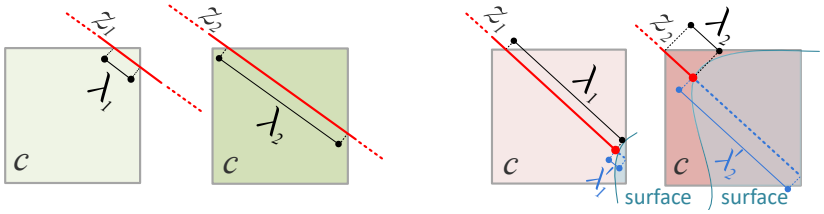
Figure 2.5: Occupancy grid frameworks generally consider that the complete state of a cell within a scan can be updated from a single measurement. Under this assumption, a slightly traversed cell will have the same occupancy probability as a cell traversed by many rays (cf. compare cells 1 and 2 in the figure). Furthermore, partially occupied cells as 3 will have same probability than a completely occupied cell if only the impact measurement is considered.

To account for this problem, we propose to update the occupancy probability of each cell by considering the ray path information (cf. Section 2.3.1), and the density of observations that can be obtained at such cell (cf. Section 2.3.2), which depends on its geometric position within the grid and distance from the sensor, respectively. Our method exploits ray path information to resolve ambiguities in partially occupied cells.

2.3.1 Occupancy probability from traversability

Different to the literature, we propose to consider the ray path information (i.e. the distance traveled by the ray) within each voxel in order to weight its occupancy probability. Thus our method models the fact that rays are only partial observations and that the information completeness depends on how all rays traverse each cell. A close idea has been presented in Schaefer et al. (2017) through the use of a decay rate that reduces the measurements confidence with the accumulated distance traveled by a ray within the cells. In our case, we consider individual voxel-based distances to modulate the occupancy probability.

As in Wurm et al. (2010) we define the fixed P_{free} and P_{occ} observation probabilities that are below and above 0.5 respectively (see Figure 2.3c), and extend their three-valued ISM by linearly weighting such probabilities according to the ray path information within the cell. Note that our choice of the three-valued model (see Figure 2.3c) is appropriate for autonomous driving applications as the precision of sensors σ is generally an order of magnitude smaller than commonly used voxel sizes ω (i.e. $\sigma = 2\ cm$ for a Velodyne HDL64 LiDAR vs. common 20 cm size voxels). We also tested the Gaussian approximation ISM (Homm et al., 2010) (see Figure 2.3a) without obtaining any significant improvement.



(a) Ray traversals – misses – (b) Ray impacts – hits –

Figure 2.6: The occupancy probability of every cell $P(c|z)$ is modulated according to the traversed distance λ of the measured ray z within the cell for both misses (a) and hits (b) observations. Higher traversed distance results in a lower occupancy probability, this is $\lambda_1 > \lambda_2 \implies P(c|z_1) < P(c|z_2)$.

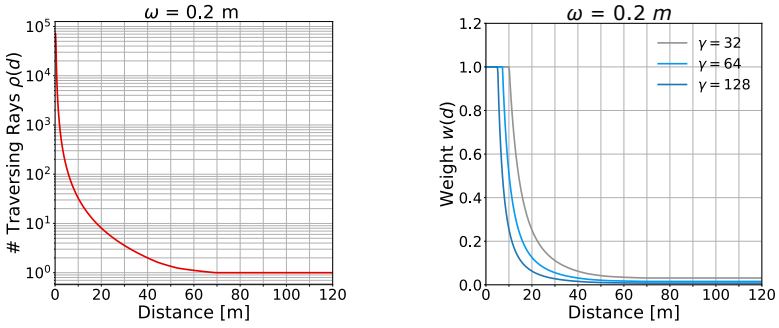
Our intuition is that the occupancy probability decreases with respect to the distance that a ray has traversed within a particular cell λ (see Figure 2.6). For the traversed cells we modulate this probability by comparing λ with the diagonal length of the cell ($\sqrt{3}\omega$), which corresponds to the maximum distance that a ray can traverse (see Figure 2.6a). In the case of impacted cells, the comparison is done with the length that the ray would have traversed within the cell if it had not encountered any obstacle ($\lambda + \lambda'$) (see Figure 2.6b), which corresponds to the maximum possible traversed distance given the current observation.

2.3.2 Weight measurement probability

Given the uneven distribution of sensory data from LiDAR scans (see Figure 1.2), closer cells will have a considerably higher number of observations than farther ones. This could lead to erroneous occupancy updates of cells away from the sensor based on a few uncertain observations. To account for this, we propose to compute the density of observations $\rho(d)$ that the sensor can measure from a cell at distance d and to weight the occupancy update of each cell according to $\rho(d)$.

More specifically, $\rho(d)$ models the approximate density of rays that might traverse a voxel of particular size ω at a given distance d, depending on the vertical and horizontal angular resolutions of the sensor φ_s and θ_s, respectively. This function has been estimated through an approximated analytical model and experimentally validated by comparing the result against synthetic data. We refer to Appendix A for details and complete derivation of the modeling function.

Our weighting function $w(d)$ is then obtained by scaling $\rho(d)$ according to γ and clamping it to 1. γ is a hyper-parameter that will affect the drop-off of the function and the distance at which the measurements will start



(a) Modeling function $\rho(d)$ (b) Weighting function $w(d)$

Figure 2.7: (a) An example of the modeling function $\rho(d)$ at $\omega = 0.2\ m$ voxel size. (b) Example of weight function $w(d)$ with different γ set at a voxel size $\omega = 0.2\ m$. For both cases we set $\theta_s = 0.16°$ and $\varphi_s = 0.4°$.

being weighted, as it can be seen on Figure 2.7b. A larger value results in a faster decay. Note that $w(d)$ shows an exponential behavior similar to $\rho(d)$. Altogether, the weighting function writes:

$$w(d) = \min\left(1, \frac{\rho(d)}{\gamma}\right). \tag{2.7}$$

2.3.3 Occupancy update

For the incremental update of our occupancy grid map we use Bayesian update (Elfes, 1989) as in Equation 2.4 and integrate our proposed inverse sensor model, where the contribution of each measurement z_k in a single scan at time t is assigned as defined in the following equation:

$$\frac{P(c|z_t^k)}{1 - P(c|z_t^k)} = \begin{cases} 0.5 + \left[\dfrac{(P_{\text{occ}} - 0.5)\,\lambda_i'}{\lambda_i + \lambda_i'}\right] w(d) & \text{for a hit,} \\ 0.5 - \left[\dfrac{(0.5 - P_{\text{free}})\lambda_i}{\sqrt{3}\omega}\right] w(d) & \text{for a miss.} \end{cases} \tag{2.8}$$

We account for all the information obtained from the complete set of measurements in a single scan (i.e. all rays traversing and/or impacting the voxel), weighting all observations according to the rays traversed distance λ and the sensor density $w(d)$.

Furthermore, we implement our occupancy updates in the log-odds form explained in Equation 2.5 for optimization and use the clamping policy from Kraetzschmar et al. (2004) shown in Equation 2.6 to bound probabilities and enable rapid changes of occupancy state of cells without needing an

equivalent number of observations (cf. Section 2.2.4). This last feature is important to enable updates in dynamic environments as is the case in autonomous driving scenarios (i.e. moving vehicles, pedestrians).

2.4 Experiments

We implement our method in C++ and use the OctoMap library (Hornung et al., 2013) which employs an optimized octree structure for real-time implementation and memory-computation efficiency. Our method is evaluated in both real and simulated data from the KITTI dataset (Geiger et al., 2013) and the CARLA simulator[1] (Dosovitskiy et al., 2017), respectively. We carry out all experiments with voxel size $\omega = 0.2m$, which represents the best trade-off between computation time and accuracy. We evaluate three different variants of our method and compare them with OctoMap (Hornung et al., 2013), which we use as baseline:

1. For our first variant, we simply apply our inverse sensor model defined in Equation 2.8, we denote this method as **ours_w_both**.

2. For the second variant, denoted as **ours_w_misses**, we only apply weighting function $w(d)$ to misses observations, meaning $w(d) = 1$ for hit measurements. This is similar to prioritizing hits as commonly found in the literature (Hornung et al., 2013; Schaefer et al., 2017).

3. Finally, a third variant is presented by ignoring $w(d)$ for both hits and misses observations, named as **ours_w_none**. This is done to evaluate the benefit of our weight function (Section 2.3.2).

Note that all variants of our method consider the complete set of measurements in a single scan (i.e. all rays traversing and/or impacting a voxel), which is different from OctoMap (Hornung et al., 2013) that considers a single observation per voxel and prioritizes hit observations over misses.

Synthetic data experimental setup (CARLA). For quantitative evaluation, we generate a voxelized ground truth by using a pre-defined number of sequential high density raw point clouds with known poses. We simulate a very high resolution LiDAR (364 layers) and used all rays to annotate free and occupied voxels in the scene – voxels that intersects the environment are annotated as occupied, while non-intersecting voxels are annotated as free –. We use CARLA, as the simulated LiDAR does not contain any measurement errors and we can obtain accurate sensor poses for frame-to-frame registration. Analogously, we use 64 layer LiDAR scans that overlaps with the voxelized ground truth for evaluation.

[1]For experiments on the CARLA simulator, we replicate the KITTI dataset sensor setup (`http://www.cvlibs.net/datasets/kitti/setup.php`).

Real data experimental setup (KITTI). Given the presence of sensor noise and registration inaccuracies for the sensor poses in the real data from KITTI, this dataset is only used for qualitative evaluation. We use one of the residential subsets of the KITTI dataset (Geiger et al., 2013), acquired from a roof mounted HDL-64E LiDAR sensor. Provided GPS-RTK data was used for frame-to-frame registration and three different sub-sequences of 60 frames without dynamic objects within the scenes were used.

2.4.1 Metrics

For quantitative evaluation we consider the **mean Average Precision (mAP)**, which enables to consider the classification confidence of the model. Classifiers with less confidence in miss-classified objects will score higher that those with high confidence.

Additionally we use a set of histograms to evaluate the confidence of predictions for all models, representing the proportion of correctly and incorrectly classified cells.

2.4.2 Performance evaluation

We now present quantitative results obtained on CARLA simulator (Section 2.4.2.1) and qualitative results on the KITTI dataset (Section 2.4.2.2). In all experiments, occupancy probabilities of all voxels are initialized with a uniform prior $P(c_i) = 0.5$ for all methods.

2.4.2.1 Quantitative results on simulated data (CARLA)

Evaluation is performed in a scene obtained after the fusion of 200 LiDAR scans acquired from different poses and that intersect the CARLA voxelized ground truth. Ground truth poses provided by the simulator are used for registration. We evaluate occupancy state of all voxels in the final scene after the fusion of all frames.

Hyper-parameter research. Initial experiments are performed to evaluate our sensor model hyper-parameters P_{free} and P_{occ}. Clamping thresholds of $P_{min} = 0.12$ and $P_{max} = 0.97$ were set to bound probability values domain. For our weighting function $w(d)$ we use $\gamma = 32$, while vertical and horizontal angular resolutions ϕ_s and θ_s are set to 0.4° and 0.16° respectively, corresponding to elevation and azimuth values of simulated LiDAR sensor.

Results are shown in Table 2.1, where each column corresponds to a set of hyper-parameters tested. It can be observed that the best performance corresponds to the variant ours_w_misses of our method (0.95 mAP for best hyper-parameters performance vs. 0.83 mAP for the OctoMap baseline). Results can be explained by the larger asymmetry between misses and

Method	P_{occ} / P_{free}	0.70† / 0.30	0.70 / 0.40	0.80† / 0.20	0.80 / 0.30	0.6 / 0.45	0.55† / 0.45
OctoMap (Hornung et al., 2013)		0.76	0.79	0.83	0.80	0.76	0.66
ours_w_both		0.70	0.73	0.71	0.74	0.71	0.64
ours_w_misses		**0.91**	**0.92**	**0.93**	**0.95**	**0.85**	**0.72**
ours_w_none		0.79	0.78	0.83	0.86	0.72	0.63

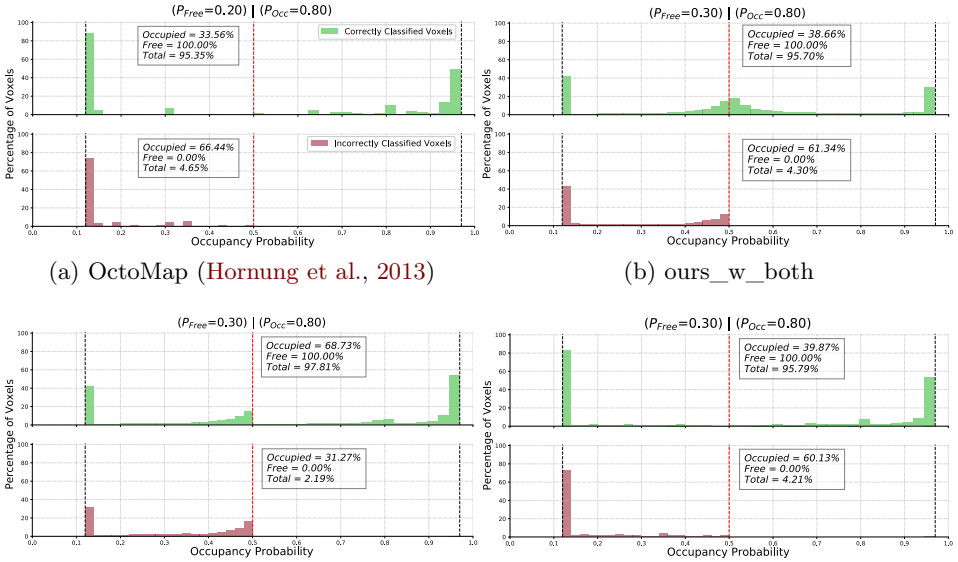
Table 2.1: Mean Average Precision (mAP) for all methods. Columns marked by † represent symmetric sensor models (i.e. P_{occ} and P_{free} are equidistant from initial prior probability $P(c_i) = 0.5$). Best hyper-parameters for each method are shown in blue.

hits observations enforced (i.e. no weight $w(d)$ applied for hits, giving them higher priority, specially at larger distances). This is a common configuration in the literature (Hornung et al., 2013; Payeur et al., 1997; Yguel et al., 2007; Kraetzschmar et al., 2004). However, one of the drawbacks of applying $w(d)$ to misses observations only is that a larger number of measurements will be needed to change the occupancy state of dynamic voxels from occupied to free, especially at larger distances. Finally, slight performance improvement of ours_w_none over OctoMap baseline (+3%) advocates for the interest of our ray path information proposal. This can be explained by the fact that partially occupied cells that have been traversed in a particular measurement will carry less impact in occupancy state changes when compared to OctoMap.

From now on, we will use the best sensor model hyper-parameters found for each method and shown in blue in Table 2.1.

Models confidence. Figure 2.8 presents a set of histograms with the confidence for correctly (top) and incorrectly (bottom) classified voxels. Classifications close to the 0.5 prior probability represents lower confidence as opposed to those at the right and left edges. A good model should be confident for correct classifications and should keep low confidence for incorrect ones. We highlight that there is a big proportion imbalance between free and occupied voxels in the ground truth scenes[2]. Again, ours_w_misses (Figure 2.8c) shows the best behavior since it correctly classifies around 69% of all occupied voxels. Furthermore, ours_w_misses reduces overconfidence, which is advantageous as explained in Section 2.2.4. Interestingly, correctly classified occupied voxels (rightmost in each top histogram) show high confidence, which is a desired behavior since false positives for occupied cells can be less tolerated for autonomous driving applications. OctoMap (Figure 2.8a) presents the worst proportion of correctly classified voxels (34%).

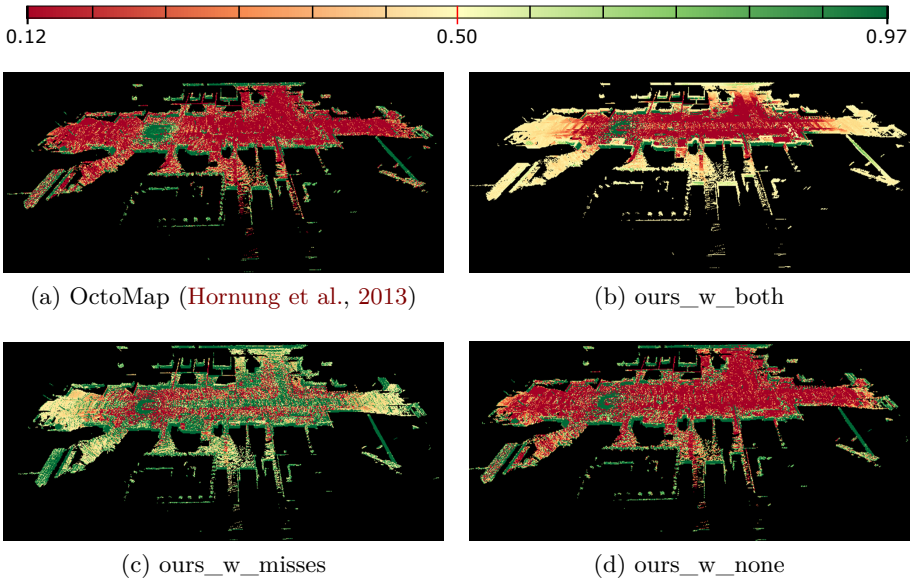
[2]free to occupied proportion is around 1:15 in a voxelized outdoor scene of dimensions $51.2 \times 51.2 \times 6.4$ m with voxel size 0.2 m



(a) OctoMap (Hornung et al., 2013) (b) ours_w_both

(c) ours_w_misses (d) ours_w_none

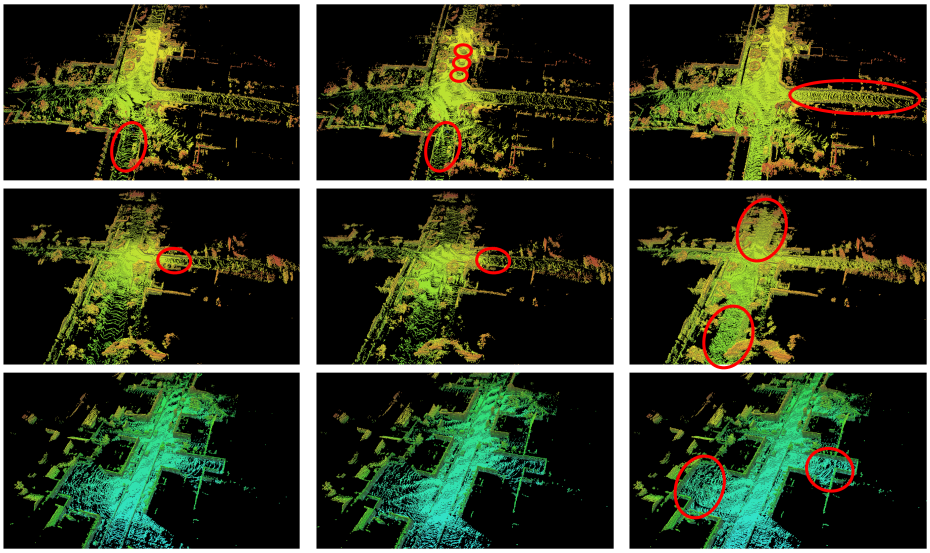
Figure 2.8: Proportion of correctly and incorrectly classified voxels and their respective occupancy probabilities for each method. Occupancy probability value can be interpreted as the confidence on the classification.



(a) OctoMap (Hornung et al., 2013) (b) ours_w_both

(c) ours_w_misses (d) ours_w_none

Figure 2.9: Correctly (greenish) and incorrectly (reddish) classified voxels and their respective occupancy probabilities. Colormap for probability values on top, borders represent clamping values $P_{min} = 0.12$ and $P_{max} = 0.97$ used.



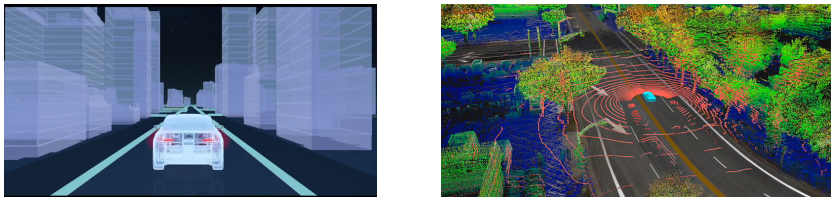
(a) OctoMap (Hornung et al., 2013) (b) ours_w_both (c) ours_w_misses

Figure 2.10: Qualitative results on the KITTI dataset (Geiger et al., 2013) on 3 non-overlapping sections of the same sequence. Highlighted regions show the better performance of our method with higher density of correctly predicted occupied voxels in the ground area.

Occupancy probabilities analysis. We show qualitative classification results in Figure 2.9. A perfect prediction would consist in a complete greenish scene. We show only the occupied voxels in the ground truth, meaning that probabilities for all voxels should lie above 0.5 for a correct classification. Voxels with probabilities close to 0.12 represent highly confident missclassified voxels as their occupancy probability is considerably lower than 0.5 – occupancy threshold –. Again, best performance of ours_w_misses method can be observed (Figure 2.9c) with the largest proportion of correctly predicted voxels (greenish) and less overconfidence for incorrectly predicted ones (reddish). Interestingly, misclassification commonly occurs on ground voxels. This shows that measurements are highly sensitive to partially occupied voxels that lie in a shallow angle with the sensor.

2.4.2.2 Qualitative results on real data (KITTI)

We evaluate all methods qualitatively on the KITTI dataset (Geiger et al., 2013). In this case we do not consider ours_w_none method for our comparison as it neglects one of our contributions. We present the results in Figure 2.10, again only occupied voxels are shown. An ideal reconstruction would consist in a dense representation with almost no free voxels in the



(a) HD-Map (b) Point cloud map matching

Figure 2.11: Occupancy grids can be used for the creation of 3D maps employed for mapping and localization. The figure shows (a) a high level visualization of an HD-map, and (b) the map matching process from a single LiDAR scan (shown ad red points).

area corresponding to the ground. Similar to previous results, note the high proportion of occupied voxels missclassified in the ground region. Results for ours_w_misses consistently show the best reconstruction among the compared methods as noticeable in red highlighted areas. This is due to the weighting function $w(d)$ applied to misses observations only which reduces the influence of rays traversing cells that are far from the sensor, leveraging the errors caused by the discretization.

2.5 Applications to autonomous driving

For autonomous driving applications it is common to register prior 3D maps that provide a static environment model used for localization and dynamic object detection (see Figure 2.11 for examples). These maps improve the robustness and performance of automated vehicles and are commonly created by using LiDAR sensors combined with GPS and inertial navigation systems (Levinson et al., 2011). Dynamic agents as vehicles and pedestrians can then be detected by background extraction from the pre-computed 3D map.

While point clouds from LiDAR provide an accurate geometrical representation, they do not explicitly provide information about unknown areas or free space and they are also memory-intensive and lack an inherent mechanism to adapt to changes in the environment. Thereof, grid based representations as our presented method are exploited to map the static environment that can be used for localization purposes by map matching. In consequence, precision of the occupancy update model is highly important. This is a common pipeline for autonomous vehicles operating in closed loop routes and pre-recorded paths such as shuttles or taxi services. Nowadays, these 3D maps not only provide geometrical but also semantic and contextual information such as speed limits, road space, lane-markings and distance to intersections.

In our work (Kiran et al., 2018), we review the use of 3D maps for autonomous driving and how they can be applied to dynamic object detection through background subtraction.

2.6 Conclusion

We have proposed a framework to statistically integrate range sensor measurements into a probabilistic occupancy grid through Bayesian fusion (Elfes, 1989). Contrary to other methods found in the literature, our method considers the ray path information by accounting for traversed distance of all rays within grid cell and models the possible density of sensor measurements according to the distance to weight occupancy updates.

Despite the improvements achieved by our method, discretization errors still occur which highlights that further work is needed on inverse sensor models to account for dynamic objects and conflictive cells in three-dimensional occupancy grids. Grid resolution plays a vital role on the accuracy of the reconstruction. While smaller voxel size would be preferred, memory and computation requirements inhibits this practice.

Further research is needed for more efficient hierarchical data structures to achieve multi-resolution representations. The idea of adapting cell size on-the-fly as presented in Einhorn et al. (2011) seems an interesting direction to explore.

Since it is not possible to achieve a continuous model with voxel-based representations and there exists limitations for considerably higher resolutions, surface representations might be preferred. This led us to the study of continuous surface methods that would enable to precisely represent the local surroundings without discretization and that will be presented in next chapter.

Chapter 3

Voxel-based Surface Reconstruction from LiDAR Data

The contributions of this chapter were published in Roldão et al. (2019):

Roldão, L., de Charette, R., and Verroust-Blondet, A. (2019). 3D surface reconstruction from voxel-based LiDAR data. In *ITSC 2019*.

1-minute demo video: `https://youtu.be/iyKShCBAW9g`

Contents

3.1 Introduction

For the following chapters of this thesis we concentrate on single-frame environment perception by exploiting a single LiDAR point cloud. Furthermore, we focus on recovering the fine-grained geometry of the scene from sparse 3D information acquired from the range sensor by computing a continuous surface of the local environment around the vehicle. While studied occupancy grids enable to model the surroundings, discretization leads to inaccuracies that require large memory and computation needs to be solved. Conversely, surface reconstruction outputs a representation that evolves in a continuous space, which might be useful for applications such as detail terrain traversability or physical modeling.

In our scope, a surface can be defined as a 2D manifold embedded in three-dimensional space $\mathbb{R}^3$ that closely represents the underlying geometry acquired from the scanning process. As already explained, different representations can be used to model this surface, including for example the raw 3D point cloud provided by range scanners or a computed continuous mesh formed by a set of triangles. In the literature, the process of creating the triangular mesh from the set of points is commonly known as *surface reconstruction* which is a research domain largely studied by the computer graphics and robotics communities.

As explained in Berger et al. (2017), the reconstruction problem is ill-posed as an infinite number of surfaces can pass through or near a given set of data points. Thereof, *prior assumptions* are commonly made in the point cloud itself (i.e. sampling density, noise level) or in the scanned shape (i.e. smoothness, symmetries, shape primitives, global regularity) to resolve ambiguities and combat imperfections in the raw point cloud as introduced in Section 1.2.2.1. For large-scale reconstruction, local implicit methods based on gradient distance fields are commonly employed (Kolluri, 2005; Bouchiba et al., 2020). Nevertheless, one of the main challenges for such techniques is the uneven sampling density. In consequence, these approaches consider a fairly homogeneous density along the point cloud scan representing the underlying surface. This inhibits their application to single-frame LiDAR surface reconstruction.

In this chapter we present an algorithm capable to perform 3D surface reconstruction from heterogeneous density data by using a voxel-based statistical representation. We evaluate our proposal on real and synthetic data and compare our results with the popular IMLS algorithm (Kolluri, 2005). Our contributions can be summarized as:

1. We present an implicit 3D surface reconstruction algorithm from heterogeneous density data by employing an adaptive neighborhood based on a Gaussian confidence function.

2. Our voxel-based representation enables to store statistical data from

underlying points that can be efficiently stored and incrementally updated.

3. Our tests in both real and simulated data show a significant improvement on the quality of the reconstruction for our approach, being able to represent fine details in areas close to the vehicle. Our method shows a better trade-off between accuracy and density of the reconstruction.

3.2 Related work

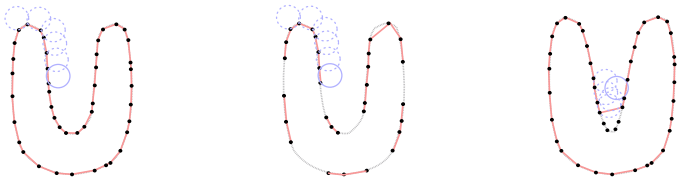
For an exhaustive state of the art in surface reconstruction methods we refer to Berger et al. (2017); You et al. (2020). Classically, surface reconstruction algorithms are classified between explicit and implicit according to the method used to describe the surface of the reconstructed scene or object. Additionally, an alternative set of methods based on deep learning have surged in recent years showing promising results.

3.2.1 Explicit methods

Explicit methods commonly use simple parametric equations or triangulation to obtain a simplex or triangular mesh. One of the most known methods is the Ball Pivoting Algorithm (BPA) (Bernardini et al., 1999) which performs interpolation of the 3D point cloud by pivoting a sphere of a given radius and iteratively joining three-point neighborhoods within the sphere by a single triangle (see Figure 3.1a). The process finishes when all points have been considered. However, the method struggles with uneven sampling densities (see Figure 3.1a) as it relies in heuristics, failing when the curvature of the manifold is larger than the ball radius as shown in Figure 3.1c.

Other approaches exploit geometric primitives by assuming the surface can be described as a set of simple canonical geometric shapes through an explicit equation. Popular approaches commonly encourage the use of random sample consensus techniques (RANSAC) (Fischler and Bolles, 1981) to iteratively calculate the best-fitting model. In Schnabel et al. (2007) the technique is applied to robustly find planes, spheres, cylinders, cones and torii that partially match the input point cloud. Additionally, Jenke et al. (2008) considers a set of detected planar primitives and reconstructs the scene by aligning and merging boundaries of adjacent primitives. However, these methods are fairly limited as some regions can be poorly described by primitives, leading to imperfect or coarse shape descriptions.

Additionally, methods based on the use of Voronoï diagrams apply Delaunay triangulation to obtain the surface mesh (Amenta and Bern, 1998; Gopi et al., 2000; Amenta et al., 2001; Boltcheva and Lévy, 2017). It can be proved that the lying surface is a subset of the Delaunay triangulation envelope if the sampling density is adequate (Cazals and Giesen, 2006).



(a) (b) (c)

Figure 3.1: Ball Pivoting Algorithm (BPA) represented in 2D for simplicity. (a) An evenly scanned surface, a circle of defined radius pivots along sampled points connecting them with edges. (b) low sampling density generates holes in the reconstruction if small radius used. (c) Small curvature can generate non-sampled points if radius used is too large. Source: Bernardini et al. (1999).

However, these methods are computationally expensive which inhibits their implementation in real-time applications. A review of surface reconstruction methods based on Voronoï can be found in Cazals and Giesen (2006).

Finally, some works propose to create local descriptors of the surface by a set of unoriented discs or planes (aka – surfels –) calculated from the points distribution inside defined neighborhoods (Pfister et al., 2000). In Ryde et al. (2013), the surface elements are locally calculated within a voxel grid structure. Surfels have been recently applied for mapping applications (Behley and Stachniss, 2018) showing interesting potential for large-scale implementations. The advantage of these methods is their ease of implementation and low computation needs, but they do not result in a continuous surface representations as the final model is composed of small disconnected surface elements.

3.2.2 Implicit methods

Implicit methods represent the scene from an implicit function commonly modeled by employing gradient distance fields. In Curless and Levoy (1996) the use of a cumulative weighted Signed Distance Function (SDF) is proposed to fuse range information from different viewpoints by using depth cameras. The information from different range images is aligned with a voxel grid where the signed distance values are fused and stored. This technique has long been used for representing 3D volumes in computer graphics (Gibson, 1998; Frisken et al., 2000) and building offline reconstruction of objects from real sensor data (Curless and Levoy, 1996).

These methods gained popularity with the introduction of KinectFusion (Newcombe et al., 2011) where a projective **Truncated Signed Distance Function** (aka **p-TSDF**) is used to fuse information in real time from multiple RGB-D images by tracking the camera poses. The TSDF encodes a

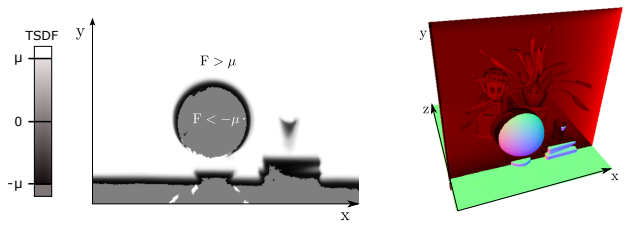
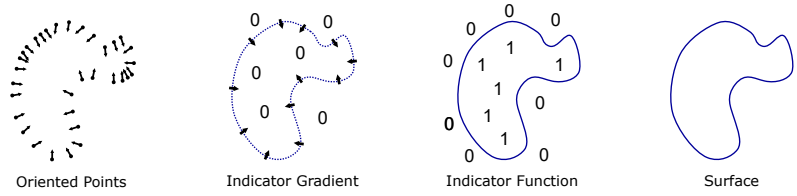


Figure 3.2: 3D scene (right) with a slice through the TSDF volume (left) showing the truncation value μ, $F > \mu$ represents the truncation on observed space (white), gradient shows smooth distance field around iso-surface ($F = 0$), and unobserved space behind truncation threshold is shown in gray. Source: Newcombe et al. (2011).

gradient distance field with opposite signs in areas inside and outside objects respectively. The surface is encoded by the *zero-crossing* (i.e. change of sign within the gradient field) as shown in Figure 3.2. For implementation, voxel grids are commonly employed and the gradient values are stored in the center or corners of each voxel. The approach has been extended to enable faster computation and process larger scenes (Whelan et al., 2012; Steinbrücker et al., 2014). Furthermore, variants have been introduced by updating the fusion scheme with different weighting functions (Oleynikova et al., 2017, 2016). However, these methods focus on multi-frame reconstruction, requiring a large number of viewpoints to output a dense representation and are sensitive to outliers.

Other implicit methods perform global approximations of the surface from where the TSDF is obtained. Poisson reconstruction (Kazhdan et al., 2006; Kazhdan and Hoppe, 2013) is a well-known technique for creating watertight meshes (i.e. meshes consisting in a closed surface with no holes and clearly defined frontiers) from a set of 3D points with oriented normals by solving the Poisson equation as shown in Figure 3.3. While the method is robust to noise and outliers, calculation of consistent oriented normals is not straightforward and the method is more appropriate for object reconstruction rather than complete scenes as a closed mesh is produced.

Previously defined global methods use the entire point set in the surface approximation process. Nevertheless, it is also possible to obtain the implicit surface through local approximations which involves computation only on small subsets of the whole point cloud. The original work of Hoppe et al. (1992) was the first to present a local implicit function from oriented point clouds by computing a signed distance to the tangent plane of the nearest point, estimated via Principal Component Analysis (PCA). However, the method does not produce a continuous surface approximation. An extension was presented in Levin (2004) using Moving Least Squares (MLS)



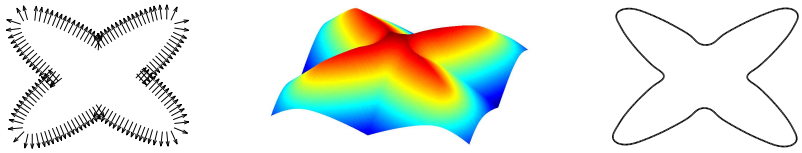
(a) (b) (c) (d)

Figure 3.3: Poisson reconstruction process illustrated in 2D. (a) Oriented points sampled from the surface result in an indicator gradient (b). The surface (d) can be obtained through the indicator function (c) whose Laplacian equals the divergence of the oriented points vector field. Source: Kazhdan et al. (2006).

through a projection operator. Shen et al. (2004) introduced a MLS-based definition of an implicit function to estimate a surface from polygonal data. The same definition was then applied to point clouds through the introduction of the popular Implicit Moving Least Squares (IMLS) method (Kolluri, 2005) which considers the weighted mean of the signed distances to the nearest tangent planes in a local neighborhood (see Figure 3.4). Extensions of the method denominated as Robust-IMLS (Fleishman et al., 2005) and Extended-IMLS (Bouchiba et al., 2020) enable to better recover sharp features in noisy point clouds and extend the function definition to larger extents respectively. Similar to Poisson reconstruction, implicit methods based on signed distance fields depend on oriented normals for correct performance, which are commonly calculated by considering the sensor position and the angle with impacted surface.

The main advantage of the last set of methods based on local scalar functions is their fast computation and their robustness to noisy point clouds. Furthermore, the averaging process of IMLS based methods enables better fitting of the underlying surface. However, these techniques struggle in the presence of heterogeneously sampled surfaces as the neighborhood definition remains constant during the whole computation of the signed distance field. For instance, if the neighborhood is too small the reconstruction will contain holes in low-density areas. On the contrary, large neighborhoods can produce loss of details given larger approximations. Thereof, we introduce an implicit local method capable to adapt the neighborhood definition according to the sampling density.

Explicit mesh from implicit gradient fields. Even though implicit functions produce an implicit gradient field, their final output is generally an explicit surface mesh commonly calculated for visualization purposes or continuous surface definition. Different methods exist in the literature for



(a) (b) (c)

Figure 3.4: Reconstructed surface (c) obtained by the *zero-crossing* of the implicit TSDF (b) obtained by IMLS from the oriented point cloud (a). Source: Kolluri (2005).

generating the mesh from the implicit scalar function and are commonly known as *meshification algorithms*. The most popular one is the marching cubes algorithm (Lorensen and Cline, 1987) which performs triangulation according to the values of the isosurface stored in the vertices of a cube as seen in Figure 3.5. It has been extended in Treece et al. (1999) to better handle ambiguities during the reconstruction by sampling the function on a tetrahedral grid.

