Semantic Scene Completion using Local Deep Implicit Functions on LiDAR Data

Rist, Christoph; Emmerichs, David; Enzweiler, Markus; Gavrila, Dariu

DOI
10.1109/TPAMI.2021.3095302

Publication date
2022

Document Version
Final published version

Published in
IEEE Transactions on Pattern Analysis and Machine Intelligence

Semantic Scene Completion Using Local Deep Implicit Functions on LiDAR Data

Christoph B. Rist, David Emmerichs, Markus Enzweiler, and Dariu M. Gavrila

Abstract—Semantic scene completion is the task of jointly estimating 3D geometry and semantics of objects and surfaces within a given extent. This is a particularly challenging task on real-world data that is sparse and occluded. We propose a scene segmentation network based on local Deep Implicit Functions as a novel learning-based method for scene completion. Unlike previous work on scene completion, our method produces a continuous scene representation that is not based on voxelization. We encode raw point clouds into a latent space locally and at multiple spatial resolutions. A global scene completion function is subsequently assembled from the localized function patches. We show that this continuous representation is suitable to encode geometric and semantic properties of extensive outdoor scenes without the need for spatial discretization (thus avoiding the trade-off between level of scene detail and the scene extent that can be covered). We train and evaluate our method on semantically annotated LiDAR scans from the Semantic KITTI dataset. Our experiments verify that our method generates a powerful representation that can be decoded into a dense 3D description of a given scene. The performance of our method surpasses the state of the art on the Semantic KITTI Scene Completion Benchmark in terms of geometric completion intersection-over-union (IoU).

Index Terms—LiDAR, semantic scene completion, semantic segmentation, geometry representation, deep implicit functions

1 Introduction

Autonomous mobile robots have to base their actions almost exclusively on an internal representation of their current environment. Perception systems are built to create and update such a representation from real-time raw sensor data. We are interested in a model of the current environment that preferably condenses the information that is important for the task at hand or makes it easy to extract relevant information. For robot navigation it is required to estimate whether a certain area is occupied by an object and what semantic meaning different objects and surfaces hold. Even non-mobile settings, e.g., mapping applications, benefit from an effective geometric and semantic completion of low-resolution or incomplete sensor data. To fulfill this need 3D completion aims to map and infer the true geometry of objects from sensor input. Semantic scene completion extends this task to larger arrangements of multiple objects and requires to predict the corresponding semantic classes.

Sensor data can only reflect partial observations of the real world. First, this is because of the physical properties of the sensors themselves which impose limits on their ultimate resolution, frequency, and minimal amount of noise with which they capture data. Second, it is because every sensor is restricted to its current perspective. Thus, after the point-of-view sensor data is mapped into the 3D scene, the result will always be characterized by a distance-decreasing sampling density, occlusions and blind spots (see regions marked A, B, C in Fig. 1 respectively). Multiple sensors mounted on a single vehicle do not alleviate that issue significantly. They are usually positioned rather close together, so that their view of the surroundings still exhibits almost the same degree of occlusions and shadows. Hence, the completion task in 3D euclidean space represents a key challenge for perception in real-time cognitive robotics: Making predictions about currently unobserved areas by the use of context and experience. This ability is only necessary for real-time perception systems. In a static world without time constraints it would be possible to just move the sensors towards areas of interest to gain evidence of their true appearance. But unlike static worlds, mobile robots need to reason about the nature of objects given only the current observations.

The semantic scene completion task is based on a correlation between the semantic class of an object or surface and its physical 3D geometry. In the case of LiDAR, the sensor observes a part of the scene's geometry. The semantics that can be deduced from this geometry can be used to then again complete the missing geometry. Regardless of the dataset in use, hidden geometry can only be completed by means of what is probable but never with absolute certainty. This probability is in turn associated with the type of objects within the scene. Naturally, human perception exhibits the same inherent limitations as computer sensors when it comes to physical limitations and the laws of 3D geometry. However, humans make up for this by fitting a powerful

- *Christoph B. Rist is with Intelligent Vehicles Group, TU Delft, 2628, CD, Delft, The Netherlands, and also with the Mercedes-Benz AG, 70597 Stuttgart, Germany. E-mail: christoph_bernd.rist@daimler.com.*
- *David Emmerichs is with Mercedes-Benz AG, 70597 Stuttgart, Germany. E-mail: david_josef.schmidt@daimler.com.*
- *Markus Enzweiler is with Esslingen University of Applied Sciences, 73728 Esslingen am Neckar, Germany. E-mail: markus.enzweiler@hs-esslingen.de.*
- *Dariu M. Gavrila is with Intelligent Vehicles Group, TU Delft, 2628, CD, Delft, The Netherlands. E-mail: d.m.gavrila@tudelft.nl.*

Manuscript received 15 Nov. 2020; revised 22 June 2021; accepted 28 June 2021. Date of publication 7 July 2021; date of current version 9 Sept. 2022.
(Corresponding author: Dariu M. Gavrila.)
Recommended for acceptance by R. Timofte.
Digital Object Identifier no. 10.1109/TPAMI.2021.3095302

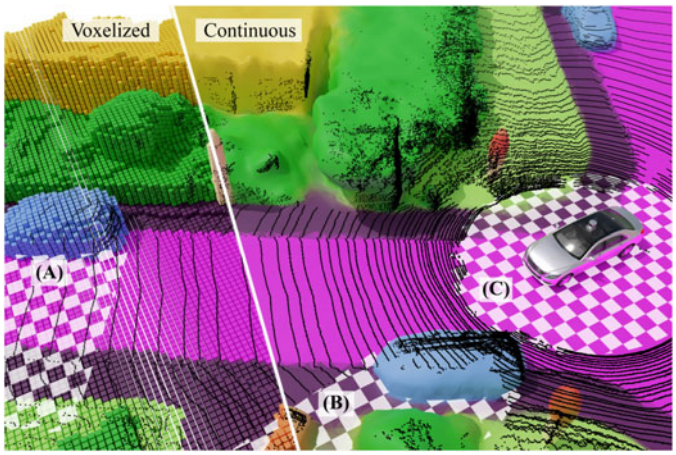


Fig. 1. Illustration of the semantic scene completion task and the output of our method. Sensors are limited in their resolution and restricted to a single perspective of their surroundings. A LiDAR scan (herein depicted as black points) is characterized by a varying degree of sparsity caused either by distance (A), occlusions from objects (B) or sensor blind spots (C). Our method is able to complete the sparse scan geometrically and semantically and can be applied to large spatial extents as typically found in outdoor environments. The underlying representation is not tied to a fixed output resolution and describes the scene using a continuous function (right side, color indicates semantic class). Therefore the geometry does not exhibit quantization artifacts resulting from a discretization into voxels (left side).

model to infer even large missing pieces of geometric and semantic information about their surroundings.

Our approach is a deep learning method that we train on a large number of semantically annotated LiDAR measurements. The model leverages the training data as prior knowledge to reason about the geometry and semantics of the complete 3D scene from a single LiDAR scan as input. We propose to represent the scene completion output with localized Deep Implicit Functions (DIFs). A DIF is a continuous function over 3D space which classifies individual positions. The composed scene completion function $\mathbf{f}^{\mathbf{c}}_{\text{LDIF}} : \mathbb{R}^3 \rightarrow [0,1]^{N+1}$ is defined over all scene positions and outputs a classification vector over N semantic classes and free space. This continuous representation avoids a trade-off between achievable spatial output resolution and the extent of the 3D scene that can be processed. Fig. 1 presents a visualization of the resulting function and a comparison to a voxelized output.

When it comes to the representation of geometry, existing works on object or scene completion focus most commonly on voxelization [1], [2], [3], [4], [5], [6], [7], [8], [9]. However, this results in satisfactory output resolutions only for volumes of limited extent. Approaches using DIFs to represent shapes [10], [11], [12], [13] only encode single objects into a fixed size latent vector. Most previous work completes 3D geometry on the assumption that the scene in question is covered evenly with sensor measurements, such as indoor scenes recorded with RGB-D cameras. In comparison, the density of a LiDAR scan decreases steadily with distance so that gaps between measurements get larger. Distance to the sensor and occlusions lead to areas where the actual ground truth geometry cannot be inferred anymore from the measurements. This label noise and the varying sparsity is a challenge for current models [6].