3.2.3 Learning-based methods

Learning-based methods are fairly new for surface reconstruction applications. A common framework is to apply learning-based normal estimation techniques (Boulch and Marlet, 2016; Guerrero et al., 2018; Lenssen et al., 2020) to aid traditional reconstruction methods. AtlasNet (Groueix et al., 2018) was among the first methods to propose direct surface reconstruction from the raw point cloud by representing 3D shapes as a set of parametric surface elements.

Additionally, implicit deep learning reconstruction techniques have been presented recently. (Park et al., 2019) regresses a continuous SDF with an *auto-decoder* structure, reconstructing watertight surfaces for objects with complex topologies but failing when applied to large-scale point clouds. Other works propose to predict occupancy or signed distance fields over a voxel grid and obtain the mesh from the predicted representation (Dai et al., 2018; Mescheder et al., 2019). Furthermore, end-to-end surface reconstruction of small objects is presented in Liao et al. (2018) through introduction of a differentiable Marching Cubes variant. Despite recent advances, such methods are commonly applied to the reconstruction of small objects and are limited for large-scale reconstruction though we highlight their large potential.

Finally, scene completion methods (Roldão et al., 2021) could also be considered for reconstruction, but we don't focus on such approaches in this chapter and refer the reader to Part II of this thesis for a deep review of these techniques.

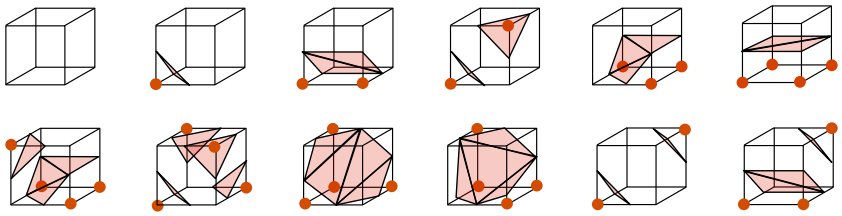


Figure 3.5: Marching cubes algorithm (Lorensen and Cline, 1987). Triangulation is obtained from the signed distance values stored within the vertices of a cube, change of sign between a cube edge represents the zero-crossing. The figure shows some of the possible combinations. Red circles represents positive vertices or vertices outside the surface.

3.3 Method

Let us now introduce our approach. We follow the same line of work presented by implicit methods that locally approximate the surface through a TSDF (Hoppe et al., 1992; Fleishman et al., 2005; Kolluri, 2005; Bouchiba et al., 2020). In contrast to these techniques that are unable to adapt to the heterogeneous density of the input data while keeping the accuracy of the reconstruction, we introduce a method that is able to handle this variable density while keeping a good trade-off between the accuracy and the density of the outputted surface.

Let us consider an input point cloud $\mathcal{P}$ obtained from any range sensor at a known viewpoint and sampled by a voxel grid. Our aim is to reconstruct the underlying 3D mesh. From the statistical distributions of the points in each voxel (Section 3.3.1), we first approximate local planar surfaces (Section 3.3.2), and then compute the implicit surface representation (Section 3.3.3) that encodes the distances to the local planar surfaces. To accommodate to the input data heterogeneous density and gain robustness against noise, we use an adaptive neighborhood kernel, resulting in a denser and smoother reconstruction. The overview of our method is shown in Figure 3.6.

3.3.1 Voxelized representation

We benefit of the work presented in previous chapter (Roldão et al., 2018) to update efficiently a regular voxel-wise representation of the point cloud $\mathcal{P} \rightarrow \mathcal{G} = \{c_1, c_2, \ldots, c_n\}$, with voxel size ω. In addition to the number of points $|c|$, each voxel c stores the 3D statistical distribution of the points lying inside, that is: the mean c_μ and the covariance c_σ. This enables a rich compact representation of the points inside each voxel, while being significantly lighter than storing all the points. Furthermore, the statistical

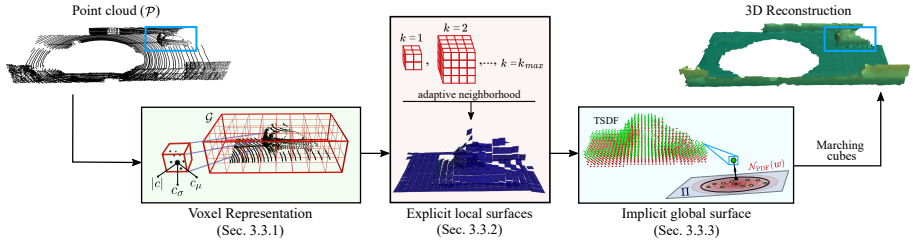


Figure 3.6: Overview of our method. From left to right: the point cloud $\mathcal{P}$; the voxel grid $\mathcal{G}$ at where the statistical distribution of the points is updated (Section 3.3.1); local surface approximations at neighborhoods $k \in [1, k_{max}]$ of the grid vertices (Section 3.3.2); TSDF calculated from the planes by considering its confidences from the statistics distribution (Section 3.3.3); final 3D reconstruction.

distribution can be computed incrementally when inserting new points.

3.3.2 Explicit local surfaces

It has been shown in (Ryde et al., 2013; Behley and Stachniss, 2018), that complex environments are efficiently approximated as a set of primitive local surfaces. This is especially suitable for mobile robotics, as robots usually evolve in well structured environments. Following this observation, we compute local planar surfaces using the 3D statistical distribution previously described. While a naive implementation would fit planar surfaces to each voxel independently, this would inherently lead to a noisy reconstruction since some voxels may have very few points, if not none, due to the heterogeneous density of LiDAR data.

In our pipeline we propose to use an adaptive neighborhood definition where local surfaces are estimated from multi-scale voxels statistics. Our neighborhood definition presents two main advantages: a) it increases the statistical robustness which improves large planar surface estimation (e.g. ground, walls), b) it counterbalances the lack of local data due to low density or occlusion.

Neighborhood definition. We define the multi-scale neighborhood at the vertices location rather than voxel centers since implicit surfaces will later be estimated at each vertex of the voxel representation. Let's consider v a vertex from the voxel-grid representation, its 8 adjacent voxels make up the first neighborhood level denoted $\mathcal{H}^1(v)$. Subsequently, the union of $\mathcal{H}^1(v)$ and the voxels adjacent to $\mathcal{H}^1(v)$ make up the neighborhood $\mathcal{H}^2(v)$. More formally, the neighborhood $\mathcal{H}^k(v)$ is composed of the $(2k)^3$ nearest voxels surrounding v. A two-dimensional illustration of $\mathcal{H}^k(v)$ at levels $k = 1$

(a) Planar surfaces ($k = 1$) (b) Resulting mesh

Figure 3.7: (a) Explicit local planar surfaces with their estimated normals for $k = 1$. (b) Its corresponding mesh reconstruction from the TSDF. Note that applying $k = 1$ neighorhood leads to a noisy surface estimation (e.g. car edges) that can be improved with our adaptive neighborhood strategy.

and $k = 2$ is presented in Figure 3.9.

Following Ryde et al. (2013), for a given neighborhood $\mathcal{H}^k$ we obtain the cardinal $|\mathcal{H}^k|$, statistical mean $\mathcal{H}^k_\mu$, and covariance $\mathcal{H}^k_\sigma$ from the merging of statistical data of all voxels in $\mathcal{H}^k$. This representation enables incremental updates if new data points are perceived. The set of equations is defined as:

$$|\mathcal{H}^k| = \sum_{c^i \in \mathcal{H}^k} |c^i| \,. \tag{3.1a}$$

$$\mathcal{H}^k_\mu = \sum_{c^i \in \mathcal{H}^k} \Gamma^i c^i_\mu \,. \tag{3.1b}$$

$$\mathcal{H}^k_\sigma = \sum_{c^i \in \mathcal{H}^k} \Gamma^i \left[c^i_\sigma + c^i_\mu \left(c^i_\mu\right)^\top - c^i_\mu \left(c^i_\mu\right)^\top \right], \tag{3.1c}$$

with $\Gamma^i = |c^i| / \sum_{c^i \in \mathcal{H}^k} |c^i|$.

Local planar estimation. Having obtained the local statistics, we use the covariance $\mathcal{H}^k_\sigma$ to estimate the local planar surface of $\mathcal{H}^k$ through a PCA if the following equation is satisfied:

$$|\mathcal{H}^k| \;\geq\; N_{min} \,, \tag{3.2}$$

where N_{min} is a hyper parameter that defines the minimum number of points needed within the neighborhood $\mathcal{H}^k$ to estimate the local plane. Suppose $(\overrightarrow{e_1}, \overrightarrow{e_2}, \overrightarrow{e_3})$ the eigen vectors, and $(\lambda_1, \lambda_2, \lambda_3)$ the eigen values associated to $\mathcal{H}^k_\sigma$ with $\lambda_1 \;\geq\; \lambda_2 \geq \;\lambda_3$, we define $\overrightarrow{e_3}$ (the least dominant axis) as the unoriented normal of the planar estimation of the surface at neighborhood

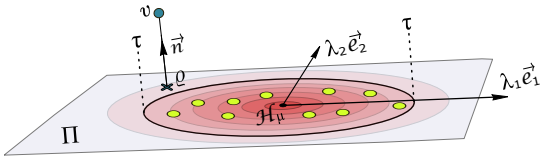


Figure 3.8: The likelihood that ϱ belongs into Π is estimated from $\mathcal{N}_{\text{PDF}}(\varrho \mid \mathcal{H}_\mu^k, \Sigma)$ shown in red. The likelihood must be higher than τ in order to consider Π as a valid plane for ϱ (k indices dropped for clarity).

$\mathcal{H}^k$.s Since normals need to be consistently oriented, the normal $\overrightarrow{n}$ of the plane is oriented towards the sensor pose s_p as follows:

$$\overrightarrow{n} = \begin{cases} \overrightarrow{e_3} & \text{if } \overrightarrow{e_3} \cdot (s_p - \mathcal{H}_\mu) > 0\,, \\ -\overrightarrow{e_3} & \text{otherwise}\,. \end{cases} \tag{3.3}$$

where $\cdot$ stands for the dot product. We denote Π the plane formed by the pair of normal and center coordinates $(\overrightarrow{n}, \mathcal{H}_\mu^k)$. An example of the local planar surfaces estimation is visible in Figure 3.7a, where each plane and its oriented normal is shown in dark and light blue respectively.

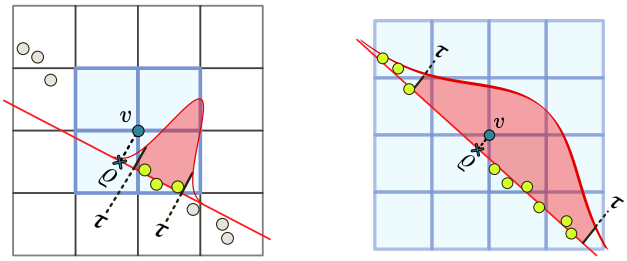
3.3.3 Implicit global surface

To reconstruct the global continuous surface, we compute the Truncated Signed Distance Field (TSDF) for each vertex $v \subseteq c$ that lies inside the truncation threshold.

To cope with the varying density of points in the point cloud, we first compute an optimal neighborhood level k' at each vertex, and then estimate the TSDF value given this optimal neighborhood.

Adaptive neighborhood. A naive implementation of our neighborhood definition would use a constant k throughout the scene. However, large k values will over smooth high density regions, while small values of k will lead to noisy estimate in low density regions. Instead, we compute the optimal neighborhood level k' for each vertex v, given the probability of v to belong to a multivariate Gaussian distribution projected on $\Pi^{k'}$ (the neighborhood planar estimation).

For each vertex, we calculate the projection ϱ^k of v onto the plane $\Pi^k(v)$ and evaluate the likelihood of this projection to belong to the Gaussian $\mathcal{N}^k$, where $\mathcal{N}^k$ is the 2D planar-Gaussian of the statistical distribution of $\mathcal{H}^k$ projected onto Π^k, as illustrated in Figure 3.8. The optimal neighborhood level k' is defined as the smallest level for which the projection of v onto Π^k has a probability to belong to the Gaussian $\mathcal{N}^k$ greater than τ (a hyper parameter). Formally, it is the smallest integer for which the probability density function $\mathcal{N}_{\text{PDF}}$ satisfies:



(a) Neighborhood $k = 1$ (b) Neighborhood $k = 2$

Figure 3.9: Analogous 2D representation of the dynamic neighborhood. Planar surface approximations are performed at different neighborhood levels $k \in [1, k_{max}]$. The considered plane corresponds to the minimum neighborhood level that satisfies Equation 3.4a, where the schematic 1D Gaussian distribution is shown in red.

$$\mathcal{N}^k_{\text{PDF}}(\varrho^k \mid \mathcal{H}^k_\mu, \Sigma) \geq \tau\,;\ \Sigma = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \tag{3.4a}$$

$$k' = \underset{k}{\text{argmin}}(\mathcal{N}^k_{\text{PDF}})\, \forall k \in [1, k_{max}]\,,\ \mathcal{N}^k_{\text{PDF}} \geq \tau. \tag{3.4b}$$

In practice, we compute the optimal k' iteratively starting at level 1 and stops when the above equation is satisfied as it is shown in Figure 3.9. To avoid exponential computation time, k is bounded as $1 \leq k \leq k_{max}$.

TSDF computation. We now compute the TSDF value for each vertex of the voxel grid, from the optimal local neighborhood level, while accounting for the normal estimation at the corresponding level. In other words, we compute TSDF(v) such that:

$$\text{TSDF}(v) = \overrightarrow{n^{k'}} \cdot (v - \mathcal{H}^{k'}_\mu)\,. \tag{3.5}$$

Our adaptive neighborhood level selection can efficiently handle varying local density and fill gaps between missing data, such as those between two adjacent LiDAR layers. Subsequently, the condition on the probability density function given in Equation 3.4a avoids the extension of the surface at its borders which is visible in qualitative results.

After TSDF computation, we use the popular marching cubes (Lorensen and Cline, 1987) explained in tion 3.2.2 to extract the mesh from the zero-crossing of the gradient field. Figure 3.7b shows the reconstructed mesh that results from the planes with constant neighborhood ($k = 1$) seen in Figures 3.7a.

3.4 Experiments

We implement our method in Python and as in the previous chapter, we evaluate its performance on both real and simulated data from the KITTI dataset (Geiger et al., 2013) and the CARLA simulator (Dosovitskiy et al., 2017), respectively. In this case, we perform frame-wise evaluation by comparing the quality of the reconstructed mesh and its distance to ground truth point cloud data. Performance is compared against IMLS (Kolluri, 2005) which we use as baseline. Results from EIMLS variant (Bouchiba et al., 2020) were also calculated with no difference on performance from IMLS. For both synthetic and real data, we used 100 frames equi-sampled from either an urban-like sequence from CARLA or the residential sequence 0018 from the public KITTI dataset.

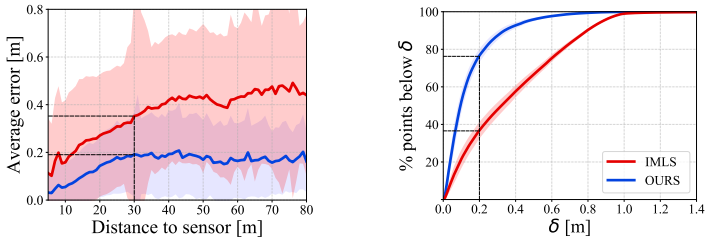
Unless stated otherwise, our hyper-parameters remain unchanged in all experiments, with: voxel size $\omega = 0.2m$, likelihood threshold $\tau = 0.2$, minimum number of points $N_{min} = 10$, and maximum neighborhood size $k_{\max} = 5$. For fair comparison, the spherical neighborhood radius of IMLS is set to $\omega \times k_{\max} = 1m$, and its k-nearest neighbor search to N_{min}. The weight parameter of IMLS is set to $h = 1/3\,(\omega \times k_{\max}) = 0.33$, for the Gaussian weight to consider points at distances equivalent to our largest neighborhood.

Noteworthy, it is impractical to have a real mesh ground-truth and KITTI can only be used for qualitative evaluation. Therefore, we use CARLA to measure the benefit of each of our contributions. The setup in CARLA replicates the top-mounted Velodyne HDL64E from the KITTI dataset. We also simulate a collocated noise-free LiDAR with abnormally high resolution (316 layers), which serves as ground-truth for the reconstruction. Subsequently, we frame the evaluation as a set-to-set distance problem, and measure the quality as the distance of the predicted mesh vertices (P) to the ground-truth (GT) set.

3.4.1 Metrics

We compute the average error for each mesh vertex as a function of the sensor distance and the cumulative *delta error* that indicates the percentage of vertices of the output meshes having an error lower than a given value (Section 3.4.2.1). Additionally, to ablate our contributions (Section 3.4.3) we use two metrics derived from the literature: the Average Error (AE), and the Haussdorf Distance (HD). The former measures the average distance error from one point to its nearest point in the other set, that is:

$$AE_{P \to GT} = \sum_{a \in P} \frac{1}{|P|} \min_{b \in GT} |a - b| \,. \tag{3.6}$$



(a) Average error (b) Delta error

Figure 3.10: Performance on the CARLA dataset from averaging of 100 frames evaluation. While the average error grows with the distance, at 30m distance our surface reconstruction error is $\approx 0.2m$ whereas IMLS (Kolluri, 2005) is $\approx 0.35m$.

The Haussdorf distance (Aspert et al., 2002) is classically used for point set distances and gives a sense of the largest minimum error, it writes:

$$\mathrm{HD}_{P \to GT} = \max_{a \in P} \min_{b \in GT} |a - b| \,. \tag{3.7}$$

As each of the two metrics are directed, we also report the symmetrical metric ($\mathrm{HD}_{\mathrm{sym}}$ and AE_{sym}) as the average of both directed metrics. For instance, the symmetrical Haussdorf metric is given by $\mathrm{HD}_{\mathrm{sym}} = 0.5(\mathrm{HD}_{P \to GT} + \mathrm{HD}_{GT \to P})$. We chose not to use the Chamfer distance used in machine learning (Fan et al., 2017) because it isn't a metric-scale and is thus harder to interpret.

3.4.2 Performance evaluation

We now present quantitative results obtained on CARLA simulator (Section 3.4.2.1) and qualitative results on the KITTI dataset (Section 3.4.2.2).

3.4.2.1 Performance on synthetic data (CARLA)

Reconstruction precision. We report the average error of our method in Figure 3.10a, and the cumulative *delta error* in Figure 3.10b. For both metrics, the accuracy of our reconstruction is significantly higher than the one obtained by IMLS, as the average error is roughly 50% lower (Figure 3.10a). Moreover, the average error of our method remains fairly constant (around $0.2m$) even at larger distances while IMLS error continues to increase.

The percentage of vertices below a given error exhibits the same behavior (Figure 3.10b) with a significant advantage for our method. Furthermore, almost 80% of the vertices of our mesh have an error lower than $0.2m$, while

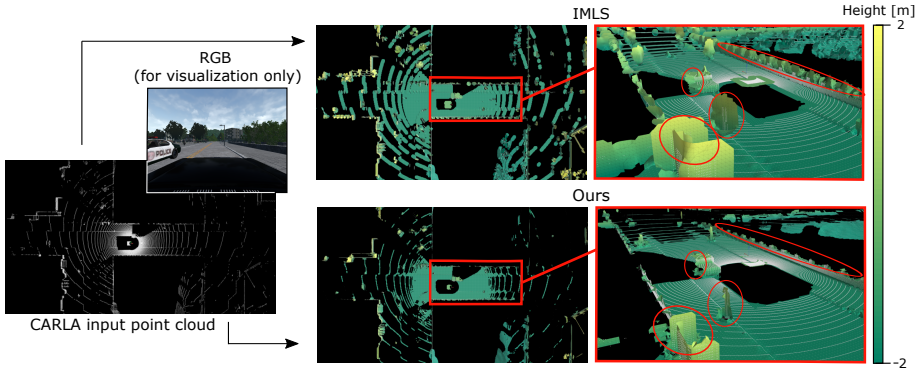


Figure 3.11: Qualitative comparison on CARLA simulator. Notice that even though IMLS outputs a denser reconstruction, it also extends all surfaces at their borders and creates a higher number of artifacts as it can be seen by the red circles at rightmost images. Our method is able to keep the structure of the surface and generates fewer artifacts, performing a more accurate reconstruction, while keeping a good density in areas near to the vehicle.

only 40% of vertices lie below the same threshold with IMLS. These results advocate for the higher precision of our method when compared to IMLS which we attribute to our adaptive neighborhood that adapts to the variable density and the Gaussian confidence function that avoids extension of the surface.

Reconstruction density. A qualitative comparison on CARLA is shown in Figure 3.11. Its is visible that IMLS outputs a more dense reconstruction of the scene but also tends to extend all surfaces, generating artifacts and inaccuracies in the reconstruction. This can be observed in the circled areas where lampposts, traffic signs and walls are abnormally enlarged by IMLS. On the other hand, our method preserves such details in the reconstruction and maintains a high density in areas close to the vehicle, in agreement with the quantitative results obtained in Figure 3.10.

3.4.2.2 Performance on real data (KITTI)

Qualitative results on KITTI residential sequence 0018 are shown in Figure 3.12. As for the results presented on the synthetic data, IMLS outputs a denser reconstruction but extends all surfaces out of the borders and generates artifacts. This can be observed at the circled areas on the rightmost images. Conversely, our method outputs a more accurate reconstruction, keeping the structure of the scanned surface and completing small regions of missing data at areas close to the sensor. Further quantitative

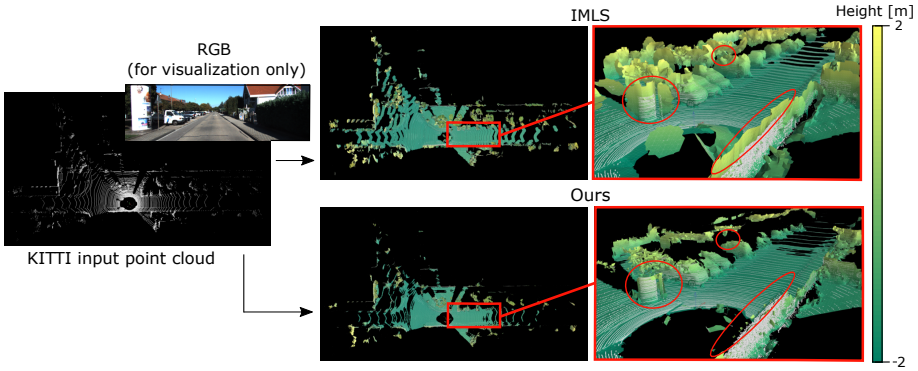


Figure 3.12: Visual comparison on KITTI dataset. IMLS performs a denser reconstruction at expenses of accuracy. Our method keeps a good trade-off between density and accuracy on the reconstruction.

results might be obtained by randomly sub-sampling the input data and calculating the reconstruction error over the non sub-sampled mesh. For qualitative performance on a large sequence, we refer to our demo video (`https://youtu.be/iyKShCBAW9g`).

3.4.3 Ablation studies

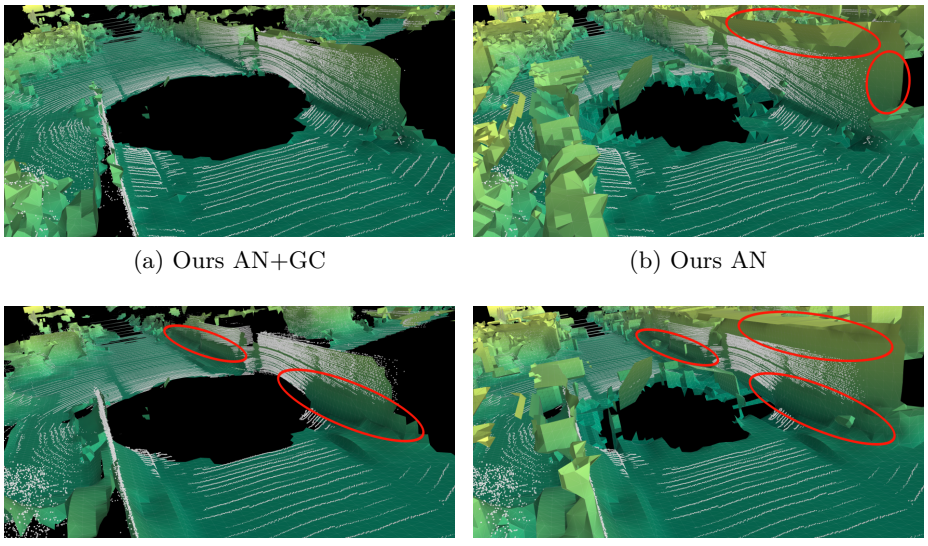
To evaluate the importance of our contributions, we compare the benefits of our Adaptive Neighborhood (**AN**, Section 3.3.3) with or without the Gaussian Confidence (**GC**, Equation 3.4a) and also with a naive Constant Neighborhood (**CN**) approach. More precisely, the neighborhood $\mathcal{H}^{k'}$ considered for the plane estimation (cf. Section 3.3.2) differs such that:

1. **AN+GC:** $\mathcal{H}^{k'}$ is the minimum neighborhood among $\left\{\mathcal{H}^1, \cdots, \mathcal{H}^{k_{max}}\right\}$ that satisfies Equation 3.2 and 3.4a.
2. **AN:** $\mathcal{H}^{k'}$ is the minimum neighborhood among $\left\{\mathcal{H}^1, \cdots, \mathcal{H}^{k_{max}}\right\}$ that satisfies Equation 3.2.
3. **CN+GC:** $\mathcal{H}^{k'}$ with $k' = k$, is considered only if Equation 3.2 and 3.4a are satisfied.
4. **CN:** $\mathcal{H}^{k'}$ with $k' = k$, is considered only if Equation 3.2 is satisfied.

In Table 3.1, the accuracy on the reconstruction directly affects the P→GT distances while the GT→P distances are mostly influenced by the reconstruction density. As expected, the benefit of our adaptive neighborhood strategy is noticeable when comparing *AN+GC* and *CN+GC*. Not only *AN+GC* exhibits higher accuracy (lower P→GT) but it also increases

Method		Average Error (m) P→GT	GT→P	Sym.	Hausdorff Distance (m) P→GT	GT→P	Sym.
IMLS (Kolluri, 2005)		0.37	0.09	0.23	4.54	8.32	**6.43**
Ours AN+GC	$\mathbf{k_{max} = 5}$	0.14	0.13	**0.14**	1.39	20.49	10.94
	$k_{max} = 3$	0.09	0.26	0.17	0.69	30.84	15.77
Ours AN	$k_{max} = 5$	0.30	0.12	0.21	1.69	20.36	11.02
	$k_{max} = 3$	0.14	0.25	0.19	0.87	30.83	15.85
Ours CN+GC	$k = 5$	0.15	0.16	0.15	1.38	20.50	10.94
	$k = 3$	0.09	0.27	0.18	0.69	30.85	15.77
	$k = 1$	0.03	3.44	1.73	0.24	65.28	32.76
Ours CN	$k = 5$	0.30	0.14	0.22	1.69	20.36	11.03
	$k = 3$	0.14	0.26	0.20	0.87	30.83	15.85
	$k = 1$	0.03	3.44	1.73	0.24	65.28	32.76

Table 3.1: Performance on synthetic data evaluated on 100 frames from simulated HDL-64 LiDAR on CARLA simulator (Dosovitskiy et al., 2017). Accuracy on the reconstruction directly affects the P→GT distances while the GT→P distances are mostly influenced by the reconstruction density. AN: Adaptive neighborhood, GC: Gaussian confidence, CN: Constant neighborhood.



(a) Ours AN+GC (b) Ours AN

(c) Ours CN+GC (d) Ours CN

Figure 3.13: Qualitative comparison of our method by removing the main components of our proposal. Images correspond to a single frame of a point cloud from CARLA. In shown images k or k_{max} equals 5. Notice that the adaptive neighborhood (AN) preserves high level of detail and the Gaussian confidence (GC) function avoids extension of the surface at its borders.

the reconstruction density (lower GT→P). When using our Gaussian Confidence (GC), there is a slight density loss (lower GT→P) but a significant accuracy increase. Qualitatively from Figure 3.13, our adaptive neighborhood (AN) helps to maintain the details of the surface by not over-smoothing the data, while the Gaussian Confidence (GC) strategy avoids to extend the surface, keeping its structure and generating fewer artifacts.

Overall, we observe that our complete pipeline (AN+GC with $k_{\max=5}$) keeps a good trade-off between accuracy and density, which is shown by the best result obtained AE_{sym} ($0.14m$ vs. $0.23m$ for IMLS). With larger neighborhood (bigger $k_{\max}$), the density of the reconstruction increases but at the expense of a lower accuracy which is intuitive as there is a need to extrapolate more the data. Since IMLS performs a denser reconstruction, lower GT→P distances are obtained, which explains the best results on the symmetric Hausdorff Distance ($10.94m$ vs. $6.43m$ for IMLS). While our method is less dense, our predicted mesh is significantly more precise ($1.39m$ vs. $4.54m$ for IMLS).

3.5 Conclusion

In this chapter we proposed a pipeline to effectively perform 3D surface reconstruction of the scene from heterogeneous density LiDAR data. Our method uses an adaptive neighborhood strategy coupled with a Gaussian confidence estimation to best predict local surfaces converted into an implicit TSDF used to obtain the final mesh. The use of a statistical grid-based representation serves to reduce memory needs, perform fast updates and directly compute the gradient field at desired neighborhood.

However, our method present some limitations that we believe are important to remark. One of them is that we rely on the computation of oriented normals which is a challenging problem. However, this limitation affects any algorithm based on signed distance fields (Hoppe et al., 1992; Kolluri, 2005; Bouchiba et al., 2020). Another limitation is the iterative nature of our confidence function which impacts the speed of our method. This could be solved by directly selecting the appropriate neighborhood according to the point density within each voxel, though we believe this could impact accuracy of the reconstruction. Our method was evaluated in both synthetic (CARLA) an real data (KITTI), outperforming the classical IMLS (Kolluri, 2005) though with sparser reconstruction. While our results show that local planar surfaces are sufficient for accurate reconstruction, the method could be extended to more complex surface definitions (i.e. polynomials).

Different to the occupancy grid approach presented in Chapter 2, the surface reconstruction enables to obtain a continuous representation which is of high interest for applications as terrain traversability assessment or physical modeling. Like in previous chapter, we found that sparsity plays

a crucial role and it is the main property to tackle when performing scene representation. However, surface reconstruction methods are not sufficient for completing large missing areas. In the next part of this manuscript, we will focus on deep-learning methods and show how they can be useful to complete such regions, drawing from our experience on 3D reconstruction, gained in these first two chapters.

Part II

3D Semantic Scene Completion

In this part, we transition towards deep learning applications in order to reconstruct and complete large regions of occluded or missing data. Furthermore, studied methods jointly perform semantic segmentation for complete understanding of the scene. This task is commonly known in the literature as *semantic scene completion*.

Chapter 4

3D Semantic Scene Completion: Survey

The contributions of this chapter have been presented in Roldão et al. (2021), submitted to the International Journal of Computer Vision (IJCV):

Roldão, L., de Charette, R., and Verroust-Blondet, A. (2021). 3D semantic scene completion: a survey. *ArXiv 2021*.

Although this work was conducted after our proposed semantic scene completion approach (Roldão et al., 2020) presented in Chapter 5, we swap the order in this manuscript for readability.

Contents

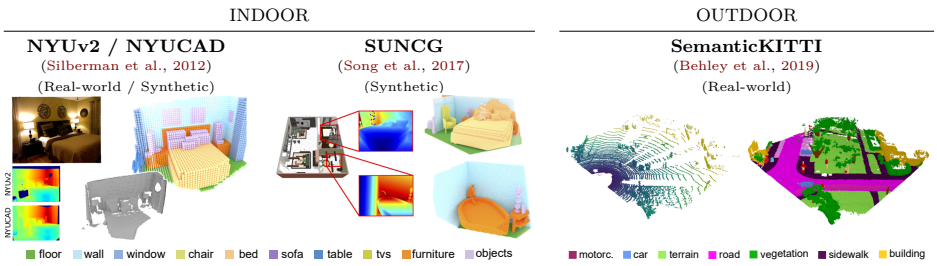


Figure 4.1: **Popular datasets for Semantic Scene Completion (SSC).** From an incomplete input view the SSC task consists in the joint estimation of both geometry and semantics of the scene. The figure shows the 4 most popular datasets for SSC, each showing input data and ground truth. The complexity of SSC lies in the completion of unobserved / occluded regions and in the sparse supervision signal (notice that real ground truth are incomplete).

4.1 Introduction

Understanding of 3D surroundings is a natural ability for humans who effortlessly leverage prior knowledge to estimate geometry and semantics, even in large occluded areas. This proves more difficult for computers which has drawn wide interest from computer vision researchers in recent years (Guo et al., 2020). Indeed, 3D scene understanding is a crucial feature for many applications, such as robotic navigation or augmented reality, where geometrical and semantics understanding is essential to leverage interaction with the real world (Garg et al., 2020). Nonetheless, vision sensors only provide partial observations of the world given their limited field of view, sparse sensing and measurement noise, capturing a partial and incomplete representation of the scene.

To address this, *scene reconstruction* algorithms focus on inferring the geometry of a scene given one or more 2D/3D observations by using interpolation methods to complete small areas of missing data (refer to Part I of this thesis). With recent advances of deep learning, Scene Completion (SC) has been introduced to complete large unseen regions by the use of data-driven approaches. Furthermore, Semantic Scene Completion (SSC) has been presented as an extension of SC, where semantics and geometry are *jointly* inferred, departing from the idea that they are entangled (Song et al., 2017). Thus the SC and SSC tasks have significantly departed from original scene reconstruction in terms of nature and sparsity of the input data. As defined in Garbade (2019), the task can be considered as a semantic spatial anticipation of the occluded sections of the scene. This means that given one or multiple sparse observations of the scene from defined viewpoints, we need to extrapolate the scene to parts that are outside the field of view of the sensor. Figure 4.1 shows samples of input and ground

truth for the four most popular SSC datasets.

The complexity of the SSC task lies in the sparsity of the input data (see holes in depth/LiDAR input), and the incomplete ground truth (resulting of frame aggregation) providing a rather weak guidance. Different from reconstruction where multiple views are aggregated, SSC requires to extract an in-depth understanding of the scene heavily relying on learned priors to resolve ambiguities. The increasing number of large scale datasets (Behley et al., 2019; Song et al., 2017; Silberman et al., 2012; Straub et al., 2019) have encouraged new SSC works in the last years. On connex topics, 3D (deep) vision has been thoroughly reviewed (Lu and Shi, 2020; Guo et al., 2020; Liu et al., 2019; Li et al., 2020) including surveys on 3D representations (Ahmed et al., 2018) and task-oriented reviews like 3D semantic segmentation (Zhang et al., 2019a; Xie et al., 2020b), 3D reconstruction (Zollhöfer et al., 2018; Pintore et al., 2020), 3D object detection (Jiao et al., 2019), etc. Still, no survey existed on this hot SSC topic and navigating the literature was not trivial.

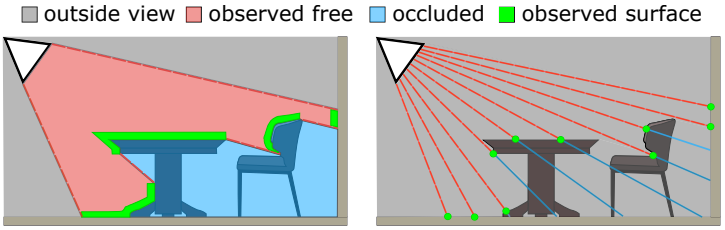
Therefore, we proposed the first comprehensive and critical review of the Semantic Scene Completion (SSC) literature, focusing on methods and datasets. Our contributions can be summarized as:

- We propose the first in-depth SSC survey, exposing a critical analysis of the SSC topic and highlight the missing areas for future research.
- We aim to provide new insights to informed readers and help new ones navigate in this emerging field which gained significant momentum in the past few years.
- We present quantitative and qualitative comparisons and in-depth analysis of existing methods for SSC.

To study SSC, this chapter is organized as follows. We first introduce and formalize SSC 4.2, briefly brushing its historical background. Ad-hoc datasets employed for the task and introduction to common 3D scene representations is covered in Section 4.3. We study the existing works in Section 4.4, highlighting the different input encodings, deep architectures, design choices, and training strategies employed. The section ends with an analysis of the current performances, followed by a discussion in Section 4.5.

4.2 Problem definition

Let x be an incomplete 3D representation of a scene, Semantic Scene Completion is the function $f(.)$ inferring a dense semantically labeled scene $\hat{y}$ such that $f(x) = \hat{y}$ best approximates the real 3D scene y. Most often, x is significantly sparser than y and the complexity lies in the inherent ambiguity, especially where large chunks of data are missing, due to sparse



(a) Cameras (b) LiDAR

Figure 4.2: **Scene acquisition.** A camera (RGB, RGB-D, Depth) senses dense volumes but produces noisy depth measurements (a), while LiDAR – more accurate – is significantly sparser (b). (Inspired from: Song et al. (2017)).

sensing or occlusions (see Figure 4.2). Subsequently, the problem cannot be addressed by interpolating data in x and is most often solved by learning priors from (x, y) pairs of sparse input and dense 3D scenes with semantic labels.