Our method requires accurate 3D measurements of a scene to be trained for geometric completion. These measurements can be obtained from one or multiple LiDAR sensors, or a LiDAR sensor that is moved through the scene, provided that all measurements can be transformed into a single reference coordinate system. If semantic annotations are not available our method can still be trained for pure completion of scene geometry.

This paper builds upon our earlier work on LiDAR-based scene segmentation [14]. For this work, we created a training procedure for semantic scene completion based on accumulated LiDAR data and conducted an extensive experimental evaluation of our design choices and parameters. In summary, our contributions are:

- We produce a representation for both geometry and semantics of 3D scenes by Deep Implicit Functions with spatial support derived from a 2D multi-resolution grid. Our combination with continuous output coordinates make dense decoding of large spatial extents feasible.
- We generate point-like training targets from time-accumulated real-world LiDAR data and the included free space information. Dynamic objects are considered separately to ensure consistency.
- In experiments on the Semantic KITTI Scene completion benchmark, we show that the proposed approaches outperform voxel-based methods on geometric completion accuracy.

2 Related Work

First, this section discusses ways to represent geometry and surfaces within the context of reconstruction algorithms. Second, related work about geometric completion is categorized into completion of single object shapes and completion of indoor scenes from synthetic or RGB-D data. Finally, we take a look at the state of the art in semantic segmentation and scene completion of outdoor scenes from real-world LiDAR data.

2.1 Geometry and Surface Representation

Most commonly the output representation for 3D scene completion is a voxel occupancy grid [4], voxelized (truncated) signed distance functions (SDFs) [1], [2], [3], [5], [15], or interpolation and CRFs [16] for sub-voxel accuracy. A differentiable deep marching cubes algorithm replaces the SDF as an intermediate representation and enables to train the surface representation end-to-end [17] but the resulting representation is still constrained to the underlying voxel resolution. The general trade-off between output resolution and computational resources is an issue for 3D representations [1]. Octree-based convolutional neural networks (CNNs) have been proposed to represent space at different resolutions and to perform gradual shape refinements [18], [19], [20], [21].

Recent works represent 3D shapes and surfaces implicitly as isosurfaces of an output function which classifies single points in euclidean 3D space [10], [11], [12], [13]. Depending on the output function's complexity this approach has the capacity and expressiveness to represent fine geometric details. An encoder creates a parameter vector that makes the output function dependent on the actual

input data for geometric reconstruction. Both the output function and encoder are represented as deep neural networks (DNNs) and trained by backpropagation. They either use oriented surfaces [12] or watertight meshes [11] from ShapeNet [22] as synthetic full-supervision training targets. These methods have improved the state of the art significantly for shape reconstruction and completion. However, their scope is limited to the reconstruction of single objects. These approaches do not generalize or scale well because of the nature of a single fixed-size feature vector that represents a shape globally.

Recently, DIFs are combined with grid structures or other support positions that improve their spatial capabilities to describe larger scene extents [23], [24] or more complex geometric details of individual objects [24], [25], [26] instead of only simple shapes.

To represent more complex details in 3D shapes, a set of local analytic 3D functions with limited support can be combined with deep implicit functions to predict occupancy [26]. The latent representations of individual small synthetic object parts can be used to assemble a large 3D scene [23]. For this purpose, synthetic objects are first auto-encoded to generate the latent space. Then, a possible representation of a scene is found by iterative inference. This setup only requires a decoder from latent grid to the 3D scene. Concurrent to our work, [24], [25] encode 3D points into a 2D grid or 3D feature volume and perform bilinear or trilinear interpolation on this feature space. Here [25] explicitly considers features from multiple resolutions and the query position in only used for interpolation, not in the decoder. [24] uses the query position for interpolation and again as concatenation to the latent feature in the decoder. The feature grid is single-resolution. For geometric reconstruction of indoor RGB-D data, the full volumetric grid performs best. With a focus on representation and reconstruction of geometry, the method is trained on synthetic watertight-meshes and uniformly sampled point clouds are used as input.

2.2 Shape Completion

Poisson surface reconstruction is a state-of-the-art reconstruction algorithm for an object's surface from measured oriented points [27]. As with other implicit representations the resulting geometry needs to be extracted by marching cubes or an iterative octree variant of marching cubes [11]. Poisson surface reconstruction handles noise and imperfect data well and adapts to different local sampling densities. However, it is of limited use on real-time real world data as it is unable to leverage prior knowledge to complete unseen or sparse regions unlike methods based on learned shape representations.