The nature of the sparse 3D input x greatly affects the task complexity. While 3D data can be obtained from a wide variety of sensors, RGB-D/stereo cameras or LiDARs are commonly employed. The former, for example, provide a dense description of the visible surfaces where missing regions correspond to occluded areas , as shown in Figure 4.2a. This reduces the SSC task to estimating semantic completion only in the occluded regions (Song et al., 2017). Conversely, LiDAR data provides considerably sparser sensing, with density decreasing afar and point-wise returns from laser beams cover an infinitesimal portion of the space leading to high proportion of unknown volume, as shown in Figure 4.2b.

The rest of this section provides the reader with a brief historical background covering early works and foundations.

4.2.1 Historical background

Semantic scene completion inspires from both shape completion and semantic segmentation, and thereof benefits from their individual insights. We briefly review their own literature pointing to the respective surveys.

Completion. Completion algorithms initially used interpolation (Davis et al., 2002) or energy minimization (Sorkine-Hornung and Cohen-Or, 2004; Nealen et al., 2006; Kazhdan et al., 2006) techniques to complete small missing regions. Completion works were first devoted to object shape completion, which infers occlusion-free object representation. For instance, some trends exploit symmetry (Pauly et al., 2008; Sipiran et al., 2014; Sung et al., 2015; De Charette and Manitsaris, 2019; Thrun and Wegbreit, 2005) and are reviewed in Mitra et al. (2013). Another common approach is to rely on *a*

priori 3D models to best fit sparse input (Li et al., 2015; Pauly et al., 2005; Shen et al., 2012; Li et al., 2017; Rock et al., 2015). In recent years, model-based techniques and new large-scale datasets enlarged the scope of action by enabling inference of complete occluded parts in both scanned objects (Dai et al., 2017b; Yuan et al., 2018; Rezende et al., 2016; Yang et al., 2019; Sharma et al., 2016; Han et al., 2017; Smith and Meger, 2017; Stutz and Geiger, 2018; Varley et al., 2017; Park et al., 2019) and entire scenes (Dai et al., 2020; Firman et al., 2016; Zimmermann et al., 2017). Moreover, contemporary research shows promising results on challenging multi-object reconstruction from single RGB image (Engelmann et al., 2021). For further analysis, we refer readers to related surveys (Yang et al., 2019; Han et al., 2019a; Mitra et al., 2013).

Semantics. Traditional segmentation techniques reviewed in Nguyen and Le (2013), were based on hand-crafted features, statistical rules and bottom-up procedures, combined with traditional classifiers. The advances on deep learning has reshuffled the cards (Xie et al., 2020b). Initial 3D deep techniques relied on multiviews processed by 2D CNNs (Su et al., 2015; Boulch et al., 2017, 2018) and was quickly replaced by the use of 3D CNNs which operate on voxels (Maturana and Scherer, 2015; Riegler et al., 2017b; Tchapmi et al., 2017; Wang et al., 2018b; Meng et al., 2019), tough suffering from memory and computation shortcomings. Point-based networks (Qi et al., 2017b,a; Thomas et al., 2019; Landrieu and Simonovsky, 2018; Wang et al., 2019b; Li et al., 2018) remedied this problem by operating on points and quickly became popular for segmentation, though generative task like SSC are still a challenge. We refer readers to dedicated surveys on semantic segmentation (Xie et al., 2020b; Gao et al., 2020).

Song *et al.* (Song et al., 2017) were the first to address semantic segmentation and scene completion jointly, showing that both tasks can mutually benefit. Since then, many SSC works gather ideas from the above described lines of work and are extensively reviewed in Section 4.4.

4.3 Datasets and representations for SSC

This section presents existing SSC datasets (Section 4.3.1) and commonly used 3D representations for SSC (Section 4.3.2).

4.3.1 Datasets

A comprehensive list of all SSC ready datasets is presented in Table 4.1. We denote as *SSC ready* any dataset containing pairs of sparse/dense data with semantics label. Note that while 14 datasets meet these criteria, only

half has been used for SSC among which the four most popular are drawn in bold in the table and previewed in Figure 4.1. Overall, there is a prevalence of indoor stationary datasets (Silberman et al., 2012; Song et al., 2017; Wu et al., 2020; Chang et al., 2017; Dai et al., 2017a; Armeni et al., 2017; Hua et al., 2016; Handa et al., 2016) as opposed to outdoors (Behley et al., 2019; Pan et al., 2020; Griffiths and Boehm, 2019).

Datasets creation. Synthetic datasets can easily be obtained by sampling 3D object meshes (Song et al., 2017; Handa et al., 2016; Fu et al., 2020) or simulating sensors in virtual environments (Dosovitskiy et al., 2017; Gaidon et al., 2016; Ros et al., 2016; Griffiths and Boehm, 2019). Their evident advantage is the virtually free labeling of data, though transferability of synthetically learned features is arguable. Real datasets are quite costly to record and annotate, and require a significant processing effort. Indoor datasets (Chang et al., 2017; Silberman et al., 2012; Dai et al., 2017a; Xiao et al., 2013; Hua et al., 2016; Armeni et al., 2017; Wu et al., 2020) are commonly captured with RGB-D or stereo sensors. Conversely, outdoor datasets (Behley et al., 2019; Caesar et al., 2020; Pan et al., 2020) are often recorded with LiDAR and optional camera. They are dominated by autonomous driving applications, recorded in (peri)-urban environment, and must subsequently account for dynamic agents (e.g. moving objects, egomotion, etc.).

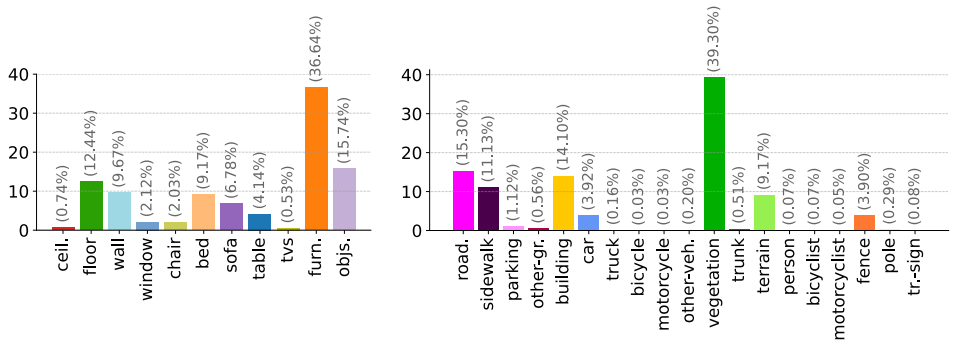
Ground truth generation. An evident challenge is the dense annotation of such datasets. Specifically, while indoor stationary scenes can be easily entirely captured from multiple views or rotating apparatus, 3D outdoor dynamic scenes are virtually impossible to capture *entirely* as it would require ubiquitous scene sensing.

Subsequently, ground truth y is obtained from the aggregation and labeling of sparse sequential data $\{y_0, y_1, ..., y_T\}$ over a small time window T. Multi-frame registration is usually leveraged to that matters, assuming that consecutive frames have overlapping field of view. For RGB-D datasets, mostly stationary and indoors, it is commonly achieved from Structure from Motion (SfM) (Nair et al., 2012; Saputra et al., 2018) or visual SLAM (Saputra et al., 2018) (aka vSLAM), which causes holes, missing data and noisy annotations (Fuentes-Pacheco et al., 2012). Such imperfections are commonly reduced by inferring dense complete geometry of objects with local matching of CAD models (Firman et al., 2016; Guo and Hoiem, 2013; Wu et al., 2020), or post-processing hole filling techniques (Straub et al., 2019). In outdoor settings, point cloud registration techniques (Pomerleau et al., 2015, 2014) enable the co-location of multiple LiDAR measurements into a single reference coordinate system (Behley et al., 2019).

Dataset	Year	Type	Nature	Input↦Ground truth	3D Sensor	# Classes	Tasks*						#Sequences	#Frames
							SSC	DE	SPE	SS	OC	SNE		
NYUv2 (Silberman et al., 2012)[a]	2012	Real-world†	Indoor	RGB-D → Mesh/Voxel	RGB-D	894 (11)	✓	✓	✓	✓	✓		1449	795/654
SUN3D (Xiao et al., 2013)	2013	Real-world	Indoor	RGB-D → Points	RGB-D	-	✓	✓	✓	✓	✓		254	-
NYUCAD (Firman et al., 2016)[b]	2013	Synthetic	Indoor	RGB-D → Mesh/Voxel	RGB-D	894 (11)	✓	✓	✓	✓	✓		1449	795/654
SceneNet (Handa et al., 2016)	2015	Synthetic	Indoor	RGB-D → Mesh	RGB-D‡	11	✓	✓		✓	✓		57	-
SceneNN (Hua et al., 2016)	2016	Real-world	Indoor	RGB-D → Mesh	RGB-D	22	✓	✓		✓	✓		100	-
SUNCG (Song et al., 2017)	2017	Synthetic	Indoor	Depth → Mesh/Voxel	RGB-D‡	84 (11)	✓			✓			45622	139368/470
Matterport3D (Chang et al., 2017)	2017	Real-world	Indoor	RGB-D → Mesh	3D Scanner	40 (11)	✓	✓		✓			707	72/18
ScanNet (Dai et al., 2017a)	2017	Real-world	Indoor	RGB-D → Mesh	RGB-D	20 (11)	✓	✓		✓	✓	✓	1513	1201/312
2D-3D-S (Armeni et al., 2017)	2017	Real-world	Indoor	RGB-D → Mesh	3D Scanner	13	✓	✓		✓	✓	✓	270	-
SUNCG-RGBD (Liu et al., 2018)[c]	2018	Synthetic	Indoor	RGB-D → Mesh/Voxel	RGB-D‡	84 (11)	✓			✓			45622	13011/499
SemanticKITTI (Behley et al., 2019)	2019	Real-world	Outdoor	Points/RGB → Points/Voxel	Lidar-64	28 (19)	✓	✓		✓			22	23201/20351
SynthCity (Griffiths and Boehm, 2019)	2019	Synthetic	Outdoor	Points → Points	Lidar‡	9	✓			✓			9	-
CompleteScanNet (Wu et al., 2020)[d]	2020	Real-world†	Indoor	RGB-D → Mesh/Voxel	RGB-D	11	✓	✓		✓	✓	✓	1513	45448/11238
SemanticPOSS (Pan et al., 2020)	2020	Real-world	Outdoor	Points/RGB → Points	Lidar-40	14	✓	✓		✓			2988	-

‡ Simulated sensor. † Synthetically augmented. [a] Mesh annotations from Guo and Hoiem (2013). [b] Derivates from **NYUv2** (Silberman et al., 2012) by rendering depth images from mesh annotation.
[c] Derivates from subset of **SUNCG** (Song et al., 2017) where missing RGB images were rendered. [d] Derivates from **ScanNet** (Dai et al., 2017a) by fitting CAD models to dense mesh.
* SSC: Semantic Scene Completion; DE: Depth Estimation; SPE: Sensor Pose Estimation; SS: Semantic Segmentation. OC: Object Classification; SNE: Surface Normal Estimation

Table 4.1: **SSC datasets**. We list here datasets readily usable for the SSC task in chronological order. Popular datasets are in **bold** and previewed in Figure 4.1. Classes show the total number of semantic classes and when it differs, SSC classes in parenthesis.



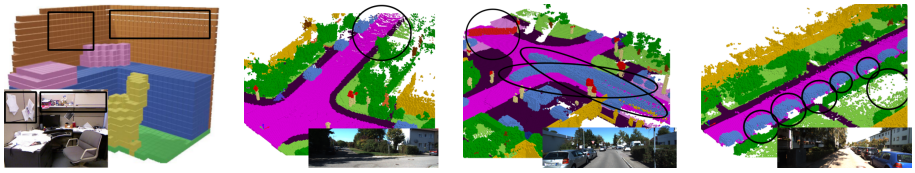
(a) NYUv2 (Silberman et al., 2012) (b) SemanticKITTI (Behley et al., 2019)

Figure 4.3: **Semantics distribution.** Class-wise frequencies of the most popular real datasets prove to be highly imbalanced.

Interestingly, while frequently referred as *dense*, the ground truth scenes are often noisy and non-continuous in real datasets, being in fact an *approximation* of the real scene. Firstly, regardless of the number of frames used, some portions of the scene remain occluded, see Figure 4.4d, especially in dynamic environments. Secondly, sensors accuracy and density tend to steadily decrease with the distance, as in Figure 4.4b. Thirdly, rigid registration can only cope with viewpoint changes, which leads to dynamic objects (e.g. moving cars) producing *traces* which impact on the learning priors is still being discussed (Rist et al., 2020a; Roldão et al., 2020; Yan et al., 2021), see Figure 4.4c. Finally, another limitation lies in the sensors, which only sense the geometrical surfaces and not the volumes, turning all solid objects into shells. To produce semantics labels, the common practice is to observe the aggregated 3D data from multiple virtual view points to minimize the labeling ambiguity. This process is tedious and prone to errors[1], visible in Figure 4.4a. Ultimately, semantics distribution is highly imbalanced as shown in the two most used indoor/outdoor datasets Figure 4.3.

Indoor datasets. From Table 4.1, **NYUv2** (Silberman et al., 2012) (aka NYU-Kinect) is the most popular indoor *real-world* dataset, composed of mainly office and house room sceneries. Despite complete 3D ground truth scene volumes not being originally provided, they have been generated in Guo and Hoiem (2013) by using 3D models and 3D boxes or planes for producing complete synthetically augmented meshes of the scene, generating 1449 pairs of RGB-D images and 3D semantically annotated volumes. Extension of the mesh volumes to 3D grids have been done in Song et al. (2017). However, mesh annotations are not always perfectly aligned with the original depth

[1] Authors of SemanticKITTI report than semantic labeling a hectare of 3D data takes approx. 4.5hr (Behley et al., 2019).



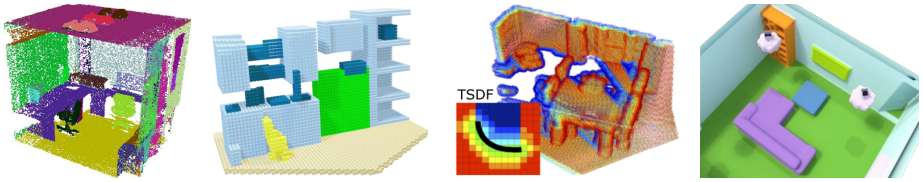
(a) Misslabeling (b) Sparse sensing (c) Object motion (d) Occlusions

Figure 4.4: **Inaccuracies of SSC ground truth.** Providing semantics and geometrics annotation in real-world datasets proves to be complex, and the resulting process is imperfect – which lead to noisy supervision signal. Ground truth mislabeling is a well known bias (a). Sparsity is common in LiDAR-based datasets (b). Object motion causes temporal traces (c), and occlusions are inevitable in dynamic scenes (d). (Source: Silberman et al. (2012); Behley et al. (2019)).

images. To solve this, many methods also report results by using depth maps rendered from the annotations directly as in Firman et al. (2016). This variant is commonly named as **NYUCAD** and provides simplified data pairs at the expense of geometric detail loss. Additional indoor real-world datasets as **Matterport3D** (Chang et al., 2017), **SceneNN** (Hua et al., 2016) and **ScanNet** (Dai et al., 2017a) can be used for completing entire rooms from incomplete 3D meshes (Dai et al., 2020, 2018). The latter has also been synthetically augmented from 3D object models and referred to as **CompleteScanNet** (Wu et al., 2020), providing cleaner annotations. Additionally, smaller **SUN3D** (Xiao et al., 2013) provides RGB-D images along with registered semantic point clouds. Note that datasets providing 3D meshes or point clouds easily be voxelized as detailed in Song et al. (2017). Additionally, Stanford **2D-3D-S** (Armeni et al., 2017) provides 360° RGB-D images, of interest for completing entire rooms (Dourado et al., 2020b). Due to real datasets small sizes, low scene variability, and annotation ambiguities, *synthetic* **SUNCG** (Song et al., 2017) (aka SUNCG-D) was proposed, being a large scale dataset with pairs of depth images and complete synthetic scene meshes. An extension containing RGB modality was presented in Liu et al. (2018) and known as **SUNCG-RGBD**. Despite its popularity, the dataset is no longer available due to copyrights infringement[2], evidencing a lack of synthetic indoor datasets for SSC, that could be addressed using **SceneNet** (Handa et al., 2016).

Outdoor datasets. **SemanticKITTI** (Behley et al., 2019) is the most popular large-scale dataset and currently the sole providing single sparse and dense semantically annotated point cloud pairs from *real-world* scenes. It derivates from the popular odometry benchmark of the **KITTI** dataset

[2]https://futurism.com/tech-suing-facebook-princeton-data



(a) Point Cloud (b) Voxel Grid (c) Implicit Surface (d) Mesh

Figure 4.5: **SSC representations.** Several 3D representations co-exist in the literature. Its choice has major impact on the method to use, as well as the memory or the computation needs. (Source: Wang et al. (2018c, 2019c); Song et al. (2017); Dai et al. (2018))

(Geiger et al., 2013), which provides careful registration and untwistering considering vehicle's ego-motion. Furthermore, voxelized dense scenes were later released as part of an evaluation benchmark with hidden test set [3]. The dataset presents big challenges given the high scene variability and the high class imbalance naturally present in outdoor scenes (Figure 4.3b). **SemanticPOSS** (Pan et al., 2020) also provides single sparse semantic point clouds and sensor poses in same format as the latter to ease its implementation. Furthermore, *synthetic* **SynthCity** additionally provides dense semantic point clouds and sensor poses. It has the advantage of excluding dynamic objects, which solves the effect of object motion (cf. Figure 4.4c), but not occlusions (cf. Figure 4.4d).

4.3.2 3D SSC representations

We now detail the common 3D representation for SSC output, shown in Figure 4.5. While voxel grid or point cloud are the most used, other implicit/explicit representations are of interest for applications like rendering, fluid simulation, etc.

Point cloud. It is a convenient and memory efficient representation, which expresses scene geometry in the 3D continuous world as a set of points lying on its surface. Different from others, point cloud omits definition of the free space. Few works have in fact applied point-based SSC (Zhong and Zeng, 2020), due to the complexity of point generative tasks. Notably, PCN (Yuan et al., 2018) was the first to address object completion followed by Wang et al. (2020d,b); Zhang et al. (2020); Wang et al. (2019d); Wen et al. (2020); Wang et al. (2020c); Tchapmi et al. (2019); Xie et al. (2020a).

Voxel grid. It encodes scene geometry as a 3D grid, in which cells describe semantic occupancy of the space. Opposed to point clouds, grids

[3] http://www.semantic-kitti.org/tasks.html

conveniently define neighborhood with adjacent cells, and thus enable easy application of 3D CNNs, which facilitates to extend deep learning architectures designed for 2D data into 3D (Song et al., 2017; Garbade et al., 2019; Guo and Tong, 2018; Roldão et al., 2020; Li et al., 2020b; Zhang et al., 2018a, 2019b; Chen et al., 2020a, 2019b; Dai et al., 2020, 2018; Dourado et al., 2020a,b; Cherabier et al., 2018). However, the representation suffers from constraining limitations and efficiency drawbacks since it represents both occupied and free regions of the scene, leading to high memory and computation needs. Voxels are also commonly used as a support for implicit surface definition which we now describe.

Implicit surface. It encodes geometry as a gradient field expressing the signed distance to the closest surface, known as the Signed Distance Function (SDF). For scene completion, the value of the gradient field is estimated at specific locations, typically at the voxel centers, for voxel grids (Dai et al., 2018, 2020), or at the point locations for point clouds (Rist et al., 2020a). Implicit surface may also be used as input (Zhang et al., 2018a; Wang et al., 2019c; Chen et al., 2019b; Song et al., 2017; Dai et al., 2020, 2018; Dourado et al., 2020a,b; Li et al., 2020b; Chen et al., 2020a; Zhang et al., 2019b) to reduce the sparsity of the input data, at the expense of greedy computation. For numerical reason, most works encode in fact a flipped version (cf. f-TSDF, Section 4.4.1). Meshes, explained in details below, can be obtained from the implicit surface, by using meshification algorithms such as marching cubes (Lorensen and Cline, 1987).

Mesh. It enables an explicit surface representation of the scene by a set of polygons. Implementing deep-learning algorithms directly on 3D meshes is challenging and most works obtain the mesh from intermediate implicit voxel-based TSDF representations (Dai et al., 2018, 2020) by minimizing distance values within voxels and applying meshification algorithms (Lorensen and Cline, 1987). Other alternatives contemplate applying view inpainting as in Han et al. (2019b) or using parametric surface elements (Groueix et al., 2018), which are more oriented to object/shape completion. Furthermore, recent learning-based algorithms such as Deep Marching Cubes (Liao et al., 2018) enable to obtain continuous meshes end-to-end from well sampled point clouds, but similarly have not been applied to fill large areas of missing information or scenes.

4.4 Methods overview

The seminal work of Song et al. (2017) first addressed Semantic Scene Completion (SSC) with the observation that semantics and geometry are 'tightly intertwined'. While there have been great progresses lately, the best methods

still perform little below 30% mIoU on the most challenging SemanticKITTI benchmark (Behley et al., 2019), advocating that there is significant margin for improvement.

Inferring a dense 3D scene from 2D or sparse 3D inputs is in fact an ill-posed problem since the input data are not sufficient to resolve all ambiguities. As such, apart from Zheng et al. (2013); Lin et al. (2013); Geiger and Wang (2015); Firman et al. (2016), all existing works rely on deep learning to learn semantics and geometric priors from large scale datasets reviewed in Section 4.3.1.

In the following we provide a comprehensive survey of the SSC literature. Unlike some historical computer vision tasks, for SSC we found little consensus and a wide variety of choices exists. As such we also focus on the remaining avenues of research to foster future works.

The survey is organized in sections which follow of a standard SSC pipeline, with each section analyzing the different line of choices. We start in Section 4.4.1 by reviewing the various input encoding strategies, broadly categorized into 2D/3D, and discuss their influence on the global problem framing. Following this, we study SSC deep architectures in Section 4.4.2. While initially, the task was addressed with vanilla 3D CNNs, other architectures have been leveraged such as 2D/3D CNNs, point-based network, or various hybrid proposals. Section 4.4.3 presents important design choices, such as contextual aggregation which greatly influences any geometrical task like SSC, or lightweight architectures to leverage the burden of 3D networks spanning from compact 3D representations to custom convolutions. We discuss the training strategies in Section 4.4.4, along with the losses and their benefits. Finally, a grouped evaluation of the metrics, methods performance, and network efficiency is in Section 4.4.5.

We provide the reader with a digest overview of the field, chronologically listing methods in Table 4.2 – where columns follow the paper structure. Because SSC definition may overlaps with some *reconstruction methods* that also address semantics, we draw the inclusion line in that SSC must also complete semantics *and* geometry of *unseen* regions. We in-distinctively report as SSC any method meeting these criteria. Looking at Table 4.2, it illustrates both the growing interest for the task and the lack of consensus.

4.4.1 Input encoding

Given the 3D nature of the task, there is an evident benefit of using 3D inputs as it already withholds geometrical insights. As such, it is easier to leverage sparse 3D surface as input in the form of occupancy grid, distance fields or point clouds. Another line of work uses RGB data in conjunction with depth data since they are spatially aligned and easily handled by 2D CNNs.

		Input Encoding (Section 4.4.1)						Architecture (Section 4.4.2)	Design choices (Section 4.4.3)					Training (Section 4.4.4)						Evaluation (Section 4.4.5)				Open source	
		2D			3D																				
		RGB	Depth/Range/HHA	Other (segs, normals, etc.)	Occ. Grid	TSDF	Point Cloud	Network type	Contextual Awareness	Position Awareness	Fusion Strategies	Lightweight Design	Refinement	End-to-end	Coarse-to-fine	Two stage	Multi-scale	Adversarial	Losses	NYU[b]	SUNCG[c]	SemanticKITTI	Other	Framework	Weights
2017	SSCNet (Song et al., 2017)[a]					✓		volume	DC					✓					CE	✓	✓	✓		Caffe	✓
2018	Guedes et al. (2018)	✓				✓		volume	DC		E			✓					CE	✓				-	
	ScanComplete (Dai et al., 2018)					✓		volume				GrpConv			✓				ℓ_1 CE		✓		✓	TF	✓
	VVNet (Guo and Tong, 2018)		✓	✓				view-volume	DC		E			✓					CE	✓	✓			TF	✓
	Cherabier et al. (2018)	✓				✓		volume	PDA		E	MSO		✓	✓		✓		CE				✓	-	
	VD-CRF (Zhang et al., 2018b)					✓		volume	DC				✓	✓					CE	✓	✓			-	
	ESSCNet (Zhang et al., 2018a)					✓		volume				GrpConv Sparse					✓		CE	✓	✓			PyTorch	✓
	ASSCNet (Wang et al., 2018d)		✓					view-volume	Mscale. CAT										CE	✓	✓			TF	
	SATNet (Liu et al., 2018)	✓	✓					view-volume	ASPP		M/L					✓			CE	✓	✓			PyTorch	✓
2019	DDRNet (Li et al., 2019)	✓	✓					view-volume	LW-ASPP		M	DDR		✓					CE	✓	✓			PyTorch	✓
	TS3D (Garbade et al., 2019)[a]	✓			✓			hybrid	DC		E					✓			CE	✓	✓	✓		-	
	EdgeNet (Dourado et al., 2020a)	✓				✓		volume	DC	✓	M			✓					CE	✓	✓			-	
	SSC-GAN (Chen et al., 2019b)					✓		volume	DC									✓	BCE CE	✓	✓			-	
	TS3D+RNet (Behley et al., 2019)		✓		✓			hybrid	DC		E					✓			CE			✓		-	
	TS3D+RNet+SATNet (Behley et al., 2019)		✓		✓			hybrid	DC		E					✓			CE			✓		-	
	ForkNet (Wang et al., 2019c)					✓		volume	DC									✓	BCE CE	✓	✓			TF	✓
	CCPNet (Zhang et al., 2019b)					✓		volume	CCP DC			GrpConv	✓	✓					CE	✓	✓			-	
	AM^{2}FNet (Chen et al., 2019a)	✓				✓		hybrid	DC	✓	M		✓	✓					BCE CE	✓				-	
2020	GRFNet (Liu et al., 2020)	✓	✓					view-volume	LW-ASPP DC		M	DDR		✓					CE	✓	✓			-	
	Dourado et al. (2020b)	✓				✓		volume	DC	✓	E		✓	✓					CE	✓				-	
	AMFNet (Li et al., 2020c)	✓	✓					view-volume	LW-ASPP		L	RAB				✓			CE	✓	✓			-	
	PALNet (Li et al., 2020b)		✓			✓		hybrid	FAM DC	✓	M			✓					PA	✓	✓			PyTorch	✓
	3DSketch (Chen et al., 2020a)	✓				✓		hybrid	DC	✓	M	DDR				✓		✓	BCE CE CCY	✓	✓			PyTorch	✓
	AIC-Net (Li et al., 2020a)	✓	✓					view-volume	FAM AIC		M	Anisotropic		✓					CE	✓	✓			PyTorch	✓
	Wang et al. (2020a)				✓			volume				Octree-based		✓					BCE CE		✓			-	
	L3DSR-Oct (Wang et al., 2019a)					✓		volume				Octree-based			✓				BCE CE		✓	✓	✓	-	
	IPF-SPCNet (Zhong and Zeng, 2020)	✓					✓	hybrid			E					✓			CE	✓				-	
	Chen et al. (2020b)					✓		volume	GA Module					✓					BCE CE	✓	✓			-	
	LMSCNet (Roldão et al., 2020)				✓			view-volume	MSFA			2D		✓			✓		CE	✓		✓		PyTorch	✓
	SCFusion (Wu et al., 2020)				✓			volume	DC				✓					✓	CE			✓	✓	-	
	S3CNet (Cheng et al., 2020)		✓	✓		✓		hybrid			L	Sparse	✓	✓					BCE CE PA			✓		-	
	JS3C-Net (Yan et al., 2021)				✓			volume				Sparse				✓			CE			✓		PyTorch	✓
	Local-DIFs (Rist et al., 2020a)						✓	point-based						✓					BCE CE SCY			✓		-	

[a] These original works were significantly extended in Guo and Tong (2018), Behley et al. (2019), or Roldão et al. (2020). [b] Includes both NYUv2 (Silberman et al., 2012), NYUCAD (Firman et al., 2016). [c] Includes both SUNCG (Song et al., 2017), SUNCG-RGBD (Liu et al., 2018). [d] Includes both ScanNet (Dai et al., 2017a), CompleteScanNet (Wu et al., 2020). **Fusion Strategies** - E, Early. M, Middle. L, Late.

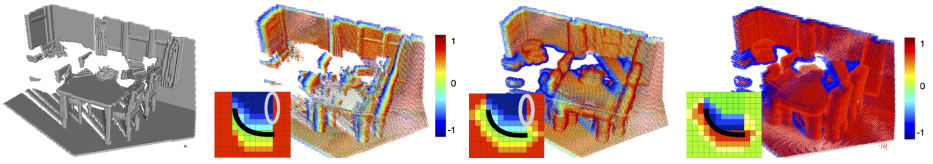
Contextual Awareness - DC, Dilated Convolutions. (LW)-ASPP, (Lightweight) Atrous Spatial Pyramid Pooling. CCP, Cascaded Context Pyramid. FAM, Feature Aggregation Module. AIC, Anisotropic Convolutional Module. GA, Global Aggregation. MSFA, Multi-scale Feature Aggregation. PDA, Primal-Dual Algorithm. **Lightweight Design** - GrpConv, Group Convolution. DDR, Dimensional Decomposition Residual Block. RAB, Residual Attention Block. MSO, MultiScale Optimization. **Losses** - Reconstruction: BCE, Binary Cross Entropy. ℓ_1, L1 norm. Semantic: CE, Cross Entropy. PA, Position Awareness. Consistency: CCY, Completion Consistency. SCY, Spatial Semantics Consistency.

Table 4.2: Semantic Scene Completion (SSC) methods.

3D grid-based. In most works, 3D occupancy grid (aka Voxel Occupancy) is used (Roldão et al., 2020; Garbade et al., 2019; Yan et al., 2021; Wu et al., 2020), encoding each cell as either *free* or *occupied*. Such representation is conveniently encoded as binary grids, easily compressed (cf. Section 4.4.3.4), but provides little input signal for the network. An alternative richer encoding is the use of TSDF (Figure 4.6c), where the signed distance d to the closest surface is computed at given 3D locations (usually, voxel centers), as in Dai et al. (2020, 2018); Chen et al. (2020a); Cherabier et al. (2018). Instead of only providing input signal at the measurement locations like occupancy grids or point cloud, TSDF provides a richer supervision signal for the network. The sign for example provides guidance on which part of the scene is occluded in the input. The greedy 3D computation can be avoided using projective TSDF (p-TSDF, cf. Figure 4.6b) which only computes the distance along the sensing path (Newcombe et al., 2011), but with the major drawback of being view-dependent. Highlighted by Song et al. (2017), another limitation of TSDF or p-TSDF lies in the strong gradients being in the free space area rather than close to the surface where the networks needs guidance. This is noticeable in Figure 4.6b, 4.6c since the red/blue gradients are afar from the surface. To move strong gradients closer to the surface, they introduced flipped TSDF (f-TSDF, Figure 4.6d) such that f-TSDF $= sign(\text{TSDF})(d_{\max} - d)$ with $d_{\max}$ the occlusion boundary, showing improvement in both network guidance and performance (Song et al., 2017). However, the field yet lacks thorough study on the benefit of TSDF encoding. While f-TSDF is still commonly used in recent SSC (Zhang et al., 2018a,b; Dourado et al., 2020a,b; Zhang et al., 2019b; Li et al., 2020b; Chen et al., 2019a) or SC (Denninger and Triebel, 2020), other approaches stick with original TSDF (Dai et al., 2018; Zhang et al., 2018b; Chen et al., 2020a; Wang et al., 2019c). Furthermore, the benefit of f-TSDF over TSDF is questioned in Zhang et al. (2018b); Garbade et al. (2019) with experiments showing it may slightly harm the performance or bring negligible improvemsent. Except for the lighter p-TSDF, all TSDF-variants require high computation times and hinder real time implementations.

3D point cloud. Despite the benefit of such representation, only three recent SSC works (Zhong and Zeng, 2020; Rist et al., 2020a,b) use point cloud as input encoding. In Zhong and Zeng (2020), RGB is used in conjunction to augment point data with RGB features, whereas Rist et al. (2020a,b) reproject point features in the top-view space. Despite the few SSC methods using points input, it is commonly used for object completion (Yuan et al., 2018; Huang et al., 2019; Xie et al., 2020a; Wang et al., 2020b).

2D representations. Alternatively, depth maps or range images (i.e. 2D polar-encoded LiDAR data) are common 2D representations storing geo-

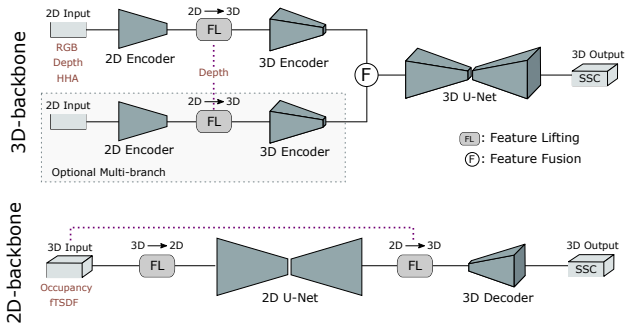


(a) Surface (b) Projective TSDF (c) TSDF (d) Flipped TSDF

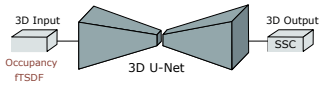
Figure 4.6: **TSDF variants**. Projective TSDF (b) is fast to obtain but view-dependent. TSDF (c) (Dai et al., 2018; Chen et al., 2019b; Wang et al., 2019c, 2020a; Chen et al., 2020b) solves the view dependency but gradient is stronger at farther areas from the surface, being inadequate for learning-based methods. In contrast, f-TSDF (d) (Song et al., 2017; Zhang et al., 2018a; Dourado et al., 2020a; Zhang et al., 2019b; Cheng et al., 2020; Dourado et al., 2020b; Chen et al., 2019a) has strongest gradient near the surface. (Source: Song et al. (2017)).

metrical information, therefore suitable candidates for the SSC task. Indeed many works (Behley et al., 2019; Cheng et al., 2020; Guo and Tong, 2018; Li et al., 2020a,b,c, 2019; Lin et al., 2013; Liu et al., 2018; Wang et al., 2018d) used either representation alone or in conjunction with other modalities. Opposite to point cloud but similarly to grid representation, such encoding enables meaningful connexity of the data and cheap processing with 2D CNNs. Some works propose to transform depth into an HHA image (Gupta et al., 2014), which keeps more effective information when compared to the single channel depth encoding (Guo and Tong, 2018; Liu et al., 2018). Additional non-geometrical modalities such as RGB or LiDAR refraction intensity provide auxiliary signals specifically useful to infer semantic labels (Behley et al., 2019; Garbade et al., 2019; Liu et al., 2018; Li et al., 2019; Liu et al., 2020; Li et al., 2020c; Chen et al., 2020a; Zhong and Zeng, 2020; Cherabier et al., 2018). In practice, some works have shown that good enough semantic labels can be inferred directly from depth/range images (Behley et al., 2019; Yan et al., 2021; Cherabier et al., 2018) to guide SSC.

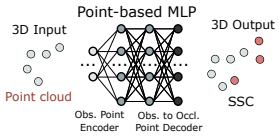
Interestingly, the vast majority of the literature only accounts for surface information while ignoring any free space data provided by the sensors (see Figure 4.2). While *free space* labels might be noisy, we believe it provides an additional signal for the network which was found beneficial in Rist et al. (2020a); Wu et al. (2020). Conversely, we highlight that encoding *unknown* information brings little improvement given experiments performed during the development of our LMSCNet network (Roldão et al., 2020) to be presented in next chapter.



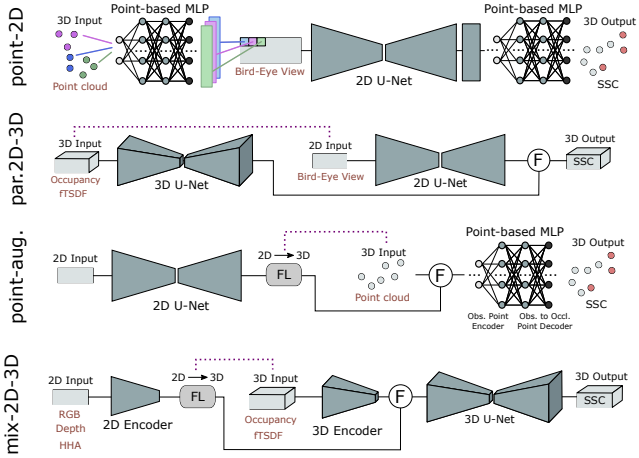
(a) View-Volume Networks



(b) Volume Networks



(c) Point-based Networks



(d) Hybrid Networks

(a) View-Volume Nets., 3D-backbone: (Guo and Tong, 2018; Liu et al., 2018; Li et al., 2019; Liu et al., 2020; Li et al., 2020c,a), 2D-backbone: (Roldão et al., 2020).