Many data-driven, learning-based and symmetry-based approaches have been proposed for shape completion. We refer to Stutz *et al.* [28] for an overview and focus on shape completion on LiDAR scans. 3D models can be used to train a DNN for the shape completion problem on synthetic data and perform inference on real LiDAR scans [29]. Alternatively, a shape prior from synthetic data can be used for amortized maximum likelihood inference to avoid the domain gap between synthetic and real data [28]. Recently, it has been shown that synthetic data can be avoided altogether by using a multi-view consistency constraint to train shape completion only from LiDAR scans without full supervision [30].

2.3 Semantic Scene Completion

For a recent comprehensive survey on semantic scene completion we refer to [31]. The subject of scene completion has first gotten momentum from the wide availability of RGB-D cameras leading to the advent of indoor semantic segmentation datasets such as the NYUv2 Depth Dataset [32] and ScanNet [33]. [2] is a pioneering work to infer full scene geometry from a single depth image in an output space of voxelized SDFs. Generalization to entirely new shapes is data-driven and implemented with voxel occupancy predicted by a structured random forest. A specially created table-top scene dataset with ground truth from a Kinect RGB-D camera is used as full-supervision training target.

A volumetric occupancy grid with semantic information can be predicted from voxelized SDFs as input in an end-to-end manner [1], [4]. They apply their methods to synthetic indoor data from the SUNCG dataset. While [4] is appropriate only on single RGB-D images, [1] extends to larger spatial extents. Multiple measures improve geometric precision and consistency: Using SDFs as output representation per voxel, an iterative increase of voxel resolution, and the division of space into interleaving voxel groups. Voxelized SDFs and semantic segmentation can be inferred by explicit fusion of single depth images with RGB data [3]. [23] validates the geometric representation power of DIFs in combination with a structured latent space approach on indoor RGB-D data of the Matterport3D dataset [34]. The details in the completion of RGB-D scans from Matterport3D can be improved by progressive spatial upsampling in the decoder and a deliberate loss formulation that does not penalize unseen areas [15].

2.4 Segmentation and Scene Completion on LiDAR Data

Numerous prior works focus on semantic segmentation of all observed data points resulting in a pixel-wise or point-wise classification of LiDAR data. These methods do not predict any labels for invisible parts of space from the sensor's perspective. However, datasets and benchmarks on real-world road scenes have defined a standard of semantic classes that is significant while simultaneously advancing the state of the art [6], [35], [36]. CNN-architectures on RGB-Images for segmentation and detection [37], [38] have inspired sensor-view based approaches in the more recent LiDAR-based segmentation task [39], [40], [41]. Neural network architectures adjust to the three dimensional nature of a segmentation or detection problem through voxelization of input data [42], [43], [44], [45], combination with sensor-view range images [46], and use of surface geometry [39]. Computation, memory efficiency and representation of details of voxel architectures can be improved by combining a coarser voxel structure with a point-feature branch for details [45] and neural architecture search [47].

The scene completion problem on real-world data has only recently been advanced by the large-scale Semantic