(b) Volume Nets.: (Song et al., 2017; Guedes et al., 2018; Zhang et al., 2018b,a; Dourado et al., 2020a; Chen et al., 2019b; Wang et al., 2019c; Zhang et al., 2019b; Wang et al., 2020a; Chen et al., 2020b; Yan et al., 2021; Dourado et al., 2020b; Wu et al., 2020; Cherabier et al., 2018; Dai et al., 2018; Wang et al., 2019a)

(c) Point-based Nets.: (Zhong and Zeng, 2020)

(d) Hybrid Nets., point-2D: (Rist et al., 2020a,b), parallel-2D-3D: (Cheng et al., 2020), point-augmented: (Zhong and Zeng, 2020), mix-2D-3D: (Garbade et al., 2019; Behley et al., 2019; Li et al., 2020b; Chen et al., 2020a)

Figure 4.7: **Architectures for SSC.** For compactness we do not display all connections but rather focus on the global architectures and information exchanges between the different networks. Ⓕ stands for any type of fusion.

4.4.2 Architecture choices

Directly linked to the choice of input encoding, we broadly categorize architectural choices in 4 groups. In details: *Volume networks* leveraging 3D CNNs to convolve volumetric grid representations, *Point-based networks* which compute features on points locations, *View-Volume networks* that learn the 2D-3D mapping of data with 2D and 3D CNNs, and *Hybrid networks* that use various network to combine modalities of different dimension. All architectures output $N \times C$ data (N the spatial dimensions and C the number of semantic classes) where the last dimension is the probability of either semantic class at the given location. In most works, the final prediction is the softmax output along the classes probabilities. We refer to Figure 4.7 for a visual illustration of either architecture type.

Volume networks. As they are convenient for processing grid data 3D CNNs (Figure 4.7b) are the most popular for SSC (Song et al., 2017; Guedes et al., 2018; Zhang et al., 2018b,a; Dourado et al., 2020a; Chen et al., 2019b; Wang et al., 2019c; Zhang et al., 2019b; Wang et al., 2020a; Chen et al., 2020b; Yan et al., 2021; Dourado et al., 2020b; Wu et al., 2020; Cherabier et al., 2018; Dai et al., 2018; Wang et al., 2019a). Since completion heavily requires contextual information it is common practice to use a UNet architecture (Ronneberger et al., 2015) (see Figure 4.7), i.e. encoder-decoder with skip connections. The benefit of the latter is not only to provide contextual information but also to enable meaningful coarser scene representation, used in SSC for outputting multi-scale predictions (Roldão et al., 2020; Zhang et al., 2018a) or for enabling coarse-to-fine refinement (Dai et al., 2018).

There are two important limitations of 3D CNNs: their cubically growing memory need, and the dilation of the sparse input manifold due to convolutions. To circumvent both, one can use sparse 3D networks like SparseConvNet (Graham et al., 2018; Zhang et al., 2018a) or Minkowski Network (Choy et al., 2019) which conveniently operate only on input geometry, thus allowing high grid resolution where each cell contains typically a single point. While they were found highly beneficial for most 3D tasks, they show limited interests for SSC since the output is expected to be more dense than the task input. Considering the output to be *sparse*, Dai et al. (2020) used a sparse encoder and a custom sparse generative decoder to restrict the manifold dilation, applied for SC rather than SSC. This is beneficial but cannot cope with large chunks of missing data. An alternative is Yan et al. (2021) that first performs pointwise semantic labeling using a sparse network. To enable a *dense* SSC output in Zhang et al. (2018a), authors merge the output of multiple *shared* SparseConvNets applied on sub-sampled non-overlapping sparse grids. Both Zhang et al. (2018a); Dai et al. (2020) are clever use of sparse convolutions but somehow limit the memory and computational benefit of the latter. In Cherabier et al. (2018),

variational optimization is used to regularize the model and avoid the need of greedy high-capacity 3D CNN.

View-volume networks. To take advantage of 2D CNNs efficiency, a common strategy is to use them in conjunction with 3D CNNs as in Guo and Tong (2018); Liu et al. (2018); Li et al. (2020a, 2019); Liu et al. (2020); Roldão et al. (2020); Li et al. (2020c), see Figure 4.7a. Two different schemes have been identified from the literature. The most common, as in Guo and Tong (2018); Liu et al. (2018); Li et al. (2020a, 2019); Liu et al. (2020); Li et al. (2020c), employs a 2D CNN encoder to extract 2-dimensional features from 2D texture/geometry inputs (RGB, depth, etc.), which are then lifted to 3D and processed by 3D CNN (Figure 4.7a, 3D-backbone). Optional modalities may be added with additional branches and mid-fusion scheme (Liu et al., 2018; Li et al., 2020a, 2019; Liu et al., 2020). In brief, the use of sequential 2D-3D CNNs conveniently benefits of different neighboring definitions, since 2D neighboring pixels might be far away in 3D and vice versa, thus providing rich feature representation, at the cost of increased processing. To address this limitation, the second scheme (Figure 4.7a, 2D-backbone) projects 3D input data into 2D, then processed with normal 2D CNN (Roldão et al., 2020) significantly lighter than its 3D counterpart. The resulting 2D features are then lifted back to 3D and decoded with 3D convolutions, retrieving the third dimension before the final prediction. This latter scheme is irrefutably lighter in computation and memory footprint but better suited for outdoor scenes (cf. Section 4.4.5), as the data exhibits main variance along two axes (i.e. longitudinal and lateral).

Point-based networks. To ultimately prevent discretization of the input data, a few recent works (Zhong and Zeng, 2020; Rist et al., 2020a,b) employs point-based networks, see Figure 4.7c. In 2018, Yuan et al. (2018) first proposed to apply permutation-invariant architecture (Qi et al., 2017b) to object completion with promising results. However, its use for SSC was hindered by the limited points capacity, the need of fixed size output, and the use of global features extraction. To date, only SPC-Net (Zhong and Zeng, 2020) relies solely on point-based network – $\mathcal{X}$Conv (Li et al., 2018) – to predict the semantics of observed and unobserved points. The fixed size limitation is circumvented by assuming regular distribution of unobserved points. Overall, we believe point-based SSC has yet attracted too few works, and is a viable avenue of research.

Hybrid networks. A number of other works combine architectures already mentioned, which we refer as *hybrid networks* (Rist et al., 2020a,b; Cheng et al., 2020; Zhong and Zeng, 2020; Garbade et al., 2019; Behley et al., 2019; Li et al., 2020b; Chen et al., 2020a), see Figure 4.7d. A com-

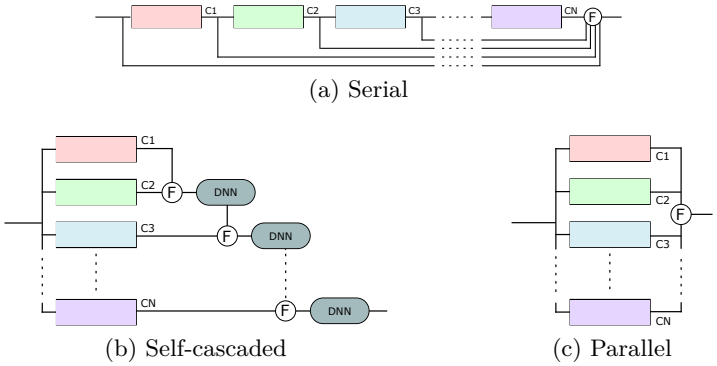
mon 2D-3D scheme combines 2D and 3D features (e.g. RGB and f-TSDF) through expert modality networks in a common latent space decoded via a 3D CNN (Garbade et al., 2019; Chen et al., 2020a) (Figure 4.7d, mix-2D-3D). This enables additional benefit with the combined use of texture and geometrical features. Similarly, IPF-SPCNet (Zhong and Zeng, 2020) performs semantic segmentation from RGB image on an initial 2D CNN and lifts image labels to augment ad-hoc 3D points (Figure 4.7d, point-augmented). In Rist et al. (2020a,b), PointNet (Qi et al., 2017b) encodes geometrical features from sub set of points latter convolved in a bird eye view (BEV) projection with 2D CNN in a hybrid architecture manner (Figure 4.7d, point-2D). Another strategy employs parallel 2D-3D branches to process the same data with different neighborhoods definition contained in the 2D projected image and 3D grid as in Li et al. (2020b). Recently, S3CNet (Cheng et al., 2020) combines 3D voxelized f-TSDF and normal features with a 2D BEV (Chen et al., 2017), passing both modalities through sparse encoder-decoder networks for late fusion (Figure 4.7d, parallel-2D-3D), achieving impressive results in outdoor scenes. A similar architecture is proposed by Abbasi et al. (2018) to perform what they refer to as *scene extrapolation* (Song et al., 2018), by performing extrapolation of a half-known scene into a complete one.

4.4.3 Design choices

The significant sparsity difference between the input data and the expected *dense* output imposes special design choices to be made, specifically to ensure efficient flow of information. In the following, we elaborate on the most important ones such as contextual awareness (Section 4.4.3.1), position awareness (Section 4.4.3.2) and fusion strategies (Section 4.4.3.3). Finally, we detail lightweight designs (Section 4.4.3.4) to efficiently process 3D large extent of sparse data, and the common refinement processes (Section 4.4.3.5).

4.4.3.1 Contextual awareness

To correctly complete missing information in the scene, it is necessary to gather contextual information from multiple scales, which enables to disambiguate between similar objects present in the scene. This makes possible to capture both local geometric details and high-level contextual information (Song et al., 2017). A common strategy to accomplish this is to add skip connections between different convolutional layers to aggregate features from different receptive fields (Song et al., 2017; Roldão et al., 2020; Zhang et al., 2018a; Guo and Tong, 2018; Garbade et al., 2019; Liu et al., 2018; Chen et al., 2019b; Dourado et al., 2020a; Zhang et al., 2019b; Chen et al., 2020a; Dai et al., 2020; Dourado et al., 2020b). Additionally, serial context



(a) Serial

(b) Self-cascaded

(c) Parallel

Figure 4.8: **Multi-scale contextual aggregation.** While context is indubitably important for SSC, different strategies are used to aggregate features from various spatial/scale contexts. Color blocks stand for convolutions with different dilation rates. Ⓕ stands for any type of fusion.

aggregation with multi-scale feature fusion can be used as proposed in Li et al. (2020b), Figure 4.8a. In Song et al. (2017), the use of dilated convolutions (aka 'atrous') (Yu and Koltun, 2016) are proposed to increase receptive fields and gather context information at low computational cost. The strategy became popular among most works (Song et al., 2017; Guo and Tong, 2018; Garbade et al., 2019; Liu et al., 2018; Wang et al., 2019c; Dourado et al., 2020a,b; Zhang et al., 2019b; Li et al., 2020c; Chen et al., 2020a; Li et al., 2020b; Roldão et al., 2020; Wu et al., 2020). Such convolutions are only suitable for dense networks (as opposed to sparse networks), and even then should only be applied in deeper layers of the network after dilation of the input manifold. In Liu et al. (2018), a feature aggregation module is introduced by using Atrous Spatial Pyramid Pooling blocks (ASPP) (Chen et al., 2018), which exploits multi-scale features by employing multiple parallel filters with different dilation rates, Figure 4.8c. A lightweight ASPP is presented in Li et al. (2019). Dilated convolutions in the ASPP module can be replaced by Residual dilated blocks (He et al., 2016) to increase spatial context and improve gradient flow. A Guided Residual Block (GRB) to amplify fused features and a Global Aggregation module to aggregate global context through 3D global pooling are proposed in Chen et al. (2020b). An additional feature aggregation strategy is presented in Zhang et al. (2019b), where multi-context aggregation is achieved by a cascade pyramid architecture, Figure 4.8b. In Cherabier et al. (2018) multi-scale features are aggregated together following a Primal-Dual optimization algorithm (Pock and Chambolle, 2011), which ensures semantically stable predictions and further acts as a regularizer for the learning.

4.4.3.2 Position awareness

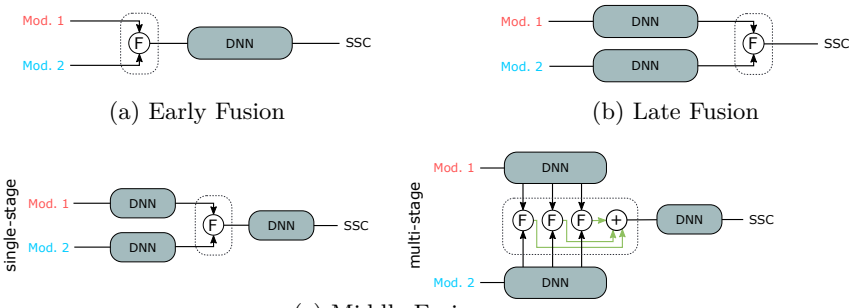
Geometric information contained in voxels at different positions has high variability, i.e. *Local Geometric Anisotropy*. In particular, voxels inside an object are homogeneous and likely to belong to a same semantic category as their neighbors. Conversely, voxels at the surface, edges and vertices of the scene provide richer geometric information due to the higher variance of their surroundings. To deal with this, PALNet (Li et al., 2020b) proposes a Position Aware loss (cf. Section 4.4.4.1), which consists in a Cross Entropy loss with individual voxel weights assigned according to their geometric anisotropy, providing slightly faster convergence and improving results. Likewise, AM^{2}FNet (Chen et al., 2019a) supervises contour information by an additional cross entropy loss as supplementary cue for segmentation.

In the same line of work, EdgeNet (Dourado et al., 2020a) calculates Canny edges (Canny, 1986) then fused with a f-TSDF obtained from the depth image. Hence, it increases gradient along the geometrical edges of the scene. Additionally, detection of RGB edges enables to identify objects lacking geometrical saliency. The same network is used in Dourado et al. (2020b) to predict complete scenes from panoramic RGB-D images.

Similarly, Chen et al. (2020a) introduces an explicit and compact geometric embedding from depth information by predicting a 3D sketch containing scene edges from an input TSDF. They show that this feature embedding is resolution-insensitive, which brings high benefit, even from partial low-resolution observations.

4.4.3.3 Fusion strategies

SSC requires to output both geometry and semantics. Though highly coupled – geometry helping semantics and vice-versa –, there is a natural benefit to use inputs of different natures for example to provide additional texture or geometry insights. We found that about two third of the literature uses multi-modal inputs though it appears less trendy in most recent works (see Table 4.2 col 'Input'). For the vast majority of multi-input works, RGB is used along various geometrical input (Guedes et al., 2018; Liu et al., 2018; Li et al., 2019; Garbade et al., 2019; Dourado et al., 2020a; Behley et al., 2019; Chen et al., 2019a; Dourado et al., 2020b; Liu et al., 2020; Li et al., 2020c; Chen et al., 2020a; Li et al., 2020a; Zhong and Zeng, 2020) as it is a natural candidate for semantics. Even without color, fusing 2D and 3D modalities is often employed because it enables richer features estimation. This is because 2D and 3D neighborhood differ, since 2D data results of planar projection along the optical axis of the sensor. Subsequently, a common strategy consists in fusing geometrical features processed with different 2D / 3D encoding to obtain richer local scene descriptors. In Behley et al. (2019) depth and occupancy are fused while Li et al. (2020b) uses depth along



(a) Early Fusion (Guedes et al., 2018; Guo and Tong, 2018; Garbade et al., 2019; Zhong and Zeng, 2020; Dourado et al., 2020b; Behley et al., 2019; Cherabier et al., 2018).

(b) Late Fusion (Li et al., 2020c; Liu et al., 2018; Cheng et al., 2020).

(c) Middle Fusion, single-stage: (Liu et al., 2018; Dourado et al., 2020a; Li et al., 2020b; Chen et al., 2020a), multi-stage.: (Li et al., 2019; Liu et al., 2020; Li et al., 2020a; Chen et al., 2019a).

Figure 4.9: **Fusion Strategies.** To accommodate for multiple input modalities (Mod. 1, Mod. 2), several fusion strategies are found in the literature. Here, Ⓕ stands for fusion and might be any type of fusion like concat Ⓒ, sum ⊕, multiply ⊗, convolutions, etc.

TSDF-like data. As mentioned earlier (cf. Section 4.4.1), TSDF provides a gradient field easing the network convergence. Finally, application-oriented fusion is also found such as fusing bird eye view along geometrical inputs as in Cheng et al. (2020) – which is better suited for outdoor SSC.

We group the type of fusion in threefold, shown in Figure 4.9. Fusion applied at the input level (*Early fusion*), at the mid-level features (*Middle fusion*), or at the late/output level (*Late fusion*). They are respectively referred as *E*, *M*, and *L* in column 'Fusion strategies' Table 4.2.

Early fusion. The most trivial approach is to concatenate inputs modalities (Guedes et al., 2018; Guo and Tong, 2018; Garbade et al., 2019; Zhong and Zeng, 2020; Dourado et al., 2020b; Behley et al., 2019; Cherabier et al., 2018) before any further processing, see Figure 4.9a. There are two strategies here: when spatially aligned (e.g. RGB/Depth) inputs can be concatenated channel wise; alternatively inputs can be projected to a shared 3D common space (*aka* features lifting). For spatially aligned modalities, it is common to use pairs of normals/depth (Guo and Tong, 2018) or RGB/semantics (Behley et al., 2019) and to process them with 2D CNNs. The second strategy lifts any 2D inputs to 3D – assuming depth information and accurate inter-sensors calibration – and process it with 3D networks. This has been done with RGB/depth (Guedes et al., 2018), depth/semantics (Garbade et al., 2019; Cherabier et al., 2018), points/semantics (Zhong and Zeng, 2020). Ex-

cept when using points, this second strategy leads to a sparse tensor since not all 3D cells have features. Noteworthy, Garbade et al. (2019); Behley et al. (2019); Zhong and Zeng (2020); Cherabier et al. (2018) use semantics, which is first estimated either from RGB or depth-like data. A 2D or 3D network processes the concatenated tensor, and while it logically outperforms single-modality input (Guo and Tong, 2018; Dourado et al., 2020a; Garbade et al., 2019) there seems to be little benefit to apply early fusion.

Middle fusion. To exploit all modalities, middle fusion uses expert networks that learn modality-centric features. A straightforward fusion strategy is employed in Chen et al. (2020a); Dourado et al. (2020a); Li et al. (2020b); Liu et al. (2018) where the features are simply concatenated and processed with a UNet style architecture (cf. Figure 4.9c, single-stage), which improves over early fusion but still limits the exchange of information between modalities. The information flow is improved in Li et al. (2019); Liu et al. (2020); Li et al. (2020a); Chen et al. (2019a) by fusing modality-centric features in a multi-stage manner (cf. Figure 4.9c, multi-stage); meaning that low-level features from different modalities are fused together and aggregated with fused mid/high level features gathered similarly. While ultimately the number of fusion stages shall be function of the input/output size, 3 stages are often used (Li et al., 2019, 2020a; Chen et al., 2019a), though Liu et al. (2020) claims 4 stages boost performances with similar input/output. The fused mechanism can be a simple summation (Li et al., 2019) or concatenation (Li et al., 2020a), but Chen et al. (2019a); Liu et al. (2020) benefit from smarter selective fusion schemes using respectively RefineNet (Lin et al., 2017) and Gated Recurrent Fusion.
Overall, the literature consensus is that middle fusion is highly efficient for SSC. The ablation studies of Liu et al. (2020) reports that any selective fusion schemes brings at least a 20% performance boost over simple sum/concat/max schemes.

Late fusion. Few works use late fusion for SSC (Li et al., 2020c; Cheng et al., 2020; Liu et al., 2018), see Figure 4.9b. The straightforward strategy in Li et al. (2020c) is to apply fusion (namely, element-wise multiplication) of two SSC branches (a 3D guidance branch, and a semantic completion branch), followed by a softmax. The benefit still appears little (5 to 10%) given the extra computational effort. Similarly, color and geometry branches are concatenated and shallowly convolved before softmax in Liu et al. (2018), also providing small benefit (less than 3%). A unique strategy was proposed in the recent S3CNet (Cheng et al., 2020) where the output of parallel 2D top-view *and* 3D SSC are fused together in a semantic-wise manner. While it was only evaluated on outdoor scenes – which setup naturally minimizes vertically overlapping semantic labels – ablation reports an overall 20% boost.

Summarizing the different strategies, *Middle fusion* appears to be the best general SSC practice, though *Late fusion* was found beneficial in some specific settings. On fused modalities, RGB/geometry fusion boosts performance but at the cost of an additional sensor need, but even using fusion of geometrical input with different encodings is highly beneficial. An interesting insight from Dourado et al. (2020a); Chen et al. (2020a) advocates that RGB *or* geometry can be fused with edges features as they provide additional boundaries guidance for the SSC network.

4.4.3.4 Lightweight designs

A few optimization techniques are often applied for lightweight SSC architectures with the aim of addressing two separate problems: how to improve the memory or computation efficiency, and how to design meaningful convolutions to improve the information flow. We detail either problem and its solutions below.

Memory and computation efficiency. Voxel grids are often used as input/output encoding of the 3D data since current datasets provide ground truth in such format. However only a tiny portion of the voxels are occupied which makes naive *dense* grid inefficient in memory and computation. Memory wise, a few works use compact hierarchical 3D representation inspired from pre-deep learning, like Kd-Tree (Bentley, 1975) and Octree (Meagher, 1982). Octree-based deep networks are often used for learning object reconstruction (Wang et al., 2017; Riegler et al., 2017a,b; Wang et al., 2018b) though little applied on real semantic scene completion problem (Cherabier et al., 2018; Wang et al., 2019a, 2020a). Meanwhile, deep Kd-Networks (Klokov and Lempitsky, 2017) proposal seems less appropriate and has not yet been applied to SSC. Computation wise, (Cherabier et al., 2018) proposed a custom network architecture with adjustable multi-scale branches which inference and backpropagation can be ran in parallel, subsequently enabling faster training and good performance with low-capacity dense 3D CNNs. Alternatively, few SSC works (Zhang et al., 2018a; Dai et al., 2020) use sparse networks like SparseConvNet (Graham et al., 2018) or Minkowski (Choy et al., 2019) which operate only in active locations through a hash table. While sparse convolutions are very memory/computation efficient, they are less suitable for completion, since they deliberately avoid filling empty voxels to prevent dilation of the input domain. To remedy this for SSC task, dense convolutions are still applied in the decoder, which subsequently reduces sparse networks efficiency. Overall, while Kd/Octree networks are highly memory efficient, the complexity of their implementation have restricted a wider application. Contrastingly, sparse networks (Graham et al., 2018; Choy et al., 2019) are more used (Zhang et al., 2018a; Cheng et al., 2020; Yan et al., 2021; Dai et al., 2020).

Efficient convolutions. A key observation is the spatial redundancy of data, since neighboring voxels contain similar information. To exploit such redundancy, Zhang et al. (2018a) proposes Spatial Group Convolutions (SGC) to divide input volume into different sparse tensors along the spatial dimensions which are then convolved with shared sparse networks. A similar strategy is followed by Dai et al. (2018), dividing the volumetric space into a set of eight interleaved voxel groups and performing an auto-regressive prediction (Reed et al., 2017). Dilated convolutions are also widely used for semantic completion methods (Song et al., 2017; Guo and Tong, 2018; Garbade et al., 2019; Liu et al., 2018; Wang et al., 2019c; Chen et al., 2019b; Li et al., 2019; Dourado et al., 2020a; Zhang et al., 2019b; Li et al., 2020c; Chen et al., 2020a; Li et al., 2020b; Liu et al., 2020; Dourado et al., 2020b; Roldão et al., 2020; Chen et al., 2020b), since they increase receptive fields at small cost, providing large context, which is crucial for scene understanding as discussed in Section 4.4.3.1. Dilated convolutions with *separated kernels* are proposed in Zhang et al. (2019b) by separating the input tensor into subvolumes. This enables to reduce the number of parameters and consider depth profiles in which depth values are continuous only in neighbouring regions.
DDRNet (Li et al., 2019) also introduces Dimensional Decomposition Residual (DDR) block, decomposing 3D convolutions into three consecutive layers along each dimension, subsequently reducing the networks parameters. In Li et al. (2020a), this concept is extended with use of anisotropic convolutions, where the kernel size of each 1D convolution is adaptively learnt during training to model the dimensional anisotropy.

4.4.3.5 Refinement

Refinement is commonly used in many vision tasks, but little applied in SSC. VD-CRF (Zhang et al., 2018b) extends SSCNet (Song et al., 2017) by applying Conditional Random Field (CRF) to refine output consistency, achieving little over 4% gain. Additionally, S3CNet (Cheng et al., 2020) presents a 3D spatial propagation network (Liu et al., 2017) to refine segmentation results after fusion of 2D semantically completed bird eye view image and 3D grid. In Zhang et al. (2019b), guided residual refinement is also applied where low level features are reintroduced at the refinement stage to boost preservation of fine-grained details, which boost results by about 12%. Additional partial refinement is applied in Dourado et al. (2020b); Wu et al. (2020) to fuse SSC predictions from different viewpoints, by either softmax applied to overlapping partitions (Dourado et al., 2020b; Wu et al., 2020) or an occupancy based fusion policy (Wu et al., 2020). Though little works address the refinement problem, some notable performance boost are found in the literature, thus being an encouraging topic to explore.

4.4.4 Training

We now detail the SSC training process, starting with the SSC losses (Section 4.4.4.1), and subsequently the implemented training strategies (Section 4.4.4.2).

4.4.4.1 Losses

We classify the SSC losses found in the literature in 3 broad categories: *reconstruction losses* which optimize geometrical accuracy, *semantics losses* which optimize semantics prediction, and *consistency losses* which guides the overall completion consistency. Note that other non-SSC losses are often added and that the type of SSC losses are commonly mixed – specifically, reconstruction+semantics (Dai et al., 2018; Wang et al., 2020a; Chen et al., 2020b; Cheng et al., 2020) or all three types (Chen et al., 2019b, 2020a; Wang et al., 2019c; Rist et al., 2020a,b). We refer to Table 4.2 for a quick overview of the losses used by each method.
In this section, we also refer to $\hat{y}$ as SSC prediction and y as ground truth, though for clarity we add subscript notation to distinguish between SSC encoding. For example, y^{mesh} corresponds to the ground truth mesh.

Reconstruction losses. These losses penalize the geometrical distance of the output $\hat{y}$ to ground truth y, in a self-unsupervised manner.

On occupancy grids outputs ($\hat{y}^{\text{occ}}$), **Binary Cross-Entropy** loss (BCE) is most often used (Rist et al., 2020a; Wang et al., 2020a; Chen et al., 2020b; Cheng et al., 2020; Rist et al., 2020b) to discriminate *free* voxels from *occupied*. Assuming a binary class mapping where all non-free semantic classes map to 'occupy'. It writes:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=0}^{N} \hat{y}_i^{\text{occ}} log(y_i^{\text{occ}}) - (1-\hat{y}_i^{\text{occ}}) log(1-y_i^{\text{occ}}) \,, \tag{4.1}$$

with N the number of voxels. The drawback of such loss is that it provides little guidance to the network due to its sparsity. Smoother guidance can be provided by outputting an implicit surface ($\hat{y}^{\text{SDF}}$) through minimization of the predicted signed distance values in $\hat{y}^{\text{SDF}}$ and corresponding SDF-encoded mesh (y^{SDF}) – using $\boldsymbol{\ell_1}$ or $\boldsymbol{\ell_2}$ norms.

On points outputs ($\hat{y}^{\text{pts}}$), the above losses can also be used to penalize distance to a ground truth mesh, though it is more common to apply points-to-points distances, thus assuming a ground truth point cloud (y^{pts}). To that end, permutation-invariant metrics are used. The **Chamfer Distance** (CD) (Fan et al., 2017) calculates the symmetrical closest point distance between two point clouds, such as:

$$\mathcal{L}_{\text{CD}} = \frac{1}{|y^{\text{pts}}|}\sum_{p\in y^{\text{pts}}} \min_{q\in\hat{y}^{\text{pts}}} \|p-q\|_2 + \frac{1}{|\hat{y}^{\text{pts}}|}\sum_{q\in\hat{y}^{\text{pts}}} \min_{p\in y^{\text{pts}}} \|q-p\|_2 \,. \tag{4.2}$$

The **Earth Mover's Distance** (EMD) Kurenkov et al. (2018); Fan et al. (2017) finds a bijection $\phi : y \rightarrow \hat{y}$ that minimizes average distance both sets, which is computationally expensive. It writes:

$$\mathcal{L}_{\text{EMD}} = \min_{\phi: y^{\text{pts}} \rightarrow \hat{y}^{\text{pts}}} \frac{1}{|y^{\text{pts}}|} \sum_{p \in y^{\text{pts}}} \|p - \phi(p)\|_2 \ . \tag{4.3}$$

Notice that standard EMD requires point sets to have same cardinality, which makes it a tough candidate for SSC. In fact, CD and EMD are commonly used for object completion tasks (Fan et al., 2017; Yuan et al., 2018) but have been little explored for SSC because of their computational greediness (Fan et al., 2017).

Semantic losses. Such losses are suitable for occupancy grids or points and can accommodate for either C classes (considering only semantics classes of occupied voxels or points) or $C + 1$ classes (considering all voxels/points and 'free space' being the additional class). Note that only the second case ($C + 1$ classes) enforce reconstruction, so the first one (C classes) would require additional *reconstruction losses*. **Cross-Entropy** loss (CE) is the preferred loss for SSC (Song et al., 2017; Roldão et al., 2020; Zhang et al., 2018a; Guo and Tong, 2018; Garbade et al., 2019; Liu et al., 2018; Li et al., 2019; Dourado et al., 2020a; Zhang et al., 2019b; Chen et al., 2019b; Li et al., 2020a,c; Cherabier et al., 2018), it models classes as independent thus considering the latter to be equidistant in the semantic space. Formally, supposing $(y, \hat{y})$ it writes:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{C} \sum_{i=0}^{N} \sum_{c=0}^{N} w_c \hat{y}_{i,c} \log \left(\frac{e^{y_{i,c}}}{\sum_{c'}^{C} e^{y_{i,c'}}} \right) , \tag{4.4}$$

assuming here that y is the one-hot-encoding of the classes (i.e. $y_{i,c} = 1$ if y_i label is c and otherwise $y_{i,c} = 0$). In practice, $(y, \hat{y})$ can be either occupancy grids $(y^{\text{occ}}, \hat{y}^{\text{occ}})$ or points $(y^{\text{pts}}, \hat{y}^{\text{pts}})$. A rare practice from Wang et al. (2019c) is to address classification with BCE (Equation 4.1) through the sum of C binary classification problems between each semantic class and the free class. However, such practice is unusual and arguably beneficial.

Recently, PALNet (Li et al., 2020b) proposed the **Position Aware** loss (PA), a weighted cross-entropy accounting for the local semantics entropy to encourage sharper semantics/geometric gradients in the completion (cf. Sec 4.4.3.2). The loss writes:

$$\mathcal{L}_{\text{PA}} = -\frac{1}{N} \sum_{i=0}^{N} \sum_{c=0}^{C} (\lambda + \alpha W_{\text{LGA}_i}) \hat{y}_{i,c}^{\text{occ}} \log \left(\frac{e^{y_{i,c}^{\text{occ}}}}{\sum_{c'}^{C} e^{y_{i,c'}^{\text{occ}}}} \right) , \tag{4.5}$$

with λ and α being simple base and weight terms, and W_{LGA_i} being the *Local Geometric Anisotropy* of i that scales accordingly to the semantic entropy in its direct vicinity (i.e. W_{LGA} lowers in locally smooth semantics

areas). We refer to Li et al. (2020b) for an in-depth explanation. From the latter, $\mathcal{L}_{\text{PA}}$ leads to small performance gain of 1–3%. Noteworthy, this loss could easily accommodate point clouds as well.

Note that *reconstruction* or *semantics* losses can only be computed on *known* ground truth location, due to the ground truth sparsity. Additionally, because SSC is a highly imbalanced problem (cf. Figure 4.3), class-balancing strategy is always used.

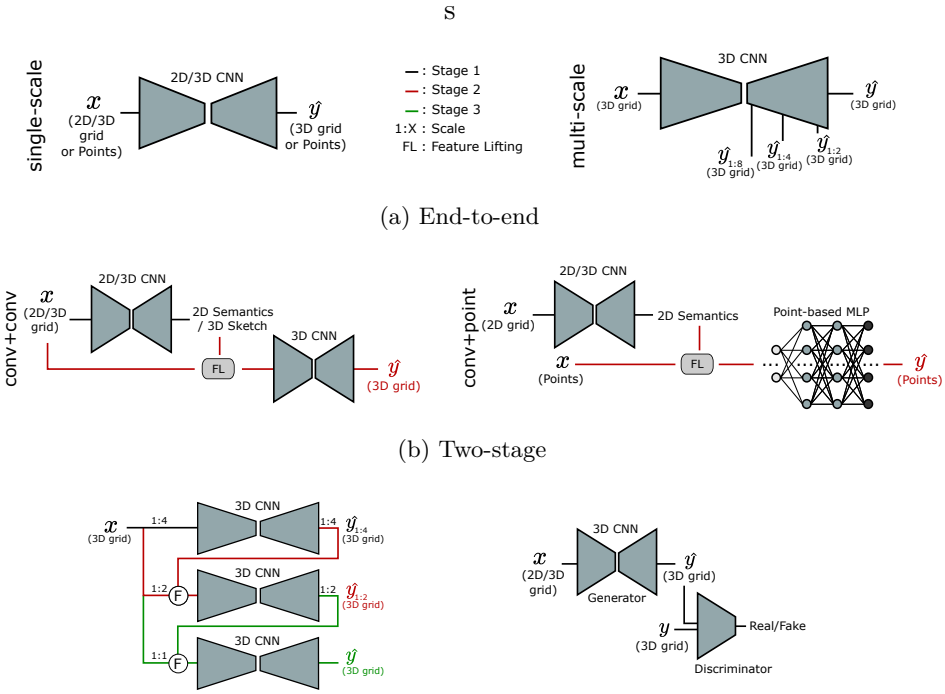
Consistency losses. Different from most *semantics* losses, these losses (Chen et al., 2020a; Rist et al., 2020a,b) provide a self-supervised semantic signal. In Chen et al. (2020a) the **completion consistency** (CCY) of predictions from multiple partitioned sparse inputs is enforced via a Kullback-Leibler divergence. Differently, Rist et al. (2020a,b) enforces **spatial semantics consistency** (SCY) by minimizing the Jenssen-Shannon divergence of semantic inference between a given spatial point and some given *support points*. This self-supervision signal is available at any position within the scene. However, the strategy for *support points* is highly application dependent and while suitable for outdoor scene which have repetitive semantic patterns, we conjecture it might not scale as efficiently to cluttered indoor scenes.

Overall, little self-supervised or even unsupervised strategies exist and we believe such type of new losses (Zhang and Chen, 2021) should be encouraged.

4.4.4.2 Training strategies

The vast majority of SSC works are trained *end-to-end*, sometimes with *multi-scale* reconstructions. Few works also employ *two-stage* training or n-stage with *coarse-to-fine* strategies, or even train with *adversarial* learning to enforce realism. Strategies are illustrated in Figure 4.10 (with link color indicating the stage) and reviewed below.

End-to-end. Most architectures (Figure 4.10a, left) are trained end-to-end and output a *single scale* SSC (Song et al., 2017; Guo and Tong, 2018; Li et al., 2019; Dourado et al., 2020a; Zhang et al., 2019b; Li et al., 2020b; Liu et al., 2020; Li et al., 2020a; Chen et al., 2020b; Garbade et al., 2019; Dourado et al., 2020b; Cherabier et al., 2018; Rist et al., 2020a; Wang et al., 2020a) – often similar to the input size. Training that way is straightforward and often offers minimal memory footprint. Noteworthy, Dai et al. (2020) – which does geometric completion only – gradually increases sparsity during training to ease completion of large missing chunks.



(a) End-to-end

(b) Two-stage

(c) Coarse-to-fine

(d) Adversarial

(a) End-to-end, single-scale: (Song et al., 2017; Guo and Tong, 2018; Li et al., 2019; Dourado et al., 2020a; Zhang et al., 2019b; Li et al., 2020b; Liu et al., 2020; Li et al., 2020c,a; Chen et al., 2020b; Wang et al., 2020a; Garbade et al., 2019; Rist et al., 2020a; Cherabier et al., 2018; Dourado et al., 2020b), multi-scale: (Zhang et al., 2018a; Roldão et al., 2020).

(b) Two-stage, conv+conv: (Li et al., 2020c; Garbade et al., 2019; Liu et al., 2018), conv+point: (Zhong and Zeng, 2020).

(c) Coarse-to-fine: (Dai et al., 2018).

(d) Adversarial: (Wang et al., 2018d, 2019c; Chen et al., 2019b; Wu et al., 2020).

Figure 4.10: **Training strategies.** Most SSC are trained end-to-end (a) outputting single or multi-scale SSC. Differently, two-stage training (b) commonly lift semantic features calculated on sparse input to a second stage network. Coarse-to-fine (c), similarly to multi-scale relies on multiple size predictions, but trains in a multi-stage coarse to fine manner. Finally, Adversarial training (d) discriminates between ground truth and predicted scenes. Ⓕ stands for fusion of any type.

To guide the training, multi-scale SSC outputs can also be supervised, typically from early layers of a UNet decoder. A simple, yet efficient *multi-scale* strategy (Roldão et al., 2020; Zhang et al., 2018a) is to minimize the sum of SSC losses at different resolution (Figure 4.10a, right), thus also enforcing coarse SSC representations in the network. In Zhang et al. (2018a), two different scales are predicted, versus four in Roldão et al. (2020) providing down to 1:8 (1 over 8) downscaled SSC. In the latter, authors also report that the decoder can be ablated to provide very fast inference at coarsest resolution (370FPS at 1:8 scale). All end-to-end networks are commonly trained from scratch, though some of them (Dourado et al., 2020a; Zhang et al., 2019b; Liu et al., 2018; Song et al., 2017; Guo and Tong, 2018) report pretraining on the synthetic SUNCG.

Two-stages. Some works also use two-stage training, where the first stage learns 2D or 3D semantics, and the second stage predicts SSC from input and inferred semantics (Figure 4.10b), as in Li et al. (2020c); Garbade et al. (2019); Zhong and Zeng (2020); Liu et al. (2018). Noteworthy, one may argue these methods are still end-to-end, since off-the-shelf semantics networks could be used. Only SATNet (Liu et al., 2018) is strictly two-stage since semantics is also refined in the second stage.

Coarse-to-fine. ScanComplete (Dai et al., 2018) also follows a multi-scale strategy somehow close to Roldão et al. (2020); Zhang et al. (2018a), though training in a coarse-to-fine manner (Figure 4.10c). In details, three sequential training are achieved at increasingly higher resolutions, with each stage network taking as input the ad-hoc sparse input and the previous stage SSC prediction (for stage>1). Interestingly, no one explored a continuous curriculum learning setting, which could yield stabler training and performance improvement. Still, Cherabier et al. (2018) (intentionally omitted Figure 4.10c) applies a unique coarse-to-fine proposal in a fully end-to-end manner, via parallel backpropagations in all scales. Of similar spirit, Dai et al. (2020) proposes an iteration-based progressive refinement during training for scene completion, but insights of such strategy are not deeply discussed.

Adversarial. Even SSC ground truth have large missing chunks of data, leading to ambiguous supervision. To address this, Wang et al. (2018d, 2019c); Chen et al. (2019b); Wu et al. (2020) use adversarial training (Figure 4.10d), since the discriminator provides an additional supervision signal. This is straightforward implemented in Wu et al. (2020); Chen et al. (2019b), where the discriminator classifies ground truth from generated SSC (aka real/fake). In Wang et al. (2018d, 2019c) of same authors, 2 discriminators are used in a somehow similar fashion to discriminate both the SSC output and the latent depth or semantics features to enforce deep shared represen-

tation. Additionally, Chen et al. (2020a) employs a Conditional Variational Autoencoder (CVAE) to generate completed border sketches to be fed to the main SSC branch. Despite few works on the matter, adversarial appears a logical choice to improve SSC consistency and provide additional self-supervision. Both Wang et al. (2018d) and Wu et al. (2020) report a 10%-15% boost on several datasets.

Finally, on implementation – where mentioned – only Dai et al. (2018); Zhong and Zeng (2020); Wang et al. (2018d, 2019c); Roldão et al. (2020) train with Adam optimizer, Chen et al. (2019b) with a mix of Adam/SGD, and all others use only SGD with momentum 0.9 and 10^{-4} weight decay, except for Zhang et al. (2019b); Dourado et al. (2020a); Li et al. (2020c); Chen et al. (2020b); Wang et al. (2017, 2020a); Chen et al. (2019b) using 5×10^{-4}. The training most often uses standard learning rate scheduler (Li et al., 2019; Zhang et al., 2019b; Li et al., 2020b; Liu et al., 2020; Li et al., 2020a; Chen et al., 2020b; Wang et al., 2017; Zhong and Zeng, 2020; Garbade et al., 2019; Wang et al., 2020a) though sophisticated scheduling (Dourado et al., 2020a) or fixed learning rate (Li et al., 2020c) are also used. Because of 3D greediness, the common practice is to train with small batch size of 1 (Li et al., 2020c), 2 (Li et al., 2019), 3 (Dourado et al., 2020a), 4 (Guo and Tong, 2018; Zhang et al., 2019b; Li et al., 2020b; Liu et al., 2020; Li et al., 2020a; Dourado et al., 2020a; Roldão et al., 2020; Zhong and Zeng, 2020; Chen et al., 2019b; Cherabier et al., 2018), 8 (Wang et al., 2017, 2020a, 2018d, 2019c) or 16 (Chen et al., 2020b) to fit in standard 12GB GPUs.

4.4.5 Evaluation

We now provide in-depth evaluation of the field, reviewing first the common metrics (Section 4.4.5.1), the qualitative and quantitative performance of the literature (Section 4.4.5.2), and the networks efficiency (Section 4.4.5.3).

4.4.5.1 Metrics

Joint Semantics-Geometry. Preferred metric for SSC is the mean **Jaccard Index** or **mean Intersection over Union** (mIoU) (Everingham et al., 2014), which considers IoU of all semantic classes for prediction, without considering free space. It writes

$$\text{mIoU} = \frac{1}{C} \sum_{c=1}^{C} \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c + \text{FN}_c}, \tag{4.6}$$

where TP_c, FP_c and FN_c are the true positives, false positives and false negatives predictions for class c, respectively. Since ground truth is commonly semi-dense for real-world datasets, evaluation is performed in known space only.

Geometry only. Because mIoU considers semantic classes, the pure geometrical reconstruction quality is not encompassed. Therefore **Intersection over Union** (*IoU*), along with **Precision** and **Recall** are commonly used on the binary free/occupy scene representation, obtained by mapping all semantic classes to *occupy*.

Alternatively, any distance metrics from Section 4.4.4.1 (i.e. $\boldsymbol{\ell_1}$, $\boldsymbol{\ell_2}$, **EMD** or **CD**) may be used as in Dai et al. (2018, 2020) though less used in real datasets, due to their lower precision when sparsity increases.

On common practice, we highlight that evaluation on real indoor or outdoor datasets is usually performed differently. This results of the common sensors setup, respectively RGB-D (indoor) and LiDAR (outdoor), providing significantly different density information. Referring to Figure 4.2, in real indoor Song et al. (2017) the geometrical IoU is evaluated on input occluded regions while the mIoU is evaluated on input occluded (blue) and observed (red) surfaces. In real outdoor Behley et al. (2019) the IoU and mIoU are commonly evaluated on the entire known space, regardless whether regions were observed or occluded in the input. Obviously, synthetic datasets can cope with either practice. In the following, we describe the common practices and report semantics metrics (mIoU) along with geometrical ones (Precision, Recall, IoU).

4.4.5.2 Performances

We report the available mIoU and IoU performance on the most popular SSC datasets in Table 4.3, which are all obtained from voxelized ground truth. For non-voxel methods the output is voxelized beforehand. Additionally, detailed classwise performance of top five methods for SemanticKITTI, NYUv2 and SUNCG are presented in Tabs. 4.4, 4.5 and 4.6, respectively. From the performance Table 4.3, the mIoU of the best methods plateaus around $75-85\%$ on synthetic indoor dataset, 41% on real indoor, and 30% on real outdoor. Importantly, note that *all* indoor datasets performance are evaluated at 1:4 of the original ground truth resolution – that is $60\times36\times60$ – for historical reasons[4]. This makes indoor / outdoor performance comparison tricky. It is interesting to note that IoU – geometrical completion (i.e. ignoring semantics) – is way higher than best mIoU. In detail, best IoU are 73% on real indoor, and 57% on real outdoor. Qualitative results of a dozen of methods are shown in Figure 4.11 for indoor datasets, and Figure 4.12 for outdoor datasets.

[4]In their seminal work, for memory reason Song et al. (2017) evaluated SSC only at the 1:4 scale. Subsequently, to provide fair comparisons between indoor datasets and methods, all other indoor SSC have been using the same resolution despite the fact that higher resolution ground truth is available. Recent experiments in Chen et al. (2020a) advocate that using higher input/output resolution boosts the SSC performance significantly.

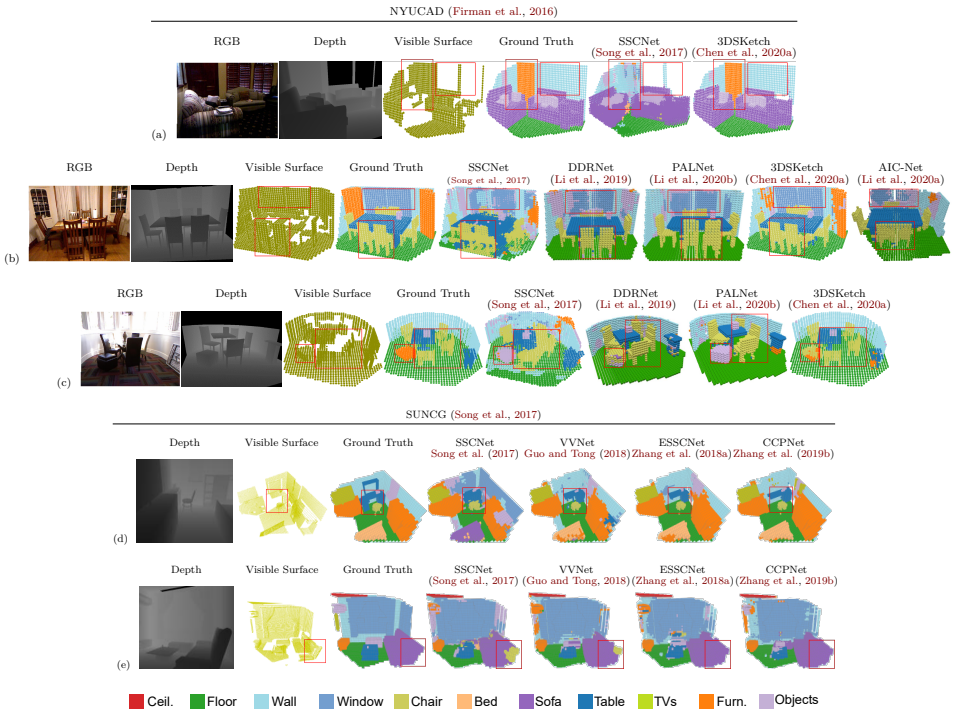


Figure 4.11: **Performance of indoor Semantic Scene Completion** on NYUCAD (Firman et al., 2016) and SUNCG (Song et al., 2017). Methods with RGB modalities (i.e. 3DSketch) enable to detect color salient objects as highlighted door in row (a). Position awareness also contributes to better reconstruction consistency and inter-class distinction as seen in columns (b), (c) by PALNet (Li et al., 2020b) and 3DSketch (Chen et al., 2020a). Multi-scale aggregation also improves reconstruction performance as seen on rows (d), (e), where CCPNet (Zhang et al., 2019b) achieves best performance on SUNCG (Song et al., 2017).

Overall, one may note the synthetic to real best performance gap of indoor datasets, which is approx. $10-35\%$ mIoU and $10-18\%$ IoU. While a difference is expected, once again it highlights that geometry has a smaller synthetic/real domain gap compared to semantics. On a general note also, most methods perform significantly better on IoU than on mIoU, demonstrating the complexity of the *semantics* scene completion. In fact, the ranking of methods differs depending on the metric. For example, on NYUv2 (indoor) CCPNet (Zhang et al., 2019b) gets best indoor mIoU (41.3%) while Chen et al. (Chen et al., 2020b) is best on IoU (73.4%), and on SemanticKITTI (outdoor) S3CNet (Cheng et al., 2020) has best mIoU (29.5%) and Local-DIFs (Rist et al., 2020a) best IoU (57.7%). Note also the large difference between best indoor/outdoor metrics. While only a handful

methods (Song et al., 2017; Zhang et al., 2018a; Liu et al., 2018; Roldão et al., 2020) are evaluated in both setups, they indeed perform significantly worse on outdoor data – though indoor/outdoor performance should be carefully compared given the different resolution. This is partially explained by the higher sparsity in outdoor datasets, visible in 'input' of Figure 4.12. Another explanation is the higher number of classes in SemanticKITTI versus NYU and the extreme class-imbalance setup given that minor classes are very rarely observed, see Figure 4.3. On general qualitative results, either indoor (Figure 4.11) or outdoor (Figure 4.12) results show that predictions are accurate in large homogeneous areas (walls/ground, floor, buildings) and most errors occur at object boundaries. This is evident in Table 4.4, where most methods achieve high performance in largest classes of SemanticKITTI, but struggle with predictions in less represented ones (e.g. bicycle, motorcycle, person). Worth mentioning, S3CNet (Cheng et al., 2020) achieves considerably larger scores in rare classes (+25%, +37%, +38% respectively), more than twice when compared to next best classed scores. The reason for such behavior is regrettably not deeply explored in their work.

Inputs. To ease interpretation, col 'Input' in Table 4.3 shows the nature of input used, where 'G' is Geometry of any type (depth, TSDF, points, etc.) – possibly several – and 'T' is Texture (RGB). From Table 4.3 using both geometry *and* texture (G+T) performs among the best indoor, such as 3DSketch (Chen et al., 2020a) which relies on textural edges and depth. Generally speaking, G+T enables the prediction of non salient geometric objects (i.e. paints, windows, doors) as shown on Figure 4.11a by the door predicted 3DSketch (Chen et al., 2020a) and missed by SSCNet (Song et al., 2017). Noteworthy, among the best mIoU methods Zhang et al. (2019b); Cheng et al. (2020); Chen et al. (2020a) all use TSDF-encoding as geometrical input. On outdoor datasets, only TS3D (Garbade et al., 2019) uses texture without significant improvement. We conjecture the reason is threefold. First, we argue the field of view being significantly bigger than indoor the color is less informative. Second, the depth sensor (LiDAR) in outdoor dataset is very precise (as opposed to RGB-D in indoor settings) and provide a more discriminative signal. Thirdly, the possibly inaccurate Camera-Lidar calibration and the absence of 1 to 1 associations.

Architecture and design choices. One may notice the good performance of hybrid networks (Chen et al., 2020a; Li et al., 2020b; Zhong and Zeng, 2020; Cheng et al., 2020) (Figure 4.7d), which we believe results of richer input signal due to the fusion of multiple modalities. We also argue that multiple neighboring definitions (2D and 3D) provide beneficial complementary signals. For instance, S3CNet combines 2D BEV and 3D f-TSDF for late fusion through post-processing refinement, achieving best semantic

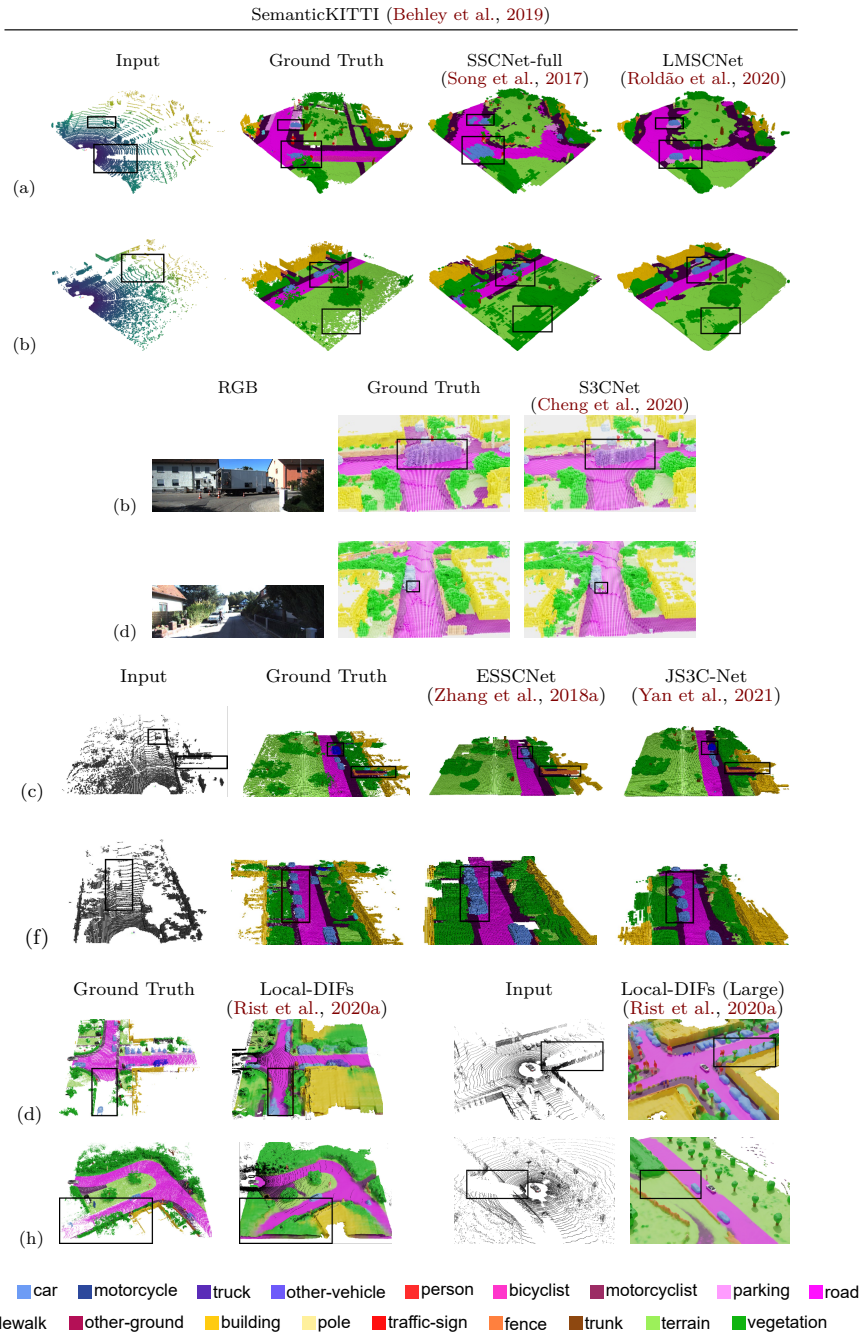


Figure 4.12: **Performance of outdoor Semantic Scene Completion** on SemanticKITTI (Behley et al., 2019). LMSCNet (Roldão et al., 2020) proposes a lightweight architecture with small performance decrease, rows (a), (b). S3CNet (Cheng et al., 2020) achieves SoA performance by their sparse bird's eye view and 3D f-TSDF feature encoders, rows (c), (d). Two-stage JS3CNet (Yan et al., 2021) performs point-wise semantic segmentation and semantic scene completion sequentially, enabling better completion as seen in rows (e), (f). Finally, Local-DIFs (Rist et al., 2020a) enables continuous surface prediction, thanks to deep implicit functions, which enable predictions of considerably larger spatial extent, rows (g), (h).

			Indoor												Outdoor			
			Real-world NYUv2 (Silberman et al., 2012) 60 × 36 × 60				Synthetic NYUCAD (Firman et al., 2016) 60 × 36 × 60				SUNCG (Song et al., 2017) 60 × 36 × 60				Real-world SemanticKITTI (Behley et al., 2019) 256 × 32 × 256			
	Method	Input	Prec.	Recall	IoU	mIoU	Prec.	Recall	IoU	mIoU	Prec.	Recall	IoU	mIoU	Prec.	Recall	IoU	mIoU
2017	SSCNet (Song et al., 2017)[a]	G	59.3†	92.9†	56.6†	30.5†	75.0†	**96.0**†	73.0†	-	76.3	95.2	73.5	46.4	31.7	83.4	29.8	9.5
	SSCNet-full (Roldão et al., 2020)	G	-	-	-	-	-	-	-	-	-	-	-	-	59.6	75.5	50.0	16.1
2018	Guedes et al. (2018)	G+T	62.5	82.3	54.3	27.5	-	-	-	-	-	-	-	-	-	-	-	-
	VVNet (Guo and Tong, 2018)	G	69.8†	83.1†	61.1†	32.9†	86.4†	92.0†	80.3†	-	90.8	91.7	84.0	66.7	-	-	-	-
	VD-CRF (Zhang et al., 2018b)	G	-	-	60.0†	31.8†	-	-	78.4†	43.0†	-	-	74.5	48.8	-	-	-	-
	ESSCNet (Zhang et al., 2018a)[b]	G	71.9	71.9	56.2	26.7	-	-	-	-	92.6	90.4	84.5	70.5	62.6	55.6	41.8	17.5
	SATNet (Liu et al., 2018)	G+T	67.3†	85.8†	60.6†	34.4†	-	-	-	-	80.7*	96.5*	78.5*	64.3*	-	-	-	-
2019	DDRNet (Li et al., 2019)	G+T	71.5	80.8	61.0	30.4	88.7	88.5	79.4	42.8	-	-	-	-	-	-	-	-
	TS3D (Garbade et al., 2019)[c]	G+T	-	-	60.0	34.1	-	-	76.1	46.2	-	-	-	-	31.6	84.2	29.8	9.5
	TS3D+DNet (Behley et al., 2019)	G	-	-	-	-	-	-	-	-	-	-	-	-	25.9	**88.3**	25.0	10.2
	TS3D+DNet+SATNet (Behley et al., 2019)	G	-	-	-	-	-	-	-	-	-	-	-	-	80.5	57.7	50.6	17.7
	EdgeNet (Dourado et al., 2020a)	G+T	79.1†	66.6†	56.7†	33.7†	-	-	-	-	93.1*	90.4*	84.8*	69.5*	-	-	-	-
	SSC-GAN (Chen et al., 2019b)	G	63.1	87.8	57.8	22.7	80.7	91.1	74.8	42.0	83.4	92.4	78.1	55.6	-	-	-	-
	ForkNet (Wang et al., 2019c)	G	-	-	63.4†	37.1†	-	-	-	-	-	-	86.9	63.4	-	-	-	-
	CCPNet (Zhang et al., 2019b)	G	78.8†	**94.3**†	67.1†	**41.3**†	**93.4**†	91.2†	**85.1**†	55.0†	**98.2**	**96.8**	**91.4**	74.2	-	-	-	-
	AM^{2}FNet (Chen et al., 2019a)	G+T	72.1	80.4	61.3	31.7	87.2	91.0	80.2	44.6	-	-	-	-	-	-	-	-
2020	GRFNet (Liu et al., 2020)	G+T	68.4	85.4	61.2	32.9	87.2	91.0	80.1	45.3	-	-	-	-	-	-	-	-
	AMFNet (Li et al., 2020c)	G+T	67.9	82.3	59.0	33.0	-	-	-	-	-	-	-	-	-	-	-	-
	PALNet (Li et al., 2020b)	G	68.7	85.0	61.3	34.1	87.2	91.7	80.8	46.6	-	-	-	-	-	-	-	-
	3DSketch (Chen et al., 2020a)	G+T	**85.0**	81.6	71.3	41.1	90.6	92.2	84.2	**55.2**	-	-	88.2*	**76.5***	-	-	-	-
	AIC-Net (Li et al., 2020a)	G+T	62.4	91.8	59.2	33.3	88.2	90.3	80.5	45.8	-	-	-	-	-	-	-	-
	Wang et al. (2020a)	G	-	-	-	-	-	-	-	-	92.1	95.5	88.1	74.8	-	-	-	-
	IPF-SPCNet (Zhong and Zeng, 2020)	G+T	70.5	46.7	39.0	35.1	83.3	72.7	63.5	50.7	-	-	-	-	-	-	-	-
	Chen et al. (2020b)	G	-	-	**73.4**	34.4	-	-	82.2	44.5	-	-	84.8	63.5	-	-	-	-
	LMSCNet (Roldão et al., 2020)	G	-	-	-	-	-	-	-	-	-	-	-	-	77.1	66.2	55.3	17.0
	LMSCNet-SS (Roldão et al., 2020)	G	-	-	62.2‡	28.4‡	-	-	-	-	-	-	-	-	**81.6**	65.1	56.7	17.6
	S3CNet (Cheng et al., 2020)	G	-	-	-	-	-	-	-	-	-	-	-	-	-	-	45.6	**29.5**
	JS3C-Net (Yan et al., 2021)	G	-	-	-	-	-	-	-	-	-	-	-	-	71.5	73.5	56.6	23.8
	Local-DIFs (Rist et al., 2020a)	G	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**57.7**	22.7

[a] Results in SemanticKITTI reported on Roldão et al. (2020). [b] Results in SemanticKITTI reported on Yan et al. (2021).
[c] Results in SemanticKITTI reported on Behley et al. (2019).
Input: Geometry (depth, range, points, etc.), Texture (RGB). † Pretraining on SUNCG. * Texture input not used due to absence in SUNCG. ‡ Own implementation.

Table 4.3: **SSC performance on the most popular datasets.** The relatively low best mIoU scores on the challenging real outdoor SemanticKitti (Behley et al., 2019) (29.5%) and real indoor NYUv2 (Silberman et al., 2012) (41.1%) show the complexity of the task. In the 'method' column, we indicate variants with an offset. To better interpret the performance, column 'Input' shows the type of input modality used where 'G' is Geometry (depth, range, points, etc.) and 'T' is Texture (RGB). Note that all indoor datasets report performance for 60 × 36 × 60 grids for historical reasons though 4x bigger input is commonly treated, cf. Section 4.4.5.2. Top 5 methods are highlighted in each column from red to white.

Method	Input	road (15.30%)	sidewalk (11.13%)	parking (1.12%)	other-gr. (0.56%)	building (14.1%)	car (3.92%)	truck (0.16%)	bicycle (0.03%)	motorcycle (0.03%)	other-veh. (0.20%)	vegetation (39.3%)	trunk (0.51%)	terrain (9.17%)	person (0.07%)	bicyclist (0.07%)	motorcyclist (0.05%)	fence (3.90%)	pole (0.29%)	tr. sign (0.08%)	mIoU
S3CNet (Yan et al., 2021)	G	42.0	22.5	17.0	7.9	**50.2**	31.2	6.7	**41.5**	**45.0**	**16.1**	39.5	**34.0**	21.2	**45.9**	**35.8**	**16.0**	**31.3**	**31.0**	**24.3**	**29.5**
JS3CNet (Yan et al., 2021)	G	64.7	39.9	34.9	**14.1**	39.4	33.3	**7.2**	14.4	8.8	12.7	**43.1**	19.6	**40.5**	8.0	5.1	0.4	30.4	18.9	15.9	23.8
Local-DIFs (Rist et al., 2020a)	G	**67.9**	**42.9**	**40.1**	11.4	40.4	**34.8**	4.4	3.6	2.4	4.8	42.2	26.5	39.1	2.5	1.1	0	29.0	21.3	17.5	22.7
TS3D+DNet+SATNet (Behley et al., 2019)	G	62.2	31.6	23.3	6.5	34.1	30.7	4.9	0	0	0.1	40.1	21.9	33.1	0	0	0	24.1	16.9	6.9	17.7
LMSCNet-SS(Roldão et al., 2020)	G	64.8	34.7	29.0	4.6	38.1	30.9	1.5	0	0	0.8	41.3	19.9	32.1	0	0	0	21.3	15.0	0.8	17.6

Table 4.4: **Detailed SSC class performance on SemanticKITTI (Behley et al., 2019) dataset.** Best 5 methods are presented and ordered in decreasing mIoU performance from top to bottom.

Method	Input	ceil. (0.74%)	floor (12.44%)	wall (9.67%)	win. (2.12%)	chair (2.03%)	bed (9.17%)	sofa (6.78%)	table (4.14%)	tvs (0.53%)	furn. (36.64%)	objs. (15.74%)	mIoU
CCPNet (Zhang et al., 2019b)†	G	25.5	**98.5**	38.8	**27.1**	27.3	**64.8**	**58.4**	21.5	**30.1**	38.4	23.8	**41.3**
3DSketch (Chen et al., 2020a)	G+T	**43.1**	93.6	40.5	24.3	30.0	57.1	49.3	**29.2**	14.3	42.5	**28.6**	41.1
ForkNet (Wang et al., 2019c)†	G	36.2	93.8	29.2	18.9	17.7	61.6	52.9	23.3	19.5	**45.4**	20.0	37.1
IPF-SPCNet (Zhong and Zeng, 2020)	G+T	32.7	66.0	**41.2**	17.2	**34.7**	55.3	47.0	21.7	12.5	38.4	19.2	35.1
SATNet (Liu et al., 2018)†	G+T	17.3	92.1	28.0	16.6	19.3	57.5	53.8	17.7	18.5	38.4	18.9	34.4

† Pretraining on SUNCG.

Table 4.5: **Detailed SSC class performance on NYUv2 (Silberman et al., 2012) dataset.** Best 5 methods are presented and ordered in decreasing mIoU performance from top to bottom.

Method	Input	ceil. (2.68%)	floor (12.27%)	wall (33.55%)	win. (5.79%)	chair (1.80%)	bed (5.95%)	sofa (4.94%)	table (2.90%)	tvs (0.36%)	furn. (15.04%)	objs. (14.73%)	mIoU
3DSketch (Chen et al., 2020a)‡	G*	97.8	91.9	**84.1**	**72.6**	60.8	86.8	81.7	**68.7**	52.6	75.7	**68.2**	**76.5**
Wang et al. (2020a)	G	98.2	92.8	76.3	61.9	**62.4**	**87.5**	80.5	66.3	**55.2**	74.6	67.8	74.8
CCPNet (Zhang et al., 2019b)	G	**99.2**	89.3	76.2	63.3	58.2	86.1	**82.6**	65.6	53.2	**76.8**	65.2	74.2
ESSCNet (Zhang et al., 2018a)	G	96.6	83.7	74.9	59.0	55.1	83.3	78.0	61.5	47.4	73.5	62.9	70.5
EdgeNet (Dourado et al., 2020a)	G*	97.2	**94.4**	78.4	56.1	50.4	80.5	73.8	54.5	49.8	69.5	59.2	69.5

* Texture input not used due to absence in SUNCG.
‡ Results provided by authors.

Table 4.6: **Detailed SSC class performance on SUNCG (Song et al., 2017) dataset.** Best 5 methods are presented and ordered in decreasing mIoU performance from top to bottom.

completion performance on SemanticKITTI (Behley et al., 2019) by a considerable margin (+5.7% mIoU) . Qualitative results of the approach are shown in Figure 4.12c, 4.12d. Similarly, JS3CNet (Yan et al., 2021) ranks second in same dataset (23.8% mIoU and 56.6% IoU) with point-wise semantic labeling through SparseConvNet architecture (Graham et al., 2018) and dense semantic completion using a point-voxel interaction module, enabling to better infer small vehicles as shown in circled areas of Figure 4.12e, 4.12f. Analogously, PALNet (Li et al., 2020b) middle fuses depth image and f-TSDF features, achieving good performance on NYUv2 (34.1% mIoU and 61.3% mIoU) and NYUCAD (46.6% mIoU and 80.8% mIoU) datasets, such performance can also be attributed to its position aware loss, to be discussed next.

Contextual awareness (Section 4.4.3.1) seems also to play an important role for the task. This is noticeable with CCPNet (Zhang et al., 2019b) encouraging results given the use of a single geometric input. We observe the

benefit of its consistency loss in the highlighted areas of Figure 4.11d, 4.11e.

On position awareness (Section 4.4.3.2) it seems to boost intra-class consistency together with inter-class distinction. For example 3DSKetch (Chen et al., 2020a) and PALNet (Li et al., 2020b), both use position awareness and achieve high performances in indoor scenes with 3DSketch ranking in top 2 in mIoU (Table 4.3), visible in Figs. 4.11a, 4.11b, 4.11c. Similarly, S3CNet dominates performance in SemanticKITTI as already mentioned, which performance is noticeable in Figs. 4.12c, 4.12d.

An interesting observation is the high density of the completion even regarding the ground truth, visible in Figs. 4.12b, 4.12g, 4.12h. This relationship is studied in Dai et al. (2020), where sparsity is exploited by removing input data to impulse unknown space completion.

4.4.5.3 Network efficiency

	Method	Params (M)	FLOPs (G)
2017	SSCNet (Song et al., 2017)[a]	0.93	163.8
	VVNet (Guo and Tong, 2018)[a]	0.69	119.2
	ESSCNet (Zhang et al., 2018a)[a]	0.16	22
	SATNet (Liu et al., 2018)[a]	1.2	187.5
2018	DDRNet (Li et al., 2019)	0.20	27.2
	CCPNet (Zhang et al., 2019b)	0.09	11.8
2020	GRFNet (Liu et al., 2020)	0.82	713
	PALNet (Li et al., 2020b)	0.22	78.8
	AIC-Net (Li et al., 2020a)	0.72	96.77
	Chen et al. (2020b)	**0.07**	**1.6**

[a] Reported in Zhang et al. (2019b).

(a) $60 \times 36 \times 60$ prediction (indoor)

	Method	Params (M)	FLOPs (G)
2017	SSCNet (Song et al., 2017)[b]	0.93	82.5
	SSCNet-full (Song et al., 2017)[b]	1.09	769.6
2019	TS3D (Garbade et al., 2019)[b]	43.77	2016.7
	TS3D+DNet (Behley et al., 2019)[b]	51.31	847.1
	TS3D+DNet+SATNet (Behley et al., 2019)[b]	50.57	905.2
2020	LMSCNet (Roldão et al., 2020)	**0.35**	**72.6**
	JS3C-Net (Yan et al., 2021)	3.1	-
	Local-DIFs (Rist et al., 2020a)	9.9	-

[b] Reported in Roldão et al. (2020).

(b) $256 \times 32 \times 256$ prediction (outdoor)

Table 4.7: **Network statistics.** Number of parameters and FLOPs are reported per method, grouped by resolution output: $60 \times 36 \times 60$ for typical indoor datasets (Silberman et al., 2012; Firman et al., 2016; Song et al., 2017) or $256 \times 32 \times 256$ for outdoor dataset (Behley et al., 2019).

In Table 4.7, network parameters and floating-point operations (FLOPs) are listed – where possible – with separation of indoor and outdoor networks because they have different output resolution. Notice the extreme variations between networks, which scale from 1:144 in number of parameters and 1:1260 in FLOPs. Chen et al. (2020b) and LMSCNet (Roldão et al.,

2020) are by far the lightest networks with the fewest parameters and lower FLOPs, in indoor and outdoor settings respectively. They also account for the lower number of operations, which can – though not necessarily (Ma et al., 2018) – contribute to faster inference times. Furthermore, the use of sparse convolutions (Graham et al., 2018) is commonly applied as a strategy to reduce memory overhead in Zhang et al. (2018a); Dai et al. (2020); Yan et al. (2021); Cheng et al. (2020).

4.5 Discussion

Despite growing interest, there are still major challenges to solve SSC as the best methods still perform poorly on real datasets (see Tabs. 4.5, 4.4). In this section, we wish to highlight important remaining issues and provide future research directions.Despite growing interest, there are still major challenges to solve SSC as the best methods still perform poorly on real datasets (see Tabs. 4.5, 4.4). In this section, we wish to highlight important remaining issues and provide future research directions.

Best practices for SSC. Among the various viable SSC choices, some were proven highly beneficial. Contextual aggregation (Section 4.4.3.1) for example, to improve the information flow. Further geometrical cues often boost SSC, whether if it is multiple geometrical representations (e.g. depth + voxel, Section 4.4.1) or boundaries (e.g. edges, Section 4.4.3.2). On training, Rist et al. (2020a) shows free space supervision close to the geometry can provide sharper inference, and we believe adversarial training (Section 4.4.4.2) is key to cope with the ground truth ambiguities. On evaluation, we encourage authors to evaluate on indoor *and* outdoor dataset exhibiting different challenges. Finally for real-time applications, more works like Roldão et al. (2020); Chen et al. (2020b) should account for lightweight and fast inference architectures (Section 4.4.5.3).

Supervision bias. An important challenge for completion results from the big imbalance ratio between free and occupied space (9:1 in both NYUv2 (Song et al., 2017; Silberman et al., 2012) and SemanticKITTI (Behley et al., 2019)) which biases the networks towards free space predictions. To deal with this problem, random undersampling of the major free class is often applied (Song et al., 2017) to reach an acceptable 2:1 ratio. The strategy reportedly improves completion performance (i.e. +4% IoU (Song et al., 2017)) and is widely employed (Zhang et al., 2018a; Garbade et al., 2019; Liu et al., 2018; Dourado et al., 2020a; Zhang et al., 2019b). Similarly, loss can be balanced to favor occupy predictions (Li et al., 2019; Liu et al., 2020). Again, few works like Rist et al. (2020a) efficiently benefit from free space information.

Semantic class balancing. Imbalance is also present in the semantic labels, specially in outdoor datasets, where there is a prevalence of road or vegetation (see Figure 4.1 and 4.3). Class-balancing can be applied to mitigate imbalanced distribution, usually weighting each class according to the inverse of its frequency (Roldão et al., 2020), though prediction of under-represented classes still suffer (e.g. pedestrian or motorcycle in Behley et al. (2019)). This may have catastrophic impact for robotics application. An approach worth mentioning is S3CNet (Cheng et al., 2020), where combined weighted cross entropy and position aware loss (cf. Section 4.4.4.1) achieve impressive improvements in under-represented classes of SemanticKITTI. We believe SSC could benefit of smarter balancing strategies.

Object motion. As mentioned in Section 4.3.1, real-world ground truth is obtained by the *rigid* registration of contiguous frames. While this corrects for ego motion, it doesn't account for scene motion and moving objects produce temporal tubes in the ground truth, as visible in SemanticKITTI (Behley et al., 2019) (Figure 4.4c). As such, to maximize performance, the SSC network must additionally predict motion of any moving objects.
To evaluate the influence of such imperfections for SSC, some works reconstruct target scenes by accounting only for a few future scans (Yan et al., 2021; Rist et al., 2020a). Results show marginal completion improvement from the application of such strategy. An alternative proposal (Kim and Kim, 2020), is to remove dynamic objects from the detection of spatial singularities after frames registration. On the challenging SemanticKITTI (Behley et al., 2019), because there are little insights to classify dynamic objects, all methods tend to predict vehicles as stationary (cf. Figure 4.12) – producing appealing results but being punished by dataset metrics. This obviously result of the dataset bias, given the abundance of parked vehicles.

The introduction of larger synthetic datasets (Dosovitskiy et al., 2017; Ros et al., 2016) could be an interesting solution to fight ground truth inaccuracies.

Datasets extendable for SSC. Because semantic labeling is the most complex and costly, we denote that a large amount of existing 3D semantics datasets (Vallet et al., 2015; Hackel et al., 2017; Roynard et al., 2018; Straub et al., 2019; Caesar et al., 2020; Tan et al., 2020; Fu et al., 2020) could also be extended to SSC at the cost of some processing effort. A selective list of these *SSC-extendable* datasets is in Table 4.8 and we believe that their use should be encouraged to serve the interest of research on SSC. Interestingly, most need little processing for SSC (e.g. sparse input generation from 3D meshes or point clouds, virtual sensor configurations) (Vallet et al., 2015; Hackel et al., 2017; Roynard et al., 2018; Straub et al., 2019; Tan et al., 2020), though some require more complex processing (e.g. ag-

Dataset	Year	Type	Nature	Data	3D Sensor	# Classes	Extension	#Sequences
IQMulus (Vallet et al., 2015)	2015	Real-wolrd	Outdoor	→ Points	Lidar	-	Sparse input scene subsampling	-
Semantic3D (Hackel et al., 2017)	2017	Real-world	Outdoor	→ Points	3D Scanner	8	Sparse input scene subsampling	30
Paris-Lille-3D (Roynard et al., 2018)	2018	Real-world	Outdoor	→ Points	Lidar-32	50	Sparse input scene subsampling	4
Replica (Straub et al., 2019)	2019	Real-world†	Indoor	→ 3D Mesh	RGB-D	88	Sparse input from virtual RGB-D	35
nuScenes (Caesar et al., 2020)	2020	Real-world	Outdoor	Points/RGB →	Lidar-32	32	Dense scenes registration	1000
Toronto-3D (Tan et al., 2020)	2020	Real-world	Outdosor	→ Points	Lidar-32	8	Sparse input scene subsampling	4
3D-FRONT (Fu et al., 2020)	2020	Synthetic	Indoor	→ Mesh	-	-	Sparse input from virtual RGB-D	-

† Synthetically augmented.

Table 4.8: **SSC-extendable datasets**. To promote research on SSC we highlight that existing 3D semantic datasets could be extended for SSC, at the cost of processing work (cf. col. 'Extension'). While some extensions could be obtained with little processing (e.g. Replica (Straub et al., 2019), 3D-Front (Fu et al., 2020)), others are significantly more complex (e.g. nuScenes (Caesar et al., 2020)).

gregation of sparse inputs (Caesar et al., 2020)). We also encourage the use of autonomous driving simulators such as CARLA (Dosovitskiy et al., 2017), SYNTHIA (Ros et al., 2016) for *synthetic* dataset generation, devoid of dynamic objects and subsequent registration problems. More extensive surveys on RGB-D and Lidar datasets are provided in Gao et al. (2020); Firman (2016).

4.6 Conclusion

In this chapter we provided a comprehensive review of 3D Semantic Scene Completion (SSC). Even though reviewed techniques achieve encouraging results in the task, performance still shows large gap for improvement in real-world data (41.3% and 29.5% mIoU on indoor and outdoor scenes respectively), evidencing its difficulty.

Furthermore, we highlighted the most important points to consider for the task, covering the input encoding and modalities applied (Section 4.4.1). While the calculation of a TSDF for input encoding is very popular, it commonly requires high computation times and it seems to bring small benefit when compared to regular occupancy encoding. Additional depth modality seems to bring good results as it provides complementary 2-dimensional neighboring information. Naturally, RGB enables performance improvement as it brings additional texture features.

We also provided an overview of the popular architectures considering both performance and computational complexity. We remark once again the relevance of multi-scale contextual information to improve performance as it enables to gather features from different scales to capture both local geometrical details and high-level contextual information (Section 4.4.3.1). We also showed that position awareness provides important guidance by exploiting the local geometric anisotropy, either through border detection techniques or the use of dedicated losses (Section 4.4.3.2).

Moreover, we covered popular lightweight architecture designs to reduce memory overhead without compromising accuracy (Section 4.4.3.4). Sparse convolutions have gained popularity in recent works as they enable to reduce the large memory needs of dense convolutions, although they must be employed in combination to the latter to enable data dilation required for the completion task. Dilated convolutions are also widely used as they provide large context through augmented receptive fields at small cost.

Finally, we presented the most common losses and metrics employed to perform and evaluate the task, reviewing and critically analyzing major aspects of proposed approaches, including important design choices to be considered, and compared their performance in popular datasets. In the next chapter we present our contribution for lightweight multiscale semantic completion of outdoor scenes.

Chapter 5

LMSCNet: Lightweight Multiscale Semantic Completion

The contributions of this chapter were published in Roldão et al. (2020):

Roldão, L., de Charette, R., and Verroust-Blondet, A. (2020). LMSCNet: Lightweight multiscale 3D semantic completion. In *3DV 2020*.

1-minute demo video: https://youtu.be/J6dYoWx4Xqw
10-minute explanation video: https://youtu.be/Wh1qrqqnOgE

The code is publicly available at:

https://github.com/cv-rits/LMSCNet

The work presented in this chapter was carried out before the 3D semantic scene completion survey (Roldão et al., 2021) covered in Chapter 4.

To avoid repetition with the latter, we do not cover related works. This reduces the length of the chapter, although we believe it carries the main algorithmic contribution of this thesis.

Contents

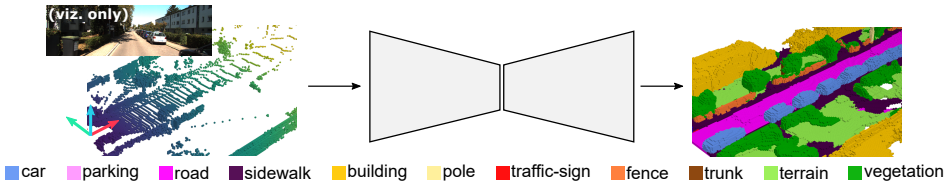


Figure 5.1: 3D LiDAR scans provide sparse data with large regions of missing information due to sensor configuration, occlusions and limited field of view. Semantic scene completion (SSC) aims to jointly complete and semantically label the sparse 3D scene.

5.1 Introduction

In the previous chapter we have highlighted the importance and evolution of the semantic scene completion (SSC) task, which aim is to jointly complete and semantically annotate entire 3D scenes. As explained in Section 4.2, the problem is commonly addressed by learning priors from pairs of sparse and dense semantically labeled 3D scenes. The topic gained popularity with the introduction of SSCNet (Song et al., 2017) which semantically completes 3D indoor scenes from RGB-D scans. Despite variety of existing approaches presented in indoor RGB-D data, only a few works (Song et al., 2017; Garbade et al., 2019; Behley et al., 2019) were presented in outdoor LiDAR scenes at the time of submission. Although, outdoor SSC has received more interest thanks to the recent introduction of the outdoor large-scale SemanticKITTI dataset (Behley et al., 2019). However, little attention has been given to lightweight semantic completion methods in general for real time applications and fast inference.

Therefore, in this chapter we propose a Lightweight Multiscale Semantic Completion Network, coined LMSCNet, for multiscale 3D semantic completion from voxelized sparse 3D LiDAR scans. As opposed to the literature, we use a 2D UNet backbone with comprehensive multiscale skip connections to enhance feature flow, along with 3D segmentation heads. To the best of our knowledge, it is the only method that uses a 2D backbone architecture for semantic completion. In our proposal, multiscale predictions are also possible given informative features map flow, preserving computation efficiency and enabling very fast inference at coarse levels. Figure 5.5 shows the multiscale output of our LMSCNet on the SemanticKITTI dataset (Behley et al., 2019), using a single sparse LiDAR scan input encoded in a voxel grid.

While some works use progressive multiscale losses (Dai et al., 2020, 2018; Li et al., 2009), the literature ignores the benefit of multiscale completion which we prove useful for reducing inference times. Our method performed on par on semantic completion and better on occupancy completion than all other published methods on the SemanticKITTI benchmark at the time of

submission – while being significantly lighter and faster –. Post-submission methods have improved the performance shown by LMSCNet, though at larger memory and computation costs. As such, our approach provides a great performance/speed trade-off for mobile-robotics applications. Ablation studies demonstrate that our method is robust to lower density inputs, and that it enables very high speed semantic completion at the coarsest level. To summarize, the main contributions of our work are:

1. A novel 3D semantic scene completion pipeline from occupancy grids.

2. A lightweight architecture with a mix of 2D/3D convolutions leading to significantly less parameters.

3. A modular multiscale pipeline that enables coarser inferences at very high speeds.

4. State of the art performance on SemanticKITTI (Behley et al., 2019) at submission time.

5. An architecture that ranks among the lightest and fastest methods for semantic scene completion on the SemanticKITTI benchmark.

In this chapter we omit the related works section as a large state of the art on Semantic Scene Completion has been already presented in Chapter 4. Therefore, we directly introduce our method in the following section.

5.2 LMSCNet

Given a sparse 3D voxel grid, our goal is to predict the 3D semantic scene representation, where each voxel is being assigned a semantic label $Cs = [c_0, c_1, \ldots, c_N]$, where N is the number of semantic classes and c_0 stands for free voxels. Our architecture, coined LMSCNet and shown in Figure 5.2, is based on a lightweight UNet style architecture to predict 3D semantic completion at multiple scales, allowing fast coarse inference, beneficial for mobile robotics applications. Instead of greedy 3D convolutions, we mostly employ 2D convolutions along the height axis; similar to a bird-eye view. While the mix of 2D/3D convolutions has been used in other tasks, to the best of our knowledge we are the first to do semantic scene completion directly from 3D data processed by a 2D CNN backbone. In the following, we detail our custom lightweight 2D/3D architecture (Section 5.2.1), the multiscale reconstruction (Section 5.2.2), and the overall training pipeline (Section 5.2.3).

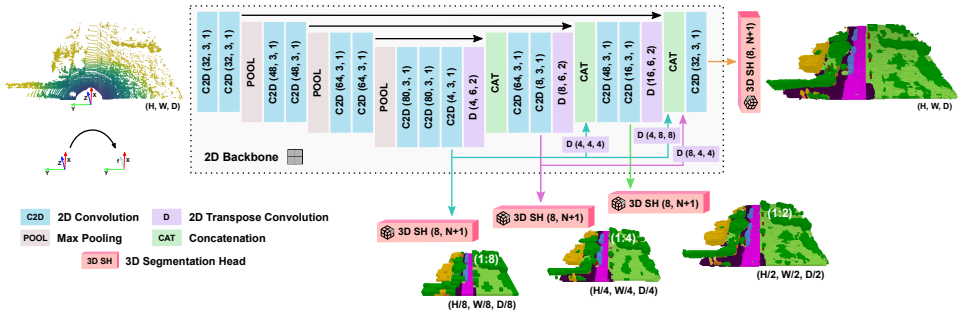


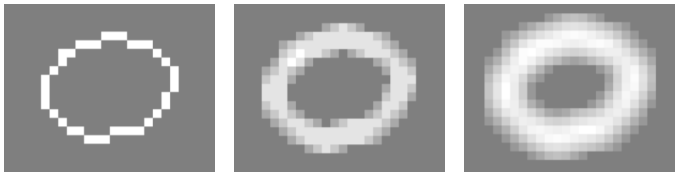
Figure 5.2: **LMSCNet: Lightweight Multiscale Semantic Completion Network.** Our pipeline uses a 2D UNet backbone architecture (in gray) with 3D segmentation heads (in red) to perform 3D semantic segmentation and completion at different scales, while preserving low complexity. Convolution parameters shown as: (number of filters, kernel size and stride). 3D segmentation heads parameters shown as: (number of filters input, number of filters output).

5.2.1 Lightweight architecture

To infer a dense output from the sparse input voxel grid, we use a standard encoder-decoder UNet architecture (Ronneberger et al., 2015) with 4 levels, thus learning features at decreasing resolutions. At each level, a series of convolution operations is applied followed by a pooling; downscaling the resolution size by 2. The reduction of spatial dimensions in UNets is beneficial for semantic tasks as it subsequently increases the kernels field-of-view at no cost. Note that dilated convolutions (a.k.a 'atrous') with increasing dilation rates cannot be used in the encoder due to the sparse input nature. Though dense convolutions in the encoder imply a dilation of the input manifold (Graham et al., 2018) as shown in Figure 5.3, we argue this is beneficial for 3D semantic completion, given the sparse↦dense nature of the task.

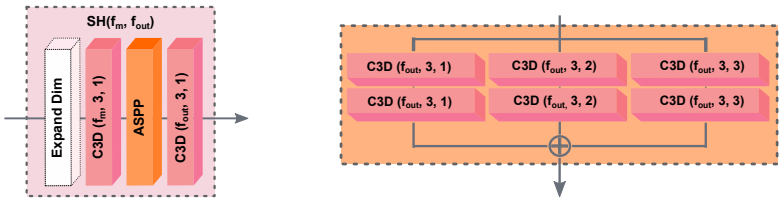
5.2.1.1 2D backbone

As already mentioned, the backbone of our network is composed solely by 2D operations to reduce computation needs. To preserve a lightweight architecture, we use 2D convolutions along the X,Y dimensions, thus turning the height dimension (Z) into a feature dimension. Notice that we directly process 3D data with our 2D UNet in contrast to other 2D/3D works that rely on 2.5D data like depth (Guo and Tong, 2018; Liu et al., 2018) or bird-eye view (Chen et al., 2017). While using 2D convolutions implies loosing 3D spatial connexity, it also enables significantly lighter operations. To further reduce the memory requirements, we keep a minimum number of features in each convolution layer. Along with the standard skip connections, we also enhance information flow in the decoder by concatenating the output of



(a) (b) (c)

Figure 5.3: Dense convolution increases the number of active (i.e. non-zero) sites, thus dilating the manifold. (a) Sparse input data. (b) Result after applying standard 3x3 convolution with constant weights 1/9. (c) Result after applying again that same convolution. Source: Graham et al. (2018)



(a) 3D Seg. Head (b) ASPP block (Liu et al., 2018)

Figure 5.4: (a) 3D segmentation heads are in charge od retrieving the 3D semantic information at multiple scales from the high level features received from the 2D backbone. (b) We use Atrous 3D convolutions – ASPP blocks from (Liu et al., 2018) – to preserve low inference complexity and enlarge receptive fields at low computational cost.

each level to all lower levels. Technically, we upsample coarse feature maps learning ad-hoc deconvolution before concatenation to lower levels, which is shown with purple deconv (aka transpose convolution) blocks in Figure 5.2. Intuitively, this enables our network to use high level features from coarser resolutions, and thus enhance the spatial contextual information.

5.2.1.2 3D segmentation head

Different from other works handling point cloud as bird-eye-view, the task of 3D semantic completion actually requires to retrieve the 3rd dimension "flatten" by the 2D convolutions. In other words, while 2D CNNs output 3D features maps, our decoder must output 4D tensor; the last dimension being the semantic class-wise probability distribution.
To address this, we introduce 3D segmentation heads depicted as red 3D blocks in Figure 5.2 and shown in details in Figure 5.4a. The heads use a series of dense and dilated convolutions. The latter, in the form of Atrous Spatial Pyramid Pooling as shown in Figure 5.4b – aka ASPP (Chen et al., 2018; Liu et al., 2018) –, is beneficial to fuse information from different

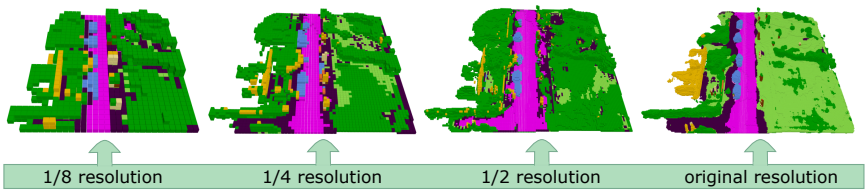


Figure 5.5: Our pipeline enables multiscale reconstruction. To supervise coarser representation, we use majority vote pooling from the original resolution ground truth.

receptive fields thanks to the convolutions with increasing dilation rates (here [1, 2 and 3]). Note that dilated convolutions though light and powerful are not appropriate for sparse inputs and as such cannot be used in the encoder. In our segmentation head, the benefit of preceding ASPP with dense 3D convolutions is dual: a) to further densify the feature maps, b) to ward off features from the segmentation heads and the backbone features. This last property is required to enable multiscale capacity which we now describe.

5.2.2 Multiscale completion

In the same vein as Zhang et al. (2018a); Dai et al. (2020), we aim to output multiscale completion to enable both coarse scene representation and faster scene completion at lower resolution – beneficial for mobile robotics applications. We subsequently attach a 3D segmentation head after each level of the 2D UNet architecture, thus providing outputs at input relative scale of $\frac{1}{2^l}$ $\forall$ $l \in \{0, 1, 2, 3\}$, e.g. half of the original size for $l = 1$. A sample output at different scales is shown in Figure 5.5. As already mentioned, we noticed experimentally the importance of separating the segmentation features from the main features of the 2D backbone, which again justifies the additional 3D convolutions in the segmentation head. The main interests of our multiscale architecture is that it infers semantic scene completion at a desired scale as needed, reducing the computation and memory requirements. This is further analyzed in Section 5.3.3.3.

5.2.3 Training strategy

We train our LMSCNet from scratch in a standard end-to-end fashion from pairs of sparse input voxel (x) and semi-dense semantically labeled voxel grid ($\hat{y}$). It is important to note that in a real setup, a dense ground truth is impractical for scene completion, due to occlusions and sensor field-of-view limitations. As such, the ground truth $\hat{y}$ is *also* sparse and encoded with N+2 classes (N semantic classes, 1 free class, 1 unknown). Similar to others

(Song et al., 2017; Liu et al., 2018; Garbade et al., 2019) we use a sparse loss strategy, backpropagating the gradient only where ground truth is known. For each scale l, we train with a cross-entropy loss defined as :

$$\mathcal{L}_l = -\sum_{c=0}^{N} w_c \hat{y}_{i,c} \log\left(\frac{e^{y_{i,c}}}{\sum_{c'}^{N} e^{y_{i,c'}}}\right), \tag{5.1}$$

where y is the network output, i a voxel index, and $\hat{y}_{i,c}$ a one-hot vector (i.e. $\hat{y}_{i,c} = 1$ if voxel i is labeled class c, otherwise $\hat{y}_{i,c} = 0$). Note that semantic tasks are by nature highly class-imbalanced problems. This is especially true in outdoor settings, which causes the prevalence of classes like road or vegetation (refer to Figure 4.3b of Chapter 4). We account for the class-imbalance nature in Equation 5.1 by weighting each class loss according to the inverse of the class-frequency f_c as in Milioto et al. (2019), thus using $w_c = \frac{1}{\log(f_c+\epsilon)}$ (with $\epsilon \ll 1$). Finally, the complete network loss is a weighted sum of all level losses[1] and writes:

$$\mathcal{L} = \sum_{l=0}^{3} \alpha_l \mathcal{L}_l \,, \tag{5.2}$$

where α_l is the per-level loss weight, written for generality, though we use $\alpha_l = 1, \forall l$ which works well and preserves multiscale capacity. Note that some of our choices were guided by faster training or inference speed. For example, unlike Zhang et al. (2018a, 2019b); Dourado et al. (2020a); Dai et al. (2020); Song et al. (2017), we avoid using Truncated Signed Distance Function variants (TSDF) that require a greedy computation time and was found to be of little benefit (Garbade et al., 2019; Behley et al., 2019). We also tried to encode input as N+2 classes, that is with *unknown* class, but we noticed little improvement – if any – at the cost of a large pre-processing time for ray casting.

5.3 Experiments

We implement our method using PyTorch and evaluate its performance in both indoor – NYUv2 (Silberman et al., 2012) – and outdoor – SemanticKITTI (Behley et al., 2019), nuScenes (Caesar et al., 2020) – scenes. We refer to Section 4.3.1 of Chapter 4 for further details on each dataset.

Outdoor scenes. We evaluate our LMSCNet method by training on the recent semantic scene completion benchmark from SemanticKITTI (Behley et al., 2019) providing 3D voxel grids from semantically labeled scans of a

[1] In Equation 5.2, losses from heterogeneous resolutions can be summed due to the ad-hoc normalization in Equation 5.1

Velodyne HDL-64E rotating LiDAR in outdoor urban scenes (Geiger et al., 2013). In Behley et al. (2019), the inputs are voxelized single scans, while the ground truth was obtained from the voxelized aggregation of successive registered scans (refer to Section 4.3.1 of previous chapter). Grids are 256×256×32 with 0.2m voxel size, and it is important to note that input *and* ground truth are sparse, with average density of 6.7% and 65.8%, respectively. We additionally perform qualitative evaluations of our method in nuScenes (Caesar et al., 2020), by voxelizing LiDAR scenes with the same dimensions as the ones from the SemanticKITTI benchmark.

Indoor scenes. We also evaluate our method in NYUv2 (Silberman et al., 2012) composed mainly of office and house room scenery. For evaluation we use the voxelized input volumes of dimensions 240×144×240 with voxel size 0.02 m, and ground truth volumes at a fourth of the input size as generated in (Song et al., 2017). Given that synthetic SUNCG dataset (Song et al., 2017) is no longer available, we train our network solely on NYUv2 for indoor scenes.

5.3.1 Metrics

We use standard mIoU as a semantic completion metric, measuring the intersection over union averaged over all classes (N semantic classes + *free*). The number of semantic classes corresponds to 20 and 12 for SemanticKITTI and NYUv2 datasets, respectively. Additionally, we consider completion metrics IoU, Precision, and Recall to provide a sense of the scene completion quality, regardless of the assigned semantic labels (i.e. considering the binary *free* / *occupied* setting). We highlight that completion is crucial for obstacle avoidance in mobile robotics.

For outdoor scenes obtained from LiDAR range scans, both IoU and mIoU metrics are calculated in all the voxels that are known in the groundtruth. Conversely, for indoor scenes scanned with RGB-D sensors, semantic completion (mIoU) is evaluated in the known surface and occluded space, while occupancy completion (IoU) is evaluated in the occluded space only. We refer to Section 4.2 of previous chapter for more details and justification.

5.3.2 Implementation details

For SemanticKITTI we use the original train/val splits with 3834/815 grids (Behley et al., 2019), adding x-y flipping augmentation for generalization. Similarly, we use original train/val splits of NYUv2 defined in Song et al. (2017) with 795/654 grids. In both cases we use Adam optimizer for training our network ($\beta_1 = 0.9$, $\beta_2 = 0.999$) with learning rate of 0.001 scaled by 0.98^{epoch}. Training fits in a single 11GB GPU with batch size 4 for SemanticKITTI and takes around 48 hours to converge (80 epochs). With the

same GPU configuration, training takes around 3 hours to converge with batch size 16 in the considerably smaller NYUv2 dataset by considering the same number of epochs for training.

5.3.3 Performance evaluation

5.3.3.1 Semantic Scene Completion

Outdoor scenes. We report performance on the *hidden* SemanticKITTI test set in Table 5.1. The evaluation was conducted on the official server, hence with the full size ground truth. We ranked first in occupancy completion and second in semantic completion against published methods at the time of submission. Performance was initially reported against four state-of-the-art methods: SSCNet (Song et al., 2017), TS3D (Garbade et al., 2019), TS3D+DNet (Behley et al., 2019) and finally TS3D+DNet+SATNet (Behley et al., 2019). Because SSCNet output is 4x downsampled, we also reported performance using deconvolution to reach full input resolution, hereafter denoted SSCNet-full. Approaches posterior to our submission include JS3C-Net (Yan et al., 2021) and S3CNet (Cheng et al., 2020), both based on sparse encoder-decoder architectures (Graham et al., 2018), and Local-DIFs (Rist et al., 2020a) using deep implicit functions through a point-based feature extractor. We first detail semantic completion performance and then demonstrate the speed and lightness of our architecture.

Overall, our network performed on par with best methods at submission time, ranking 2nd on the semantic completion metric (mIoU), though more recent methods outperform us. However, we highlight that our method is significantly lighter and faster than all other approaches (refer to Section 5.3.3.4). Note also that TS3D uses additional RGB input, and all TS3D+DNet use also LiDAR refraction intensity. Furthermore, all post-submission methods use the more informative input point cloud rather than the discretized voxelized grids provided by the SemanticKITTI benchmark. Despite this, our method still performs 2nd among all methods on the completion metric (IoU) and outperforms all methods previous to submission by a comfortable margin. Again, completion is of high importance for practical mobile robotics applications. The highly imbalanced class frequencies (shown in parenthesis in Table 5.1) illustrate the task complexity. Specifically, we outperformed existing methods at submission time on the largest four classes but performed on par or lower on the rest, which advocates for the need of improvement in our balancing strategy. S3CNet (Cheng et al., 2020) shows outstanding performance in rare classes, more than twice when compared to next best classed scores. Regrettably, the reason for such behavior is not deeply explored in their work.

Approach	scene completion			semantic scene completion																			
	precision	recall	IoU	road (15.30%)	sidewalk (11.13%)	parking (1.12%)	other-ground (0.56%)	building (14.1%)	car (3.92%)	truck (0.16%)	bicycle (0.03%)	motorcycle (0.03%)	other-vehicle (0.20%)	vegetation (39.3%)	trunk (0.51%)	terrain (9.17%)	person (0.07%)	bicyclist (0.07%)	motorcyclist (0.05%)	fence (3.90%)	pole (0.29%)	traffic-sign (0.08%)	mIoU
Pre-submission																							
SSCNet (Song et al., 2017)	31.7	83.4	29.8	27.6	17.0	15.6	6.0	20.9	10.4	1.8	0	0	0.1	25.8	11.9	18.2	0	0	0	14.4	7.9	3.7	9.5
*SSCNet-full (Song et al., 2017)	59.6	75.5	50.0	51.2	30.8	27.1	6.4	34.5	24.3	1.2	0.5	0.8	4.3	35.3	18.2	29.0	0.3	0.3	0	19.9	13.1	6.7	16.1
TS3D (Garbade et al., 2019)	31.6	84.2	29.8	28.0	17.0	15.7	4.9	23.2	10.7	2.4	0	0	0.2	24.7	12.5	18.3	0	0.1	0	13.2	7.0	3.5	9.5
TS3D+DNet (Garbade et al., 2019)	25.9	88.3	25.0	27.5	18.5	18.9	6.6	22.1	8.0	2.2	0.1	0	4.0	19.5	12.9	20.2	2.3	0.6	0	15.8	7.6	7.0	10.2
TS3D+DNet+SATNet (Garbade et al., 2019)	80.5	57.7	50.6	62.2	31.6	23.3	6.5	34.1	30.7	4.9	0	0	0.1	40.1	21.9	33.1	0	0	0	24.1	16.9	6.9	17.7
Our submission																							
LMSCNet	77.1	66.2	55.3	64.0	33.1	24.9	3.2	38.7	29.5	2.5	0	0	0.1	40.5	19.0	30.8	0	0	0	20.5	15.7	0.5	17.0
LMSCNet-singlescale	81.6	65.1	56.7	64.8	34.7	29.0	4.6	38.1	30.9	1.5	0	0	0.8	41.3	19.9	32.1	0	0	0	21.3	15.0	0.8	17.6
Post-submission																							
S3CNet (Cheng et al., 2020)	-	-	45.6	42.0	22.5	17.0	7.9	50.2	31.2	6.7	41.5	45.0	16.1	39.5	34.0	21.2	45.9	35.8	16.0	31.3	31.0	24.3	29.5
JS3C-Net (Yan et al., 2021)	71.5	73.5	56.6	64.7	39.9	34.9	14.1	39.4	33.3	7.2	14.4	8.8	12.7	43.1	19.6	40.5	8.0	5.1	0.4	30.4	18.9	15.9	23.8
Local-DIFs (Rist et al., 2020a)	-	-	57.7	67.9	42.9	40.1	11.4	40.4	34.8	4.4	3.6	2.4	4.8	42.2	26.5	39.1	2.5	1.1	0	29.0	21.3	17.5	22.7

* Own implementation, derived from Song et al. (2017).

Table 5.1: Comparison of published methods on the official SemanticKITTI (Behley et al., 2019) benchmark (hidden test set). Best results at submission time are highlighted on blue and best post-submission results in red.

In addition to the multiscale proposal (LMSCNet), we also report results for LMSCNet-singlescale – a variation of LMSCNet where we train with $\mathcal{L} = \mathcal{L}_0$ –, which logically performs a little better at full size though at the cost of loosing crucial multiscale capacity.

Indoor scenes. While it was not included in our initial research (Roldão et al., 2020), for completeness of the thesis we evaluate LMSCNet on indoor scenes. Importantly, one has to note that LMSCNet using 2D backbone is inherently more adapted to outdoor scenes having lower variance along the 3rd axis. Table 4.5 lists methods trained solely on NYUv2 for fair comparison. As expected, LMSCNet performs average in mIoU among depth-only methods (28.4% vs 41.1% best), although being 2nd in IoU (62.2% vs 73.4%). When using additional texture modality (i.e. RGB-D) all methods outperform LMSCNet on mIoU, though we still rank 3rd in IoU. Given common practice, all methods produce output at a fourth of the input resolution. Therefore, we train only a section of our UNet architecture for single scale predictions at 1/4 of the input size.

5.3.3.2 Qualitative performance

We compare qualitatively full size outputs of our LMSCNet and SSCNet-full in Figure 5.6, with view pairs from 4 scenes of the SemanticKITTI validation set[2]. At the rightmost, ground truth visualization also illustrates the sparse supervision complexity since holes are still visible. Our method produces visually smoother semantic labels, easily noticeable in rows 5-8, and is able to reconstruct thin structures, like trees or cars (rows 6 or 7). For a comprehensive analysis, we further test the same model (trained on 64-layer LiDAR from SemanticKITTI) on the popular nuScenes dataset (Caesar et al., 2020), which uses a 32-layers LiDAR. Figure 5.8 shows that our network better adjusts to the change of density and maintains the smoothness in the reconstruction. Qualitative results on a section of the validation set sequence of SemanticKITTI can be found on our demo video `https://youtu.be/J6dYoWx4Xqw`.

We also show qualitative results for indoor scenes on the NYUv2 validation set in Figure 5.7. We compare our single-scale architecture against best ranked method 3DSketch (Chen et al., 2020a). We use our own implementation of their architecture for obtaining the visual results. It can be observed that our method performs a more noisy semantic reconstruction, specially in large areas such as the walls. Similarly, LMSCNet struggles to predict the windows in all samples. This is expected as they are hard to detect without color information.

[2]Note that SemanticKITTI benchmark (i.e. test set) does not provide any visual results. Hence, we omit TS3D baselines due to retraining complexities and their use of additional modalities (RGB or LiDAR intensity).

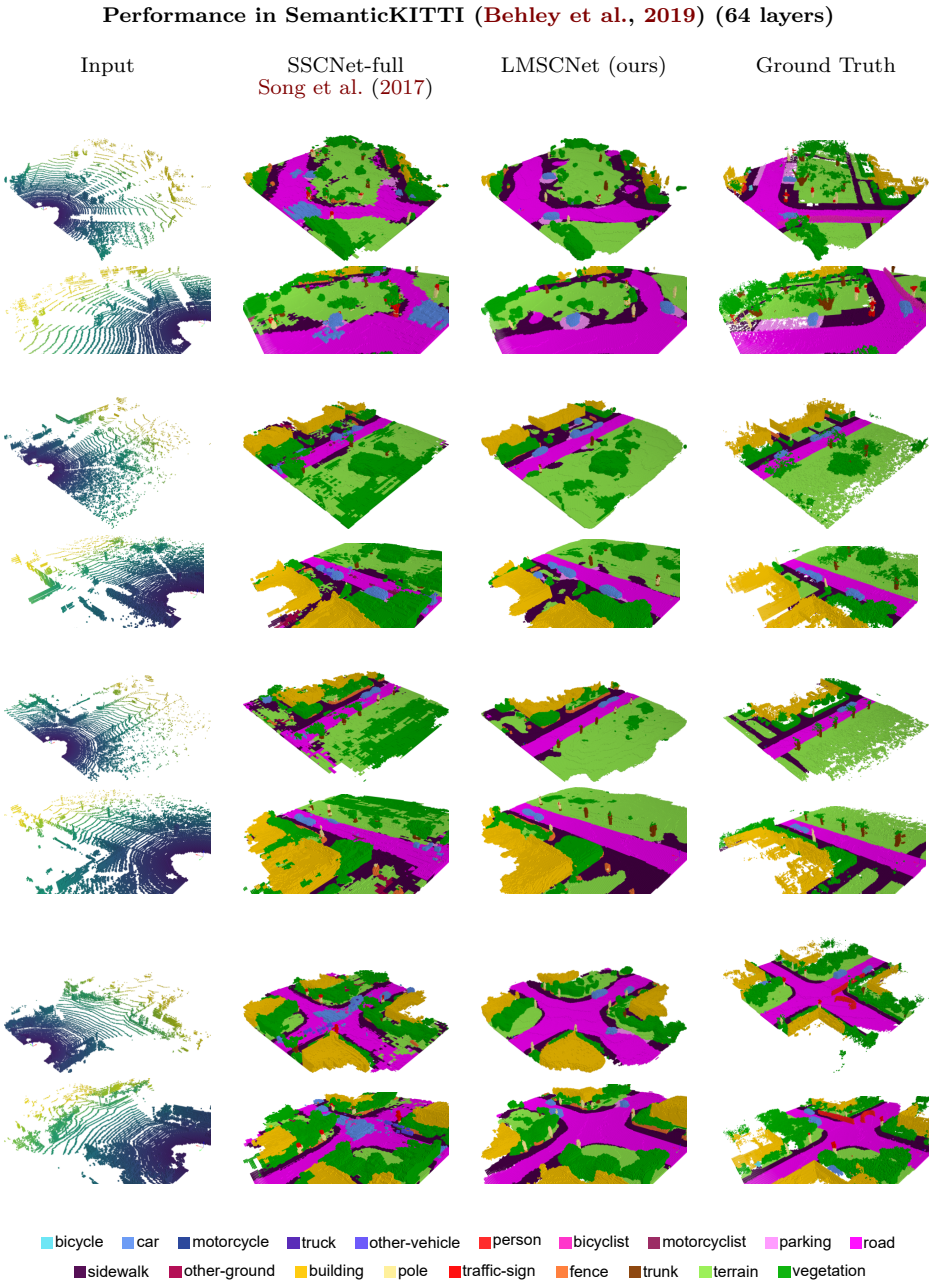
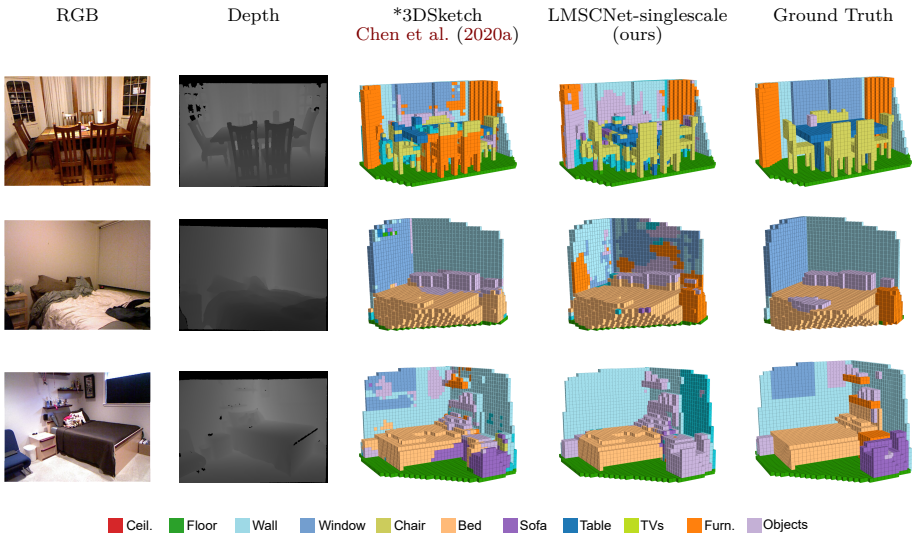


Figure 5.6: Qualitative 3D semantic completion at full size on the SemanticKITTI (Behley et al., 2019) validation set. Each pairs of rows show a single scene with different viewpoints. Compared to SSCNet-full (Song et al., 2017), our LMSCNet provides smoother semantics labels and is capable of retrieving finer details. This is evident when looking at the cars (rows 7-8) or the trees (rows 5-6).

Method	input	scene completion			semantic scene completion											
		precision	recall	IoU	ceil. (0.74%)	floor (12.44%)	wall (9.67%)	win. (2.12%)	chair (2.03%)	bed (9.17%)	sofa (6.78%)	table (4.14%)	tvs (0.53%)	furn. (36.64%)	objs. (15.74%)	mIoU
Pre-submission																
SSCNet (Song et al., 2017)	G	57.0	94.5	55.1	15.1	94.7	24.4	0	12.6	32.1	35.0	13.0	7.8	27.1	10.1	24.7
ESSCNet (Zhang et al., 2018a)	G	71.9	71.9	56.2	17.5	75.4	25.8	6.7	15.3	53.8	42.4	11.2	0	33.4	11.8	26.7
TS3D (Garbade et al., 2019)	G+T	-	-	60.0	9.7	93.4	25.5	21.0	17.4	55.9	49.2	17.0	27.5	39.4	19.3	34.1
DDRNet (Li et al., 2019)	G+T	71.5	80.8	61.0	21.1	92.2	33.5	6.8	14.8	48.3	42.3	13.2	13.9	35.3	13.2	30.4
SSC-GAN (Chen et al., 2019b)	G	63.1	87.8	57.8	-	-	-	-	-	-	-	-	-	-	-	22.7
AM^{2}FNet (Chen et al., 2019a)	G+T	72.1	80.4	61.3	19.3	92.6	26.1	11.1	19.1	51.9	47.0	16.7	14.9	35.9	14.0	31.7
GRFNet (Liu et al., 2020)	G+T	68.4	85.4	61.2	24.0	91.7	33.3	19.0	18.1	51.9	45.5	13.4	13.3	37.3	15.0	32.9
CCPNet (Zhang et al., 2019b)	G	74.2	90.8	63.5	23.5	96.3	35.7	20.2	25.8	61.4	56.1	18.1	28.1	37.8	20.1	38.5
AMFNet (Li et al., 2020c)	G+T	67.9	82.3	59.0	16.7	89.2	27.3	19.2	20.2	56.1	50.4	15.1	13.5	36.8	18.0	33.0
PALNet (Li et al., 2020b)	G	68.7	85.0	61.3	23.5	92.0	33.0	11.6	20.1	53.9	48.1	16.2	24.2	37.8	14.7	34.1
3DSketch (Chen et al., 2020a)	G+T	85.0	81.6	71.3	43.1	93.6	40.5	24.3	30.0	57.1	49.3	29.2	14.3	42.5	28.6	41.1
AIC-Net (Li et al., 2020a)	G+T	62.4	91.8	59.2	23.2	90.8	32.3	14.8	18.2	51.1	44.8	15.2	22.4	38.3	15.7	33.3
Our submission																
LMSCNet-singlescale	G	71.5	71.5	62.2	28.3	93.5	32.3	2.5	14.4	44.3	35.1	15.0	0	30.5	16.4	28.4
Post-submission																
IPF-SPCNet (Zhong and Zeng, 2020)	G+T	70.5	46.7	39.0	32.7	66.0	41.2	17.2	34.7	55.3	47.0	21.7	12.5	38.4	19.2	35.1
Chen et al. (Chen et al., 2020b)	G	-	-	73.4	-	-	-	-	-	-	-	-	-	-	-	34.4

Input: Geometry (depth, range, points, etc.), Texture (RGB).

Table 5.2: Comparison of published methods on the NYUv2 (Silberman et al., 2012) dataset. Best results at submission time are highlighted on blue and best post-submission results in red. We only show methods trained solely on NYUv2 for a fair comparison.



* Own implementation to compute predicted scenes.

Figure 5.7: Qualitative 3D semantic completion at a fourth of the input resolution on the NYUv2 (Silberman et al., 2012) validation set. LMSCNet performs correctly though output is noisy when compared to 3DSketch (Chen et al., 2020a). We highlight that 3DSketch additionally uses RGB modality which enables to detect texture salient objects (e.g. windows).

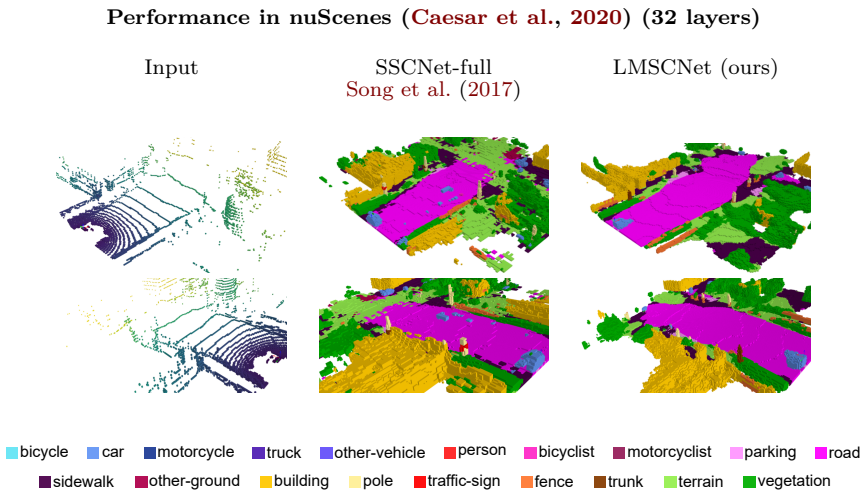


Figure 5.8: Inference results on nuScenes (Caesar et al., 2020) with 32-layers LiDAR, while being trained on 64-layers SemanticKITTI. Our method performs well with sharp scene labeling, despite the change of input density.

LMSCNet scale	IoU	mIoU
1:1 (full size)	54.22	16.78
1:2	56.27	16.78
1:4	59.36	17.19
1:8	65.45	17.37

Table 5.3: LMSCNet multiscale semantic completion performance on SemanticKITTI validation set. We reach similar performance at all levels, even better at the coarsest resolutions .

5.3.3.3 Multiscale performance

Table 5.3 shows multiscale performance of our method on the SemanticKITTI validation set, where the scale is relative to the full size resolution (level 0). From Section 5.2.2, scale at level l is $\frac{1}{2^l}$. Ground truths at lower resolution were obtained from majority vote pooling of the full size ground truth. From the above table, our architecture maintains a good performance in all resolutions, with best performance logically reached at the lowest resolution (highest level). Qualitative multiscale completion is visible in Figure 5.5. We believe that our custom 3D segmentation heads aid in the disentanglement of the 3D segmentation features from the 2D backbone high level features, thus contributing to the multiscale capacity of our architecture. Additionally, at coarser resolution our network reaches very fast inference, which will be described in details in the following section.

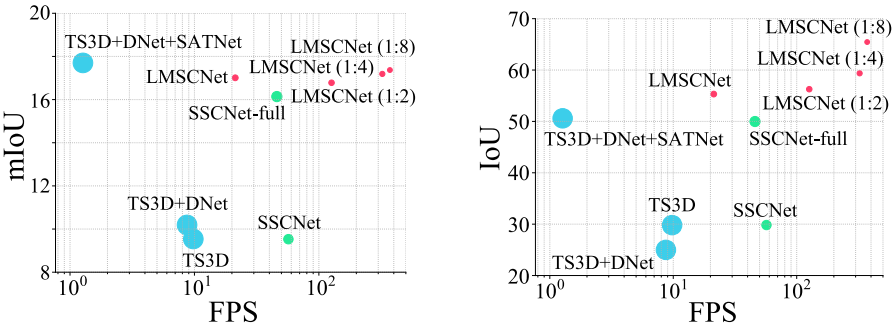


Figure 5.9: Architectures performance versus speed (markers are scaled with # of parameters) of all methods at submission time. Notice that TS3D+DNet+SATNet is the only better method on semantics (+0.69 mIoU) though less time performant (x17 slower) and worse on completion (-4.72 IoU).

5.3.3.4 Architectures comparison

Table 5.4 reports networks statistics for our architecture and all above mentioned baselines at submission time. We only provide the number of parameters of the post-submitted methods as these are the only data provided in their works. All performances were evaluated with same input scene size as the SemanticKITTi benchmark. From the table, even at full size LMSCNet has significantly less parameters (0.35M) and lower computational cost for inference (72.6G FLOPs). Compared to any TS3D baselines, it is at least an order of magnitude faster. However, SSCNet (original or full) is twice faster than LMSCNet, though with more parameters and worse performance (cf. Table 5.4). Since lighter models does not *always* run faster due to the sequentiality of some operations on GPU, we conjecture the higher speed of SSCNet is caused by the lower number of convolutional operations compared to LMSCNet full scale (16 vs. 25).

In last rows of Table 5.4, we report statistics for coarser completion, removing *unnecessary* parts of our network at inference. Lower resolution inference allows significant speedups in the processing, reaching 372 FPS at the highest scale – being 6x faster than SSCNet and 300x faster than TS3D+DNet+SATNet–. Figure 5.9 illustrates the architectures performance versus speed. Notice that *even at full scale* we provide a better speed-performance balance. Because semantic completion is an application of high interest for mobile robotics, like autonomous driving, our lighter architecture is beneficial for embedded GPUs and enables coarse scene analysis at high speed.

Method	Params (M)	FLOPs (G)	FPS
Pre-submission			
*SSCNet (Song et al., 2017)	0.93	82.5	56.90
*SSCNet-full (Song et al., 2017)	1.09	769.6	45.94
*TS3D (Garbade et al., 2019)	43.77	2016.7	9.79
*TS3D+DNet (Behley et al., 2019)	51.31	847.1	8.72
*TS3D+DNet+SATNet (Behley et al., 2019)	50.57	905.2	1.27
Our submission			
LMSCNet	0.35	72.6	21.28
LMSCNet (1:2)	0.32	13.7	126.38
LMSCNet (1:4)	0.28	5.7	323.46
LMSCNet (1:8)	0.24	4.4	372.24
Post-submission			
JS3C-Net (Yan et al., 2021)	3.1	-	-
Local-DIFs (Rist et al., 2020a)	9.9	-	-

* Own implementation to compute network statistics

Table 5.4: Network statistics. Even at full resolution LMSCNet (ours) has significantly less parameters with lower FLOPs. On a speed basis, we are twice slower than SSCNet-full (Song et al., 2017) which performs worse than us (see Table 5.1). Still, our multiscale versions – denoted *LMSCNet (1:x)* – enable very fast inference.

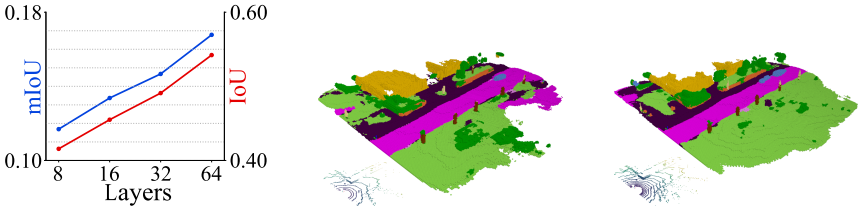
5.3.4 Ablation studies

To study the benefit of our design choices, we conduct a series of ablation studies on SemanticKITTI validation set. This is done by modifying important blocks of our architecture and evaluating its performance.

Influence of input resolution. We evaluate our robustness, by retrieving the original 64-layers KITTI scans used in SemanticKITTI and simulating 8/16/32 layers LiDARs with layers subsampling[3], as in Jaritz et al. (2018).

Figure 5.10 shows quantitative and qualitative performance using simulated and original LiDAR. As expected, lower layers input deteriorate the performance, especially in areas far from the sensor location, but our network still performs reasonably well on semantics (mIoU) and completion (IoU). This is visible in the middle image as 8 layers input (2.10% density) is sufficient to retrieve the general outline of the scene.

[3]Every 2nd, 4th and 8th layer are subsampled to simulate 32, 16 and 8 layer LiDARs, respectively. Unlike Jaritz et al. (2018), note that data SemanticKITTI uses KITTI odometry set in which data is already untwisted.



(a) Quant. Performance (b) 8 layers (c) 16 layers

Figure 5.10: Semantic scene completion results from simulated lower resolution LiDAR sensors (downsampled from 64 layers input). Even with only 8 layers input our LMSCNet correctly predicts the scene outline.

Method	IoU	mIoU
LMSCNet (ours)	54.22	16.78
w/o Deconv	52.79	15.64
w/o ASPP	53.81	16.21
w/o Multiscale UNet	53.54	16.22

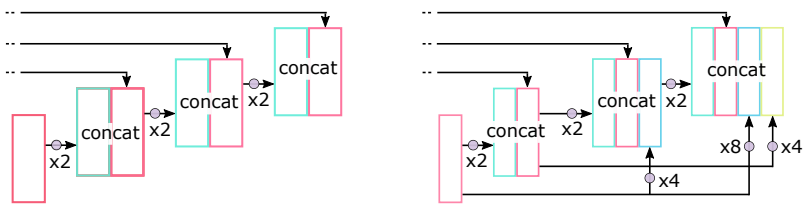
Table 5.5: Ablation study of our model design choices on the SemanticKITTI Behley et al. (2019) validation set.

Deconv versus Upsampling. As we aimed to preserve a lightweight architecture, we tried to remove the parameters-greedy deconv layers from our network (cf. Figure 5.2), replacing them with zero-cost up-sampling layers. From Table 5.5, performance *without deconv* introduces a 1.43% and 1.14% performance drop for completion and semantic completion respectively, with only 3% less parameters.

Dilated convolutions. We evaluate the benefit of dilated convolutions in the decoder by ablating ASPP blocks from the segmentation head (see Figure 5.4). Table 5.5 indicates that mIoU drops by 0.41% without ASPP. We conjecture that the boost of ASSP results come from the increasing receptive fields of the inner dilated convolutions, providing richer features.

Multiscale UNet decoder. As illustrated in Figure 5.11, unlike vanilla UNet decoder we concatenate the features at the end of each decoder level to all other levels. This is intended to aggregate multiscale features and should intuitively help considering coarser semantic features for fine resolutions. Furthermore, the importance of multiscale feature aggregation for SSC is highlighted in Section 4.4.3.2.

We assess the benefit of our *multiscale UNet* by evaluating *Vanilla UNet* in the last row of Table 5.5, which shows that our proposal boosts completion by 0.68% and semantic completion by 0.56%.



(a) Vanilla UNet decoder (b) Multiscale UNet decoder

Figure 5.11: Decoders comparison. While Vanilla UNet decoder only considers features from the previous level (a), we instead use Multiscale UNet where all coarser levels enhance spatial contextual information (b). Circles show intermediary operations to reach required feature maps size.

5.4 Discussion

Review of our work. We believe that our architecture proposal presents an interesting alternative for mobile robotic applications. UNet architectures have been employed by a wide variety of methods for semantic scene completion before (Zhang et al., 2018a; Dourado et al., 2020a; Dai et al., 2020) and after (Wang et al., 2020a; Zhong and Zeng, 2020; Yan et al., 2021; Cheng et al., 2020; Rist et al., 2020a)

However, no other method envisaged the use of 2D convolutions to reduce memory needs. Only the work of Zimmermann et al. (2017) employs a complete 2D architecture for 3D occupancy completion. While this choice produces a lighter architecture, recent post-submission methods using complete 3D architectures through the use of sparse convolutions (Yan et al., 2021; Cheng et al., 2020) or point-based networks (Rist et al., 2020a) have shown better performance.

A limitation of our approach is that the use of additional modalities will importantly increase the architecture complexity as the number of filters for each convolutional layer will be scaled with the number of employed modalities. Furthermore, we believe that scenes with high variance degree in all dimensions affect the performance of our architecture given the spatial connectivity loss by the use of 2D convolutions. This can be observed in our performance shown in NYUv2 (Silberman et al., 2012) dataset.

Data challenges. As the ground truth data from SemanticKITTI (Behley et al., 2019) is generated by accumulating successive LiDAR frames, moving objects lead to traces in the 3D data due to this accumulation over several time steps. Although this creates problems on the evaluation, we show that it does not carry big impact on performance given the larger proportion of parked vehicles on the SemanticKITTI scenes. This can be observed in some of the qualitative results presented in Figure 5.6. The issue could be

solved by removing supervision along the moving agents in the ground truth scenes or using simulation environments such as CARLA (Dosovitskiy et al., 2017). However, recent works have presented similar strategies showing only a slight gain in performance (Rist et al., 2020a; Yan et al., 2021).

5.5 Conclusion

In this chapter we presented a novel method, coined LMSCNet for 3D semantic scene completion, which benefits from mixing 2D/3D convolutions to preserve lightweight architecture, while enabling multiscale inference. Our method has been tested in both indoor and outdoor scenarios trough the use of 3-dimensional scenes from the NYUv2 (Silberman et al., 2012) and the SemanticKITTI (Behley et al., 2019) and nuScenes (Caesar et al., 2020) datasets, respectively.

On the challenging SemanticKITTI benchmark, our method performed on par with state of the art approaches for semantic completion with a much lighter architecture and at faster inference speeds. For completion, our method outperformed all approaches at the time of submission. Results show that the loss of 3D spatial connexity caused by the 2D backbone of our architecture does not impair performance. Our proposal is robust to much lower input density LiDARs without dramatic drops on performance. Results show that even at very low resolution LiDAR (8 layers), our method is still capable to correctly predict the outline of the scene.

We additionally show generalization of our architecture by testing it on unseen urban scenes of the nuScenes dataset, which considerably differ from the SemanticKITTI scenes used for training. We highlight the impressive performance of learning based methods enabling to complete entire scenes despite occlusions and missing information, in contrast to traditional methods considered in the first part of this thesis. We encourage the use of our method and development of more advanced scene completion methods for further applications such as road detection, obstacle detection and tracking, and ultimately, vehicle navigation.

Chapter 6

Conclusion

6.1 Contributions

In this thesis we presented different techniques to generate a 3D model of the environment from sparse LiDAR point clouds intended for autonomous driving applications. This was achieved through the use of both traditional computer vision techniques relying on geometrical and physical priors, and deep learning techniques relying on data-driven priors.

In the first part of the thesis we investigated the use of occupancy grids for multi-frame aggregation of LiDAR scans and proposed to take advantage of the ray-casting operation to analyze the distance traversed by the rays within each voxel to adjust the occupancy probability update. Furthermore, we introduced a new weighting modeling to account for the density of observations at a given cell and avoid overconfidence at low density regions. Experiments showed that our proposed method reduces inaccuracies present in partially occupied cells for both real and simulated data. The benefit of 3D maps representation is shown useful for scene reconstruction, localization and mapping applications.

We also proposed a pipeline to calculate a continuous surface representation of the surroundings from the input point cloud. Our method is robust against heterogeneous density data inherent to LiDAR sensors. This is achieved through the use of an adaptive neighborhood strategy that adapts to the point density. The surface representation is of high interest for physical modeling or fluid simulation. We showed through our experiments that our method achieves an accurate reconstruction at close surroundings and is capable to complete small areas of missing data through interpolation which results from the use of a gradient distance field calculated from the local explicit planar estimation.

In the second part of the thesis, we switched to contemporary deep learning methods to complete and semantically label sparse 3D scenes through the semantic scene completion task. Given little consensus and the wide variety of approaches existing in the literature, we presented the first in-depth survey to highlight the most important points to consider for the topic. We also compared performance of state-of-the-art approaches in both indoor and outdoor datasets.

Finally we presented LMSCNet, a lightweight multiscale semantic completion network from a voxelized input point cloud. Our method employs

a 2D CNN backbone to reduce memory and computation overhead by encoding one of the spatial dimensions as a feature dimension. Custom 3D segmentation heads predict the semantic 3D scene at different resolutions thanks to our UNet design architecture. Our finding shows that the loss of 3D spatial connexity caused by our 2D backbone *does not* impair the performance. Moreover, our architecture achieves real-time inference at different resolutions, being significantly lighter and faster that all other approaches proposed in the literature on the SemanticKITTI dataset. The robustness of our method was proved in low resolution point clouds. Furthermore, our experiments showed good generalization into the nuScenes dataset.

6.2 Future work

Unlike Part I of this thesis, recent works now use deep learning for reconstruction (Groueix et al., 2018; Park et al., 2019), mostly relying on intermediate representations as TSDFs or voxel occupancy (Dai et al., 2017b; Park et al., 2019; Mescheder et al., 2019; Chen and Zhang, 2019). Of interest, DeepMarchingCubes (Liao et al., 2018) showed that it is possible to perform the task end-to-end and predict sub-voxel accurate 3D shapes of arbitrary topology. However, the approach requires large memory needs and is limited to low resolution outputs.

This limitation typically affects all 3D CNN based algorithms. Further research in optimization techniques such as sparse convolutions (Graham et al., 2018) or hierarchical data structures like octrees (Riegler et al., 2017b; Wang et al., 2018b) to deal with sparse 3D data is encouraged. Alternately, point-based networks (Qi et al., 2017b,a; Li et al., 2018; Wang et al., 2019b; Thomas et al., 2019; Richard et al., 2020) have gained momentum but are still limited for reconstruction and completion of large scenes and have been directly applied only to completion of single objects (Yuan et al., 2018; Tchapmi et al., 2019; Wang et al., 2020c; Wen et al., 2020).

Another interesting line of research would consist in the use of monocular images (RGB) to perform semantic scene completion by relying on 3D data for self-supervision as it is done for depth estimation (Fu et al., 2018; Gur and Wolf, 2019). Recent works have already shown that it is possible to learn geometry from 2D images (Fan et al., 2017; Wang et al., 2018a; Yin et al., 2020). These methods would present an interesting alternative to the considerably more expensive LiDAR sensors.

Finally, while we mostly studied frame-wise scene reconstruction and completion, it would be interesting to exploit large input sequences to fuse information across the temporal domain to improve performance. Semantic scene reconstruction and completion from multiple scans has been recently presented in Wu et al. (2020). This could provide meaningful temporal information such as speed and intentions of moving agents in the scene.

Publications

This thesis led to the following publications:

- **Roldão, L.**, de Charette, R. and Verroust-Blondet, A.
  A statistical update of grid representations from range sensors.
  arXiv 2018.

- Ravi Kiran, B., **Roldão, L.**, Irastorza, B., Verastegui, R., Suss, S., Yogamani, S., Talpaert, V., Lepoutre, A., and Trehard, G.
  Real-time dynamic object detection for autonomous driving using prior 3d-maps
  ECCV Workshop 2018.

- **Roldão, L.**, de Charette, R. and Verroust-Blondet, A.
  3D Surface reconstruction from voxel-based lidar data.
  ITSC 2019.

- **Roldão, L.**, de Charette, R. and Verroust-Blondet, A.
  LMSCNet: Lightweight Multiscale 3D Semantic Completion.
  3DV 2020.

- **Roldão, L.**, de Charette, R. and Verroust-Blondet, A.
  3D Semantic Scene Completion: a Survey.
  ArXiv 2021 - Submitted to IJCV.

Bibliography

Abbasi, A., Kalkan, S., and Sahillioglu, Y. (2018). Deep 3D semantic scene extrapolation. *The Visual Computer 2018*.

Adarve, J., Perrollaz, M., Makris, A., and Laugier, C. (2012). Computing occupancy grids from multiple sensors using linear opinion pools. In *ICRA 2012*.

Ahmed, E., Saint, A., Shabayek, A. E. R., Cherenkova, K., Das, R., Gusev, G., Aouada, D., and Ottersten, B. (2018). A survey on deep learning advances on different 3D data representations. *arXiv 2018*.

Amenta, N. and Bern, M. (1998). Surface reconstruction by voronoi filtering. In *SCG 1998*.

Amenta, N., Choi, S., and Kolluri, R. K. (2001). The power crust. In *SMA 2001*.

Armeni, I., Sax, S., Zamir, A., and Savarese, S. (2017). Joint 2D-3D-semantic data for indoor scene understanding. *ArXiv 2017*.

Aspert, N., Cruz, D., and Ebrahimi, T. (2002). MESH: measuring errors between surfaces using the hausdorff distance. *ICME 2002*.

Bares, J., Hebert, M., Kanade, T., Krotkov, E., Mitchell, T. M., Simmons, R., and Whittaker, W. (1989). Ambler: an autonomous rover for planetary exploration. *IEEE Computer 1989*.

Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Philos. Trans. R. Soc. 1763*.

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., and Gall, J. (2019). SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In *ICCV 2019*.

Behley, J. and Stachniss, C. (2018). Efficient surfel-based SLAM using 3D laser range data in urban environments. In *RSS 2018*.

Bennewitz, M., Stachniss, C., Behnke, S., and Burgard, W. (2009). Utilizing reflection properties of surfaces to improve mobile robot localization. In *ICRA 2009*.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *CACM 1975*.

Berger, J. O. (1988). Statistical decision theory and bayesian analysis. *Springer: Series in Statistics*.

Berger, M., Tagliasacchi, A., Seversky, L. M., Alliez, P., Guennebaud, G., Levine, J. A., Sharf, A., and Silva, C. T. (2017). A survey of surface reconstruction from point clouds. *CGF 2017*.

Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C. T., and Taubin, G. (1999). The ball-pivoting algorithm for surface reconstruction. *TVCG 1999*.

Bhoi, A. (2019). Monocular depth estimation: A survey. *ArXiv 2019*.

Boltcheva, D. and Lévy, B. (2017). Surface reconstruction by computing restricted voronoi cells in parallel. *CAD 2017*.

Bouchiba, H., Santoso, S., Deschaud, J.-E., Rocha-Da-Silva, L., Goulette, F., and Coupez, T. (2020). Computational fluid dynamics on 3D point set surfaces. *J. Comput. Phys. 2020*.

Boulch, A., Guerry, J., Saux, B. L., and Audebert, N. (2018). SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks. *Comput. & Graph. 2018*.

Boulch, A. and Marlet, R. (2016). Deep learning for robust normal estimation in unstructured point clouds. *CGF 2016*.

Boulch, A., Saux, B. L., and Audebert, N. (2017). Unstructured point cloud semantic labeling using deep segmentation networks. In *Eurographics Workshop 2017*.

Bresson, G., Alsayed, Z., Yu, L., and Glaser, S. (2017). Simultaneous localization and mapping: A survey of current trends in autonomous driving. *T-IV 2017*.

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2020). nuScenes: A multimodal dataset for autonomous driving. In *CVPR 2020*.

Canny, J. (1986). A computational approach to edge detection. *TPAMI 1986*.

Cazals, F. and Giesen, J. (2006). Delaunay triangulation based surface reconstruction: Ideas and algorithms. *Effective Comput. Geom. for Curves and Surfaces 2006*.

Chang, A. X., Dai, A., Funkhouser, T. A., Halber, M., Nießner, M., Savva, M., Song, S., Zeng, A., and Zhang, Y. (2017). Matterport3D: Learning from RGB-D data in indoor environments. In *3DV 2017*.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. (2015). ShapeNet: An information-rich 3D model repository. *ArXiv 2015*.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2018). DeepLab: Semantic image segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *TPAMI 2018*.

Chen, R., Huang, Z., and Yu, Y. (2019a). AM2FNet: Attention-based multiscale & multi-modality fused network. *ROBIO 2019*.

Chen, X., Lin, K.-Y., Qian, C., Zeng, G., and Li, H. (2020a). 3D sketch-aware semantic scene completion via semi-supervised structure prior. In *CVPR 2020*.

Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. (2017). Multi-view 3D object detection network for autonomous driving. In *CVPR 2017*.

Chen, X., Xing, Y., and Zeng, G. (2020b). Real-time semantic scene completion via feature aggregation and conditioned prediction. In *ICIP 2020*.

Chen, Y., Garbade, M., and Gall, J. (2019b). 3D semantic scene completion from a single depth image using adversarial training. In *ICIP 2019*.

Chen, Z. and Zhang, H. (2019). Learning implicit fields for generative shape modeling. In *CVPR 2019*.

Cheng, R., Agia, C., Ren, Y., Li, X., and Bingbing, L. (2020). S3CNet: A sparse semantic scene completion network for LiDAR point clouds. In *CoRL 2020*.

Cherabier, I., Schönberger, J. L., Oswald, M., Pollefeys, M., and Geiger, A. (2018). Learning priors for semantic 3D reconstruction. In *ECCV 2018*.

Choy, C., Gwak, J., and Savarese, S. (2019). 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In *CVPR 2019*.

Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *SIGGRAPH 1996*.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T. A., and Nießner, M. (2017a). ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR 2017*.

Dai, A., Diller, C., and Nießner, M. (2020). SG-NN: Sparse generative neural networks for self-supervised scene completion of RGB-D scans. In *CVPR 2020*.

Dai, A., Qi, C. R., and Nießner, M. (2017b). Shape completion using 3D-encoder-predictor CNNs and shape synthesis. In *CVPR 2017*.

Dai, A., Ritchie, D., Bokeloh, M., Reed, S., Sturm, J., and Nießner, M. (2018). ScanComplete: Large-scale scene completion and semantic segmentation for 3D scans. In *CVPR 2018*.

Davis, J., Marschner, S., Garr, M., and Levoy, M. (2002). Filling holes in complex surfaces using volumetric diffusion. *3DPVT 2002*.

De Charette, R. and Manitsaris, S. (2019). 3D reconstruction of deformable revolving object under heavy hand interaction. *arXiv 2019*.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM. *JRSS 1977*.

Denninger, M. and Triebel, R. (2020). 3D scene reconstruction from a single viewport. In *ECCV 2020*.

Dia, R. (2020). Towards environment perception using integer arithmetic for embedded application. *PhD Thesis, Université Grenoble Alpes, 2020*.

Dia, R., Mottin, J., Rakotovao, T. A., Puschini, D., and Lesecq, S. (2017). Evaluation of occupancy grid resolution through a novel approach for inverse sensor modeling. In *IFAC 2017*.

Dong, P. and Chen, Q. (2017). LiDAR remote sensing and applications. *Taylor & Francis Series Remote Sens. Appl.*

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An open urban driving simulator. In *CoRL 2017*.

Dourado, A., de Campos, T. E., Kim, H. S., and Hilton, A. (2020a). EdgeNet: Semantic scene completion from RGB-D images. *ICPR 2020*.

Dourado, A., Kim, H., de Campos, T. E., and Hilton, A. (2020b). Semantic scene completion from a single 360-Degree image and depth map. In *VISIGRAPP 2020*.

Einhorn, E., Schröter, C., and Groß, H. (2011). Finding the adequate resolution for grid mapping - cell sizes locally adapting on-the-fly. In *ICRA 2011*.

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *IEEE Computer 1989*.

Engelmann, F., Rematas, K., Leibe, B., and Ferrari, V. (2021). From points to multi-object 3D reconstruction. In *CVPR 2021*.

Everingham, M., Eslami, S., Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2014). The Pascal visual object classes challenge: A retrospective. *IJCV 2014*.

Fan, H., Su, H., and Guibas, L. (2017). A point set generation network for 3D object reconstruction from a single image. In *CVPR 2017*.

Fan, R., Wang, H., Cai, P., and Liu, M. (2020). SNE-RoadSeg: Incorporating surface normal information into semantic segmentation for accurate freespace detection. In *ECCV 2020*.

Firman, M. (2016). RGB-D datasets: Past, present and future. In *CVPR Workshop 2016*.

Firman, M., Aodha, O. M., Julier, S. J., and Brostow, G. J. (2016). Structured prediction of unobserved voxels from a single depth image. In *CVPR 2016*.

Fischler, M. and Bolles, R. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *CACM 1981*.

Fleishman, S., Cohen-Or, D., and Silva, C. T. (2005). Robust moving least-squares fitting with sharp features. In *SIGGRAPH 2005*.

Frisken, S., Perry, R. N., Rockwood, A., and Jones, T. (2000). Adaptively sampled distance fields: a general representation of shape for computer graphics. In *SIGGRAPH 2000*.

Fu, H., Cai, B., Gao, L., Zhang, L.-X., Li, C., Xun, Z., Sun, C., Fei, Y., Zheng, Y., Li, Y., Liu, Y., Liu, P., Ma, L., Weng, L., Hu, X., Ma, X., Qian, Q., Jia, R., Zhao, B., and Zhang, H. (2020). 3D-FRONT: 3D furnished rooms with layOuts and semaNTics. *ArXiv 2020*.

Fu, H., Gong, M., Wang, C., Batmanghelich, K., and Tao, D. (2018). Deep ordinal regression network for monocular depth estimation. In *CVPR 2018*.

Fuentes-Pacheco, J., Ascencio, J. R., and Rendón-Mancha, J. M. (2012). Visual simultaneous localization and mapping: a survey. *Artif. Intell. 2012*.

Gaidon, A., Wang, Q., Cabon, Y., and Vig, E. (2016). VirtualWorlds as proxy for multi-object tracking analysis. In *CVPR 2016*.

Gao, B., Pan, Y., Li, C., Geng, S., and Zhao, H. (2020). Are we hungry for 3D LiDAR data for semantic segmentation? *ArXiv 2020*.

Garbade, M. (2019). Semantic segmentation and completion of 2D and 3D scenes. *PhD Thesis, University of Bonn, 2019*.

Garbade, M., Sawatzky, J., Richard, A., and Gall, J. (2019). Two stream 3D semantic scene completion. In *CVPR Workshop 2019*.

Garg, R., Wadhwa, N., Ansari, S., and Barron, J. (2019). Learning single camera depth estimation using dual-pixels. In *ICCV 2019*.

Garg, S., Sünderhauf, N., Dayoub, F., Morrison, D., Cosgun, A., Carneiro, G., Wu, Q., Chin, T., Reid, I. D., Gould, S., Corke, P., and Milford, M. (2020). Semantics for robotic mapping, perception and interaction: A survey. *Found. Trends Robot. 2020*.

Gartshore, R., Aguado, A., and Galambos, C. (2002). Incremental map building using an occupancy grid for an autonomous monocular robot. In *ICARCV 2002*.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *IJRR 2013*.

Geiger, A. and Wang, C. (2015). Joint 3D object and layout inference from a single RGB-D image. In *GCPR 2015*.

Gibson, S. (1998). Using distance maps for accurate surface representation in sampled volumes. In *SIGGRAPH 1998*.

Gopi, M., Krishnan, S., and Silva, C. T. (2000). Surface reconstruction based on lower dimensional localized delaunay triangulation. *CGF 2020*.

Graham, B., Engelcke, M., and van der Maaten, L. (2018). 3D semantic segmentation with submanifold sparse convolutional networks. In *CVPR 2018*.

Griffiths, D. and Boehm, J. (2019). Synthcity: A large scale synthetic point cloud. *ArXiv 2019*.

Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M. (2018). AtlasNet: A Papier-Mâché approach to learning 3D surface generation. In *CVPR 2018*.

Guedes, A. B. S., de Campos, T. E., and Hilton, A. (2018). Semantic scene completion combining colour and depth: preliminary experiments. *ArXiv 2018*.

Guerrero, P., Kleiman, Y., Ovsjanikov, M., and Mitra, N. (2018). PCPNet learning local shape properties from raw point clouds. *CGF 2018*.

Guo, R. and Hoiem, D. (2013). Support surface prediction in indoor scenes. *ICCV 2013*.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., and Bennamoun, M. (2020). Deep learning for 3D point clouds: A survey. *TPAMI 2020*.

Guo, Y.-X. and Tong, X. (2018). View-Volume network for semantic scene completion from a single depth image. In *IJCAI 2018*.

Gupta, S., Girshick, R. B., Arbeláez, P., and Malik, J. (2014). Learning rich features from RGB-D images for object detection and segmentation. In *ECCV 2014*.

Gur, S. and Wolf, L. (2019). Single image depth estimation trained via depth from defocus cues. *CVPR 2019*.

Hackel, T., Savinov, N., Ladicky, L., Wegner, J. D., Schindler, K., and Pollefeys, M. (2017). Semantic3D.net: A new large-scale point cloud classification benchmark. *ISPRS 2017*.

Han, X., Laga, H., and Bennamoun, M. (2019a). Image-based 3D object reconstruction: State-of-the-art and trends in the deep learning era. *TPAMI 2019*.

Han, X., Li, Z., Huang, H., Kalogerakis, E., and Yu, Y. (2017). High-resolution shape completion using deep neural networks for global structure and local geometry inference. In *ICCV 2017*.

Han, X., Zhang, Z., Du, D., Yang, M., Yu, J., Pan, P., Yang, X., Liu, L., Xiong, Z., and Cui, S. (2019b). Deep reinforcement learning of volume-guided progressive view inpainting for 3D point scene completion from a single depth image. In *CVPR 2019*.

Handa, A., Patraucean, V., Badrinarayanan, V., Stent, S., and Cipolla, R. (2016). SceneNet: Understanding real world indoor scenes with synthetic data. In *CVPR 2016*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR 2016*.

Hebert, M., Caillas, C., Krotkov, E., Kweon, I.-S., and Kanade, T. (1989). Terrain mapping for a roving planetary explorer. In *ICRA 1989*.

Homm, F., Kaempchen, N., Ota, J., and Burschka, D. (2010). Efficient occupancy grid computation on the GPU with LiDAR and radar for road boundary detection. In *IV 2010*.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. (1992). Surface reconstruction from unorganized points. In *SIGGRAPH 1992*.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Auton. Robot. 2013*.

Hua, B.-S., Pham, Q.-H., Nguyen, D., Tran, M., Yu, L.-F., and Yeung, S. (2016). SceneNN: A scene meshes dataset with annotations. In *3DV 2016*.

Huang, H., Chen, H., and Li, J. (2019). Deep neural network for 3D point cloud completion with multistage loss function. In *CCDC 2019*.

Janai, J., Güney, F., Behl, A., and Geiger, A. (2020). Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. *Found. Trends Comput. Graph. Vis. 2020*.

Jaritz, M., de Charette, R., Wirbel, É., Perrotton, X., and Nashashibi, F. (2018). Sparse and dense data with CNNs: Depth completion and semantic segmentation. In *3DV 2018*.

Jenke, P., Krückeberg, B., and Straßer, W. (2008). Surface reconstruction from fitted shape primitives. In *VMV 2008*.

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., and Qu, R. (2019). A survey of deep learning-based object detection. *IEEE Access 2019*.

Kaufman, E., Lee, T., Ai, Z., and Moskowitz, I. S. (2016). Bayesian occupancy grid mapping via an exact inverse sensor model. In *ACC 2016*.

Kazhdan, M. and Hoppe, H. (2013). Screened poisson surface reconstruction. *TOG 2013*.

Kazhdan, M. M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *SGP 2006*.

Kelly, A., Stentz, A., Amidi, O., Bode, M., Bradley, D., Diaz-Calderon, A., Happold, M., Herman, H., Mandelbaum, R., Pilarski, T., Rander, P., Thayer, S., Vallidis, N., and Warner, R. (2006). Toward reliable off road autonomous vehicles operating in challenging environments. *IJRR 2006*.

Kim, G. and Kim, A. (2020). Remove, then revert: Static point cloud map construction using multiresolution range images. In *IROS 2020*.

Kiran, B. R., Roldão, L., Irastorza, B., Verastegui, R., Süss, S., Yogamani, S., Talpaert, V., Lepoutre, A., and Trehard, G. (2018). Real-time dynamic object detection for autonomous driving using prior 3D-maps. In *ECCV Workshop 2018*.

Klokov, R. and Lempitsky, V. (2017). Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models. In *ICCV 2017*.

Kolluri, R. (2005). Provably good moving least squares. In *SIGGRAPH 2005*.

Konolige, K. (1997). Improved occupancy grids for map building. *Auton. Robot. 1997*.

Kraetzschmar, G., Gassull, G., and Uhl, K. (2004). Probabilistic quadtrees for variable-resolution mapping of large environments. In *IFAC 2004*.

Kurenkov, A., Ji, J., Garg, A., Mehta, V., Gwak, J., Choy, C. B., and Savarese, S. (2018). DeformNet: Free-form deformation network for 3D shape reconstruction from a single image. In *WACV 2018*.

Kwon, Y., Kim, D., An, I., and Yoon, S. (2019). Super rays and culling region for real-time updates on grid-based occupancy maps. *T-RO 2019*.

Laga, H., Jospin, L. V., Boussaïd, F., and Bennamoun, M. (2020). A survey on deep learning techniques for stereo-based depth estimation. *TPAMI 2020*.

Lalonde, J.-F., Vandapel, N., Huber, D., and Hebert, M. (2006). Natural terrain classification using three-dimensional ladar data for ground robot mobility. *JFR 2006*.

Landrieu, L. and Simonovsky, M. (2018). Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR 2018*.

Lang, A., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. (2019). PointPillars: Fast encoders for object detection from point clouds. In *CVPR 2019*.

Lenssen, J. E., Osendorfer, C., and Masci, J. (2020). Deep iterative surface normal estimation. In *CVPR 2020*.

Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G. A., Fletcher, L., Frazzoli, E., Huang, A. S., Karaman, S., Koch, O., Kuwata, Y., Moore, D., Olson, E., Peters, S., Teo, J., Truax, R., Walter, M. R., Barrett, D., Epstein, A., Maheloni, K., Moyer, K., Jones, T., Buckley, R., Antone, M. E., Galejs, R., Krishnamurthy, S., and Williams, J. (2009). A perception-driven autonomous urban vehicle. *STAR 2009*.

Levin, D. (2004). Mesh-independent surface interpolation. *MATHVISUAL 2004*.

Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J. Z., Langer, D., Pink, O., Pratt, V., Sokolsky, M., Stanek, G., Stavens, D., Teichman, A., Werling, M., and Thrun, S. (2011). Towards fully autonomous driving: Systems and algorithms. *IV 2011*.

Li, D., Shao, T., Wu, H., and Zhou, K. (2017). Shape completion from a single RGB-D image. *TVCG 2017*.

Li, J., Han, K., Wang, P., Liu, Y., and Yuan, X. (2020a). Anisotropic convolutional networks for 3D semantic scene completion. In *CVPR 2020*.

Li, J., Liu, Y., Gong, D., Shi, Q., Yuan, X., Zhao, C., and Reid, I. D. (2019). RGB-D based dimensional decomposition residual network for 3D semantic scene completion. In *CVPR 2019*.

Li, J., Liu, Y. W., Yuan, X., Zhao, C., Siegwart, R., Reid, I., and Cadena, C. (2020b). Depth based semantic scene completion with position importance aware loss. *RA-L 2020*.

Li, L.-J., Socher, R., and Fei-Fei, L. (2009). Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *CVPR 2009*.

Li, S., Zou, C., Li, Y., Zhao, X., and Gao, Y. (2020c). Attention-based multi-modal fusion network for semantic scene completion. In *AAAI 2020*.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. (2018). PointCNN: convolution on X-transformed points. In *NIPS 2018*.

Li, Y., Dai, A., Guibas, L., and Nießner, M. (2015). Database-assisted object retrieval for real-time 3D reconstruction. *CGF 2015*.

Li, Y., Ma, L., Zhong, Z., Liu, F., Chapman, M. A., Cao, D., and Li, J. (2020). Deep learning for LiDAR point clouds in autonomous driving: A review. *TNNLS 2020*.

Li, Y. and Ruichek, Y. (2013). Building variable resolution occupancy grid map from stereoscopic system — a quadtree based approach. In *IV 2013*.

Liao, Y., Donné, S., and Geiger, A. (2018). Deep Marching Cubes: Learning explicit surface representations. In *CVPR 2018*.

Lin, D., Fidler, S., and Urtasun, R. (2013). Holistic scene understanding for 3D object detection with RGB-D cameras. In *ICCV 2013*.

Lin, G., Milan, A., Shen, C., and Reid, I. (2017). RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR 2017*.

Liu, S., Hu, Y., Zeng, Y., Tang, Q., Jin, B., Han, Y., and Li, X. (2018). See and think: Disentangling semantic scene completion. In *NIPS 2018*.

Liu, S., Mello, S. D., Gu, J., Zhong, G., Yang, M.-H., and Kautz, J. (2017). Learning affinity via spatial propagation networks. In *NIPS 2017*.

Liu, W., Sun, J., Li, W., Hu, T., and Wang, P. (2019). Deep learning on point clouds and its application: A survey. *IEEE Sensors 2019*.

Liu, Y. W., Li, J., Yan, Q., Yuan, X., Zhao, C.-X., Reid, I., and Cadena, C. (2020). 3D gated recurrent fusion for semantic scene completion. *ArXiv 2020*.

Lorensen, W. and Cline, H. (1987). Marching cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH 1987*.

Lu, H. and Shi, H. (2020). Deep learning for 3D point cloud understanding: A survey. *arXiv 2020*.

Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. (2018). Shufflenet V2: Practical guidelines for efficient CNN architecture design. In *ECCV 2018*.

Malleson, C., Guillemaut, J.-Y., and Hilton, A. (2019). 3d reconstruction from rgb-d data. *RGB-D Image Anal. Process. Adv. Comput. Vis. Pattern Recog.*

Martens, W., Poffet, Y., Soria, P. R., Fitch, R., and Sukkarieh, S. (2017). Geometric priors for gaussian process implicit surfaces. *RA-L 2017*.

Maturana, D. and Scherer, S. A. (2015). VoxNet: A 3D convolutional neural network for real-time object recognition. In *IROS 2015*.

Meagher, D. (1982). Geometric modeling using octree encoding. *Comput. Graph. Image Process. 1982*.

Meng, H.-Y., Gao, L., Lai, Y., and Manocha, D. (2019). VV-Net: Voxel VAE net with group convolutions for point cloud segmentation. In *ICCV 2019*.

Mescheder, L. M., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. (2019). Occupancy networks: Learning 3D reconstruction in function space. In *CVPR 2019*.

Milioto, A., Vizzo, I., Behley, J., and Stachniss, C. (2019). RangeNet ++: Fast and accurate LiDAR semantic segmentation. In *IROS 2019*.

Milstein, A. (2008). Occupancy grid maps for localization and mapping. *IntechOpen: Motion Planning*.

Mitra, N., Pauly, M., Wand, M., and Ceylan, D. (2013). Symmetry in 3D geometry: Extraction and applications. *CGF 2013*.

Moravec, H. (1988). Sensor fusion in certainty grids for mobile robots. *AI Mag. 1988*.

Moravec, H. and Elfes, A. (1985). High resolution maps from wide angle sonar. *ICRA 1985*.

Nair, R., Lenzen, F., Meister, S., Schäfer, H., Garbe, C., and Kondermann, D. (2012). High accuracy TOF and stereo sensor fusion at interactive rates. In *ECCV Workshop 2012*.

Nealen, A., Igarashi, T., Sorkine-Hornung, O., and Alexa, M. (2006). Laplacian mesh optimization. In *GRAPHITE 2006*.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR 2011*.

Nguyen, A. and Le, H. (2013). 3D point cloud segmentation: A survey. In *RAM 2013*.

Nguyen, T.-N., Michaelis, B., Al-Hamadi, A., Tornow, M., and Meinecke, M. (2012). Stereo-camera-based urban environment perception using occupancy grid and object tracking. In *ITSC 2012*.

Oleynikova, H., Millane, A., Taylor, Z., Galceran, E., Nieto, J., and Siegwart, R. (2016). Signed distance fields: A natural representation for both mapping and planning. In *RSS Workshop 2016*.

Oleynikova, H., Taylor, Z., Fehr, M., Siegwart, R., and Nieto, J. (2017). Voxblox: Incremental 3D euclidean signed distance fields for on-board mav planning. In *IROS 2017*.

Özyesil, O., Voroninski, V., Basri, R., and Singer, A. (2017). A survey of structure from motion. *Acta Numerica 2017*.

Pan, Y., Gao, B., Mei, J., Geng, S., Li, C., and Zhao, H. (2020). SemanticPOSS: A point cloud dataset with large quantity of dynamic instances. In *IV 2020*.

Park, J. J., Florence, P., Straub, J., Newcombe, R. A., and Lovegrove, S. (2019). DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR 2019*.

Pathak, K., Birk, A., Poppinga, J., and Schwertfeger, S. (2007). 3D forward sensor modeling and application to occupancy grid based sensor fusion. In *IROS 2007*.

Pauly, M., Mitra, N., Giesen, J., Groß, M., and Guibas, L. (2005). Example-based 3D scan completion. In *SGP 2005*.

Pauly, M., Mitra, N., Wallner, J., Pottmann, H., and Guibas, L. (2008). Discovering structural regularity in 3D geometry. In *SIGGRAPH 2008*.

Payeur, P., Hebert, P., Laurendeau, D., and Gosselin, C. (1997). Probabilistic octree modeling of a 3D dynamic environment. In *ICRA 1997*.

Payeur, P., Laurendeau, D., and Gosselin, C. (1998). Range data merging for probabilistic octree modeling of 3D workspaces. In *ICRA 1998*.

Pfister, H., Zwicker, M., Baar, J. V., and Gross, M. (2000). Surfels: surface elements as rendering primitives. In *SIGGRAPH 2000*.

Pintore, G., Mura, C., Ganovelli, F., Perez, L. J. F., and an Enrico Gobbetti, R. P. (2020). State-of-the-art in automatic 3D reconstruction of structured indoor environments. *CGF 2020*.

Pock, T. and Chambolle, A. (2011). Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *ICCV 2011*.

Pomerleau, F., Colas, F., and Siegwart, R. (2014). A survey of rigid 3D pointcloud registration algorithms. In *AMBIENT 2014*.

Pomerleau, F., Colas, F., and Siegwart, R. (2015). A review of point cloud registration algorithms for mobile robotics. *Found. Trends Robot. 2015*.

Qi, C. R., Su, H., Mo, K., and Guibas, L. (2017a). PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR 2017*.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS 2017*.

Rabbani, T., Heuvel, F., and Vosselman, G. (2006). Segmentation of point clouds using smoothness constraint. *ISPRS 2006*.

Rakotovao, T. (2017). Integer occupancy grids : a probabilistic multi-sensor fusion framework for embedded perception. *PhD Thesis, Université Grenoble Alpes, 2017*.

Rakotovao, T. A., Mottin, J., Puschini, D., and Laugier, C. (2015). Real-time power-efficient integration of multi-sensor occupancy grid on many-core. In *ARSO Workshop 2015*.

Reed, S., Oord, A., Kalchbrenner, N., Colmenarejo, S. G., Wang, Z., Chen, Y., Belov, D., and Freitas, N. D. (2017). Parallel multiscale autoregressive density estimation. In *ICML 2017*.

Remil, O., Xie, Q., Xie, X., Xu, K., and Wang, J. (2017). Surface reconstruction with data-driven exemplar priors. *CAD 2017*.

Rezende, D. J., Eslami, S., Mohamed, S., Battaglia, P., Jaderberg, M., and Heess, N. (2016). Unsupervised learning of 3D structure from images. In *NIPS 2016*.

Richard, A., Cherabier, I., Oswald, M., Pollefeys, M., and Schindler, K. (2020). KAPLAN: A 3D point descriptor for shape completion. In *3DV*, volume abs/2008.00096.

Riegler, G., Ulusoy, A. O., Bischof, H., and Geiger, A. (2017a). OctNetFusion: Learning depth fusion from data. In *3DV 2017*.

Riegler, G., Ulusoy, A. O., and Geiger, A. (2017b). OctNet: Learning deep 3D representations at high resolutions. In *CVPR 2017*.

Rist, C. B., Emmerichs, D., Enzweiler, M., and Gavrila, D. M. (2020a). Semantic scene completion using local deep implicit functions on LiDAR data. *ArXiv 2020*.

Rist, C. B., Schmidt, D., Enzweiler, M., and Gavrila, D. M. (2020b). SCSSnet: Learning spatially-conditioned scene segmentation on LiDAR point clouds. In *IV 2020*.

Rock, J., Gupta, T., Thorsen, J., Gwak, J., Shin, D., and Hoiem, D. (2015). Completing 3D object shape from one depth image. In *CVPR 2015*.

Roldão, L., de Charette, R., and Verroust-Blondet, A. (2018). A statistical update of grid representations from range sensors. *ArXiv 2018*.

Roldão, L., de Charette, R., and Verroust-Blondet, A. (2019). 3D surface reconstruction from voxel-based LiDAR data. In *ITSC 2019*.

Roldão, L., de Charette, R., and Verroust-Blondet, A. (2020). LMSCNet: Lightweight multiscale 3D semantic completion. In *3DV 2020*.

Roldão, L., de Charette, R., and Verroust-Blondet, A. (2021). 3D semantic scene completion: a survey. *ArXiv 2021*.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI 2015*.

Ros, G., Sellart, L., Materzynska, J., Vázquez, D., and López, A. M. (2016). The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR 2016*.

Roynard, X., Deschaud, J.-E., and Goulette, F. (2018). Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. *IJRR 2018*.

Ryde, J., Dhiman, V., and Platt, R. (2013). Voxel planes: Rapid visualization and meshification of point cloud ensembles. *IROS 2013*.

Saputra, M. R. U., Markham, A., and Trigoni, A. (2018). Visual SLAM and structure from motion in dynamic environments. *CSUR 2018*.

Schaefer, A., Luft, L., and Burgard, W. (2017). An analytical LiDAR sensor model based on ray path information. *RA-L 2017*.

Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient RANSAC for point-cloud shape detection. *CGF 2007*.

Sharma, A., Grau, O., and Fritz, M. (2016). VConv-DAE: Deep volumetric shape learning without object labels. In *ECCV Workshop 2016*.

Shen, C., O'Brien, J., and Shewchuk, J. (2004). Interpolating and approximating implicit surfaces from polygon soup. *TOG 2004*.

Shen, C.-H., Fu, H., Chen, K., and Hu, S. (2012). Structure recovery by part assembly. *TOG 2012*.

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from RGB-D images. In *ECCV 2012*.

Sipiran, I., Gregor, R., and Schreck, T. (2014). Approximate symmetry detection in partial 3D meshes. *CGF 2014*.

Smith, E. and Meger, D. (2017). Improved adversarial systems for 3D object generation and reconstruction. In *CoRL 2017*.

Song, S., Yu, F., Zeng, A., Chang, A. X., Savva, M., and Funkhouser, T. A. (2017). Semantic scene completion from a single depth image. In *CVPR 2017*.

Song, S., Zeng, A., Chang, A. X., Savva, M., Savarese, S., and Funkhouser, T. (2018). Im2Pano3D: Extrapolating 360° structure and semantics beyond the field of view. In *CVPR 2018*.

Sorkine-Hornung, O. and Cohen-Or, D. (2004). Least-squares meshes. In *ICSMA 2004*.

Steinbrücker, F., Sturm, J., and Cremers, D. (2014). Volumetric 3D mapping in real-time on a CPU. In *ICRA 2014*.

Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J., Mur-Artal, R., Ren, C., Verma, S., Clarkson, A., Yan, M., Budge, B., Yan, Y., Pan, X., Yon, J., Zou, Y., Leon, K., Carter, N., Briales, J., Gillingham, T., Mueggler, E., Pesqueira, L., Savva, M., Batra, D., Strasdat, H. M., Nardi, R. D., Goesele, M., Lovegrove, S., and Newcombe, R. A. (2019). The Replica dataset: A digital replica of indoor spaces. *ArXiv 2019*.

Stutz, D. and Geiger, A. (2018). Learning 3D shape completion from laser scan data with weak supervision. In *CVPR 2018*.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3D shape recognition. In *ICCV 2015*.

Sung, M., Kim, V. G., Angst, R., and Guibas, L. (2015). Data-driven structural priors for shape completion. *TOG 2015*.

Sunyoto, H., van der Mark, W., and Gavrila, D. (2004). A comparative study of fast dense stereo vision algorithms. In *IV 2004*.

Tan, W., Qin, N., Ma, L., Li, Y., Du, J., Cai, G., Yang, K., and Li, J. (2020). Toronto-3D: A large-scale mobile LiDAR dataset for semantic segmentation of urban roadways. In *CVPR Workshop 2020*.

Tchapmi, L. P., Choy, C., Iro, Gwak, J., and Savarese, S. (2017). SEGCloud: Semantic segmentation of 3D point clouds. In *3DV 2017*.

Tchapmi, L. P., Kosaraju, V., Rezatofighi, H., Reid, I., and Savarese, S. (2019). TopNet: Structural point cloud decoder. In *CVPR 2019*.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. (2019). KPConv: Flexible and deformable convolution for point clouds. In *ICCV 2019*.

Thrun, S. (2003). Learning occupancy grid maps with forward sensor models. *Auton. Robot. 2003*.

Thrun, S., Burgard, W., and Fox, D. (2005). Probabilistic robotics. *The MIT Press: Probabilistic Robotics*.

Thrun, S. and Wegbreit, B. (2005). Shape from symmetry. In *ICCV 2005*.

Treece, G. M., Prager, R., and Gee, A. (1999). Regularised marching tetrahedra: improved iso-surface extraction. *Comput. & Graph. 1999*.

Triebel, R., Pfaff, P., and Burgard, W. (2006). Multi-level surface maps for outdoor terrain mapping and loop closing. In *IROS 2006*.

Vallet, B., Brédif, M., Serna, A., Marcotegui, B., and Paparoditis, N. (2015). TerraMobilita/iQmulus urban point cloud analysis benchmark. *Comput. & Graph. 2015*.

Varley, J., DeChant, C., Richardson, A., Ruales, J., and Allen, P. (2017). Shape completion enabled robotic grasping. In *IROS 2017*.

Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., and Jiang, Y.-G. (2018a). Pixel2Mesh: Generating 3D mesh models from single RGB images. In *ECCV 2018*.

Wang, P., Liu, Y., Guo, Y., Sun, C., and Tong, X. (2017). O-CNN: octree-based convolutional neural networks for 3D shape analysis. *TOG 2017*.

Wang, P.-S., Liu, Y., and Tong, X. (2020a). Deep Octree-based CNNs with output-guided skip connections for 3D shape and scene completion. In *CVPR Workshop 2020*.

Wang, P.-S., Sun, C., Liu, Y., and Tong, X. (2018b). Adaptive O-CNN: A patch-based deep representation of 3D shapes. *TOG 2018*.

Wang, W., Yu, R., Huang, Q., and Neumann, U. (2018c). SGPN: Similarity group proposal network for 3D point cloud instance segmentation. In *CVPR 2018*.

Wang, X., Ang, M., and Lee, G. H. (2020b). Cascaded refinement network for point cloud completion. In *CVPR 2020*.

Wang, X., Ang, M., and Lee, G. H. (2020c). Point cloud completion by learning shape priors. In *IROS 2020*.

Wang, X., Oswald, M., Cherabier, I., and Pollefeys, M. (2019a). Learning 3D semantic reconstruction on octrees. In *GCPR 2019*.

Wang, Y., Sun, Y., Liu, Z., Sarma, S., Bronstein, M., and Solomon, J. (2019b). Dynamic graph CNN for learning on point clouds. *TOG 2018*.

Wang, Y., Tan, D. J., Navab, N., and Tombari, F. (2018d). Adversarial semantic scene completion from a single depth image. In *3DV 2018*.

Wang, Y., Tan, D. J., Navab, N., and Tombari, F. (2019c). ForkNet: Multi-branch volumetric semantic completion from a single depth image. In *ICCV 2019*.

Wang, Y., Tan, D. J., Navab, N., and Tombari, F. (2020d). SoftPoolNet: Shape descriptor for point cloud completion and classification. In *ECCV 2020*.

Wang, Y., Wu, S., Huang, H., Cohen-Or, D., and Sorkine-Hornung, O. (2019d). Patch-based progressive 3D point set upsampling. In *CVPR 2019*.

Weinmann, M., Jutzi, B., and Mallet, C. (2013). Feature relevance assessment for the semantic interpretation of 3D point cloud data. *ISPRS 2013*.

Weiss, T., Schiele, B., and Dietmayer, K. (2007). Robust driving path detection in urban and highway scenarios using a laser scanner and online occupancy grids. In *IV 2007*.

Wen, X., Li, T., Han, Z., and Liu, Y.-S. (2020). Point cloud completion by skip-attention network with hierarchical folding. In *CVPR 2020*.

Whelan, T., Kaess, M., Fallon, M., Johannsson, H., Leonard, J., and McDonald, J. (2012). Kintinuous: Spatially extended kinectfusion. In *AAAI 2012*.

Wu, S.-C., Tateno, K., Navab, N., and Tombari, F. (2020). SCFusion: Real-time incremental scene reconstruction with semantic completion. In *3DV 2020*.

Wurm, K. M., Hornung, A., Bennewitz, M., Stachniss, C., and Burgard, W. (2010). Octomap : A probabilistic, flexible, and compact 3D map representation for robotic systems. In *ICRA Workshop 2010*.

Xiao, J., Owens, A., and Torralba, A. (2013). SUN3D: A database of big spaces reconstructed using SfM and object labels. In *ICCV 2013*.

Xie, H., Yao, H., Zhou, S., Mao, J., Zhang, S., and Sun, W. (2020a). GRNet: Gridding residual network for dense point cloud completion. In *ECCV 2020*.

Xie, Y., Tian, J., and Zhu, X. X. (2020b). Linking points with labels in 3D: A review of point cloud semantic segmentation. *Geosci. Remote Sens. 2020*.

Yan, X., Gao, J., Li, J., Zhang, R., Li, Z., Huang, R., and Cui, S. (2021). Sparse single sweep LiDAR point cloud segmentation via learning contextual shape priors from scene completion. In *AAAI 2021*.

Yang, B., Rosa, S., Markham, A., Trigoni, N., and Wen, H. (2019). Dense 3D object reconstruction from a single depth view. *TPAMI 2019*.

Yguel, M., Aycard, O., and Laugier, C. (2007). Update policy of dense maps: Efficient algorithms and sparse representation. *STAR 2007*.

Yin, W., Zhang, J.-M., Wang, O., Niklaus, S., Mai, L., Chen, S., and Shen, C. (2020). Learning to recover 3D scene shape from a single image. *ArXiv 2020*.

You, C. C., Lim, S. P., Tan, J. S., Lee, C. K., and Khaw, Y. M. J. (2020). A survey on surface reconstruction techniques for structured and unstructured data. In *ICOS 2020*.

Yu, F. and Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. In *ICLR 2016*.

Yuan, W., Khot, T., Held, D., Mertz, C., and Hebert, M. (2018). PCN: Point completion network. In *3DV 2018*.

Zhang, G. and Chen, Y. (2021). A metric for evaluating 3D reconstruction and mapping performance with no ground truthing. *ArXiv 2021*.

Zhang, J., Zhao, H., Yao, A., Chen, Y., Zhang, L., and Liao, H. (2018a). Efficient semantic scene completion network with spatial group convolution. In *ECCV 2018*.

Zhang, J., Zhao, X., Chen, Z., and Lu, Z. (2019a). A review of deep learning-based semantic segmentation for point cloud. *IEEE Access 2019*.

Zhang, L., Wang, L., Zhang, X., Shen, P., Bennamoun, M., Zhu, G., Shah, S. A. A., and Song, J. (2018b). Semantic scene completion with dense CRF from a single depth image. *Neurocomputing 2018*.

Zhang, P., Liu, W., Lei, Y., Lu, H., and Yang, X. (2019b). Cascaded context pyramid for full-resolution 3D semantic scene completion. In *ICCV 2019*.

Zhang, W., Yan, Q., and Xiao, C. (2020). Detail preserved point cloud completion via separated feature aggregation. In *ECCV 2020*.

Zheng, B., Zhao, Y., Yu, J. C., Ikeuchi, K., and Zhu, S.-C. (2013). Beyond point clouds: Scene understanding by reasoning geometry and physics. In *CVPR 2013*.

Zhong, M. and Zeng, G. (2020). Semantic point completion network for 3D semantic scene completion. In *ECAI 2020*.

Zimmermann, K., Petrícek, T., Salanský, V., and Svoboda, T. (2017). Learning for active 3D mapping. In *ICCV 2017*.

Zollhöfer, M., Stotko, P., Görlitz, A., Theobalt, C., Nießner, M., Klein, R., and Kolb, A. (2018). State of the Art on 3D reconstruction with RGB-D cameras. *CGF 2018*.

Appendix A

Statistical Update of Occupancy Grid Maps

In the following we will detail the density function presented in Section 2.3.2 of Chapter 2 which is used to weight the measurements according to their distance from the sensor as shown in Equation 2.7.

A.1 Density function $\rho(d)$ – Development and validation

The density function $\rho(d)$ has been obtained by several approximations, but still reflects an accurate model of the reality.

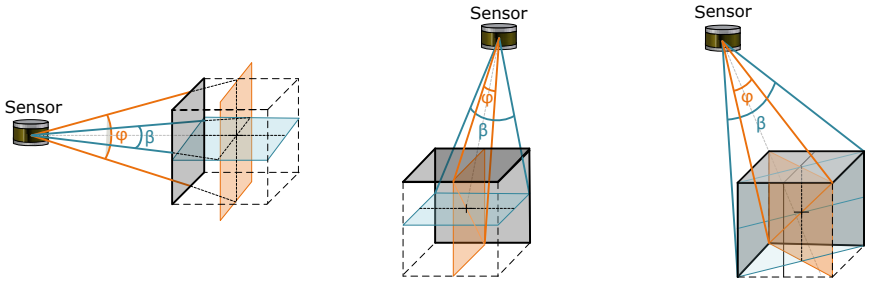
Sensor-voxel measurement density model. Let's start by considering a LiDAR sensor placed at a given point in space. To approximate the density of rays that can traverse a voxel of particular size ω placed at distance d from the sensor, we account for three different cases according to the number of faces of the voxel seen by the sensor (from one to three depending on the voxel position). Referring to Figure A.1, the three cases can be modeled as:

$$\alpha_1(d) = \frac{2}{\varphi_s} \tan^{-1}\left(\frac{\omega}{2d-\omega}\right) \frac{2}{\theta_s} \tan^{-1}\left(\frac{\omega}{2d-\omega}\right). \tag{A.1a}$$

$$\alpha_2(d) = \frac{2}{\varphi_s} \tan^{-1}\left(\frac{\sqrt{2}\omega}{2d-\sqrt{2}\omega}\right) \frac{2}{\theta_s} \tan^{-1}\left(\frac{\omega}{2d-\sqrt{2}\omega}\right). \tag{A.1b}$$

$$\alpha_3(d) = \frac{2}{\varphi_s} \tan^{-1}\left(\frac{\sqrt{3}\omega}{2d-\sqrt{3}\omega}\right) \frac{2}{\theta_s} \tan^{-1}\left(\frac{\sqrt{3}\omega}{2d-\sqrt{3}\omega}\right), \tag{A.1c}$$

where φ_s and θ_s correspond to the vertical and horizontal angular resolutions of the sensor. Note that the three cases considered simplify the calculations but do not lead to an exact model. Cases 3 and 1 represent upper and lower bounds, respectively.



(a) 1 face seen (α_1) (b) 2 faces seen (α_2) (c) 3 faces seen (α_3)

Figure A.1: Cases considered to model the number of rays that traverse a voxel of size ω at distance d. For each case, faces seen by sensor are highlighted in gray.

The modeling function $\rho(d)$ can be then considered as the weighted arithmetic mean of these three values (Equation A.2). The weights are assigned considering the percentage of voxels that fit in any of the three different cases at the surface of a sphere of radius d.

$$\rho(d) = \frac{\eta_1(d)\,\alpha_1(d) + \eta_2(d)\,\alpha_2(d) + \eta_3(d)\,\alpha_3(d)}{\eta_t(d)}, \tag{A.2}$$

where $\eta_t(d)$ approximates the number of voxels of size ω at the surface of the sphere:

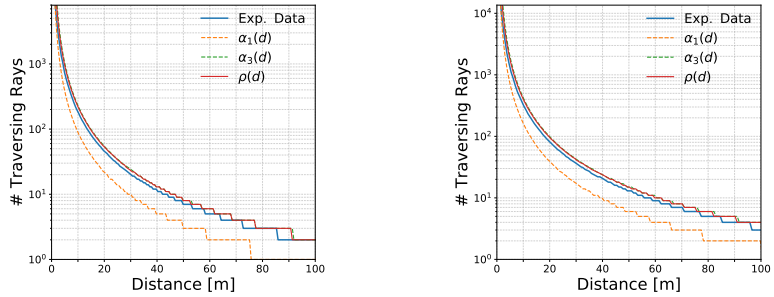
$$\eta_t(d) \approx \frac{4\,\pi\,d^2}{\omega^2}. \tag{A.3}$$

If we suppose the sensor placed at the center of the sphere, the number of voxels where two faces can be seen $\eta_2(d)$, are the ones placed within the perpendicular planes to the axes (Oxy, Oxz and Oyz), except for the voxels aligned with the axes where only one face can be seen $\eta_1(d)$. The rest of voxels are represented by $\eta_3(d)$.

$$\eta_1(d) = 6. \tag{A.4a}$$

$$\eta_2(d) \approx 3\left(\frac{2\,\pi\,d}{w}\right) - 12. \tag{A.4b}$$

$$\eta_3(d) \approx \frac{4\,\pi\,d^2}{\omega^2} - \eta_1(d) - \eta_2(d). \tag{A.4c}$$



(a) $w = 0.6m$ (b) $w = 0.8m$

Figure A.2: Our model was validated with experimental data by inserting a spherical synthetic pointcloud of radius r in a voxel grid of cell size ω. Figures show comparison between experimental (depicted in blue) and the estimated density function $\rho(d)$ (shown in red) at two different voxel sizes.

Notice that for larger distances, most of the voxels will belong to η_3. Moreover, if $d \gg w$, then η_3 can be approximated to η_t and our modeling function can be simplified as $\rho(d) \approx \alpha_3(d)$.

Model validation. To validate our model, let us compare its results with the ones obtained by inserting a synthetic spherical pointcloud in a voxel grid at different resolutions and counting the number of rays traversing each cell in a $100m$ distance range, this can be seen on Figure A.2. Notice that the green and orange curves represent the upper and lower bounds $\alpha_3(d)$ and $\alpha_1(d)$, respectively. The blue line represents the results obtained with the experiment, which are close to the upper bound since as it was stated previously, if $d \gg w$ most of the voxels are represented by $\alpha_3(d)$. The density function $\rho(d)$, as presented in Equation A.2 can also be seen in red. Thereof, the modeling function is used to obtain our weighting function $w(d)$.

RÉSUMÉ

Dans cette thèse, nous nous intéressons à des problèmes liés à la reconstruction et la complétion des scènes 3D à partir de nuages de points de densité hétérogène. Dans ce cadre, nous proposons différentes méthodes utiles lors de la création d'un modèle 3D de l'environnement.

Dans une première partie, nous étudions l'utilisation de grilles d'occupation tridimensionnelles pour la reconstruction d'une scène 3D à partir de plusieurs observations, ce qui est utile pour la localisation d'un robot mobile ou d'un véhicule autonome et les applications de cartographie. Nous proposons d'exploiter les informations de trajet des rayons pour résoudre des ambiguïtés dans les cellules partiellement occupées. Notre approche permet de réduire les imprécisions dues à la discrétisation et d'effectuer des mises à jour d'occupation des cellules dans des scénarios dynamiques.

Puis, dans le cas où le nuage de points correspond à une seule observation de la scène, nous introduisons un algorithme de reconstruction de surface implicite 3D capable de traiter des données de densité hétérogène en utilisant une stratégie de voisinages adaptatifs. Notre méthode permet de compléter de petites zones manquantes de la scène et génère une représentation continue de la scène adaptée à la modélisation physique ou à l'évaluation de la traversabilité du terrain.

Enfin, nous nous intéressons aux approches d'apprentissage profond adaptées à la complétion sémantique d'une scène 3D, c'est-à-dire qui permettent de compléter les données d'entrée 3D et d'effectuer une segmentation sémantique de ces données. Après avoir présenté une étude approfondie des méthodes existantes, nous introduisons une nouvelle méthode de complétion sémantique multi-échelle appropriée aux scénarios en extérieur. Pour ce faire, nous proposons une architecture constituée d'un réseau neuronal convolutif hybride basé sur une branche principale 2D pour réduire les coûts de calcul et comportant des têtes de segmentation 3D pour prédire la scène sémantique complète à différentes échelles. Notre approche est nettement plus légère et plus rapide que les approches existantes, tout en ayant une efficacité similaire.

MOTS CLÉS

Reconstruction 3D de la scène, complétion sémantique 3D de la scène, segmentation sémantique 3D, reconstruction de surface, nuages de points, grilles de voxels, conduite autonome.

ABSTRACT

In this thesis, we address the challenges of 3D scene reconstruction and completion from sparse and heterogeneous density point clouds. Therefore proposing different techniques to create a 3D model of the surroundings.

In the first part, we study the use of 3-dimensional occupancy grids for multi-frame reconstruction, useful for localization and HD-Maps applications. This is done by exploiting ray-path information to resolve ambiguities in partially occupied cells. Our sensor model reduces discretization inaccuracies and enables occupancy updates in dynamic scenarios.

We also focus on single-frame environment perception by the introduction of a 3D implicit surface reconstruction algorithm capable to deal with heterogeneous density data by employing an adaptive neighborhood strategy. Our method completes small regions of missing data and outputs a continuous representation useful for physical modeling or terrain traversability assessment.

We dive into deep learning applications for the novel task of semantic scene completion, which completes and semantically annotates entire 3D input scans. Given the little consensus found in the literature, we present an in-depth survey of existing methods and introduce our lightweight multiscale semantic completion network for outdoor scenarios. Our method employs a new hybrid pipeline based on a 2D CNN backbone branch to reduce computation overhead and 3D segmentation heads to predict the complete semantic scene at different scales, being significantly lighter and faster than existing approaches.

KEYWORDS

3D scene reconstruction, 3D semantic scene completion, 3D semantic segmentation, surface reconstruction, point clouds, voxel grids, autonomous driving.