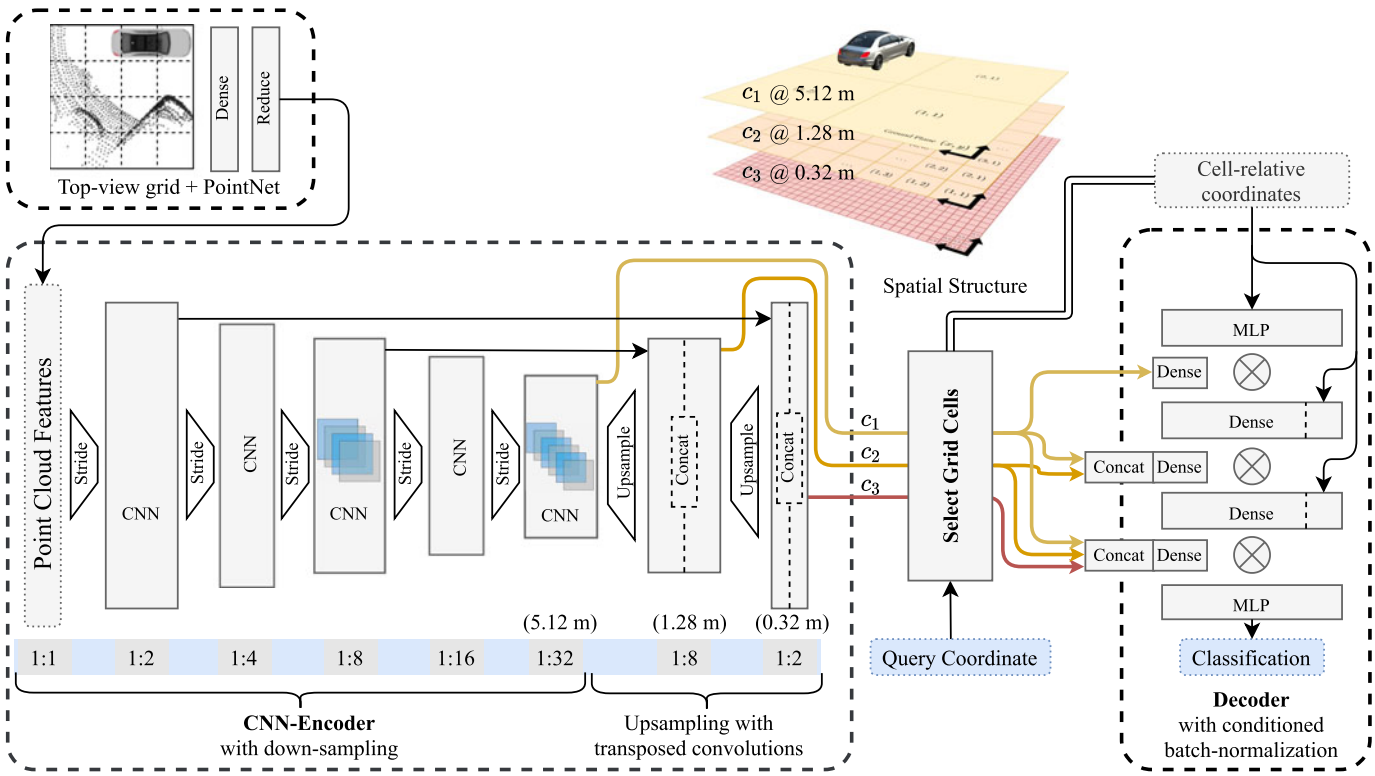


Fig. 2. *Network architecture:* The feature extractor creates a top-view feature map of the input point cloud. The CNN-encoder outputs feature maps at three different resolutions that make up the latent representation of the 3D scene. The decoder classifies individual coordinates within the 3D scene extent. Latent feature vectors and relative-coordinates are processed by conditioned batch normalization in the decoder.

KITTI dataset [6] featuring point-wise semantic annotations on LiDAR together with a private test set and a segmentation benchmark for semantic scene completion. Methods originally applied to scene completion from depth images [3], [4] can be adapted for LiDAR scene completion: The Semantic KITTI authors [6] adapt the Two-Stream (TS3D) approach [3] which is originally applied to depth images of indoor scenes of the NYUv2 dataset. TS3D combines geometric information from a depth image and a predicted semantic segmentation from an RGB image in a volumetric voxel grid. For Semantic KITTI outdoor scenes, they use a state-of-the-art DeepNet53 segmentation network trained on Cityscapes [36] and SatNet [48] for voxel output.

The three recent methods LMSCNet [7], JS3CNet [8], and S3CNet [9] only use LiDAR data as input. The usage of U-net architectures for down-, upsampling, and spatial context is a common architectural pattern. LMSCNet [7] operates on the voxelized LiDAR input and uses a 2D-CNN backbone for feature extraction. The voxelized output is inferred with a monolithic hybrid-network that predicts the completion end-to-end. LMSCNet can output a lower-resolution coarse version of a scene at an intermediate stage. However, their experiments show that the single-output version trained only on the highest resolution performs slightly better than the multi-scale version trained with multiple-resolution losses.

S3CNet [9] and JS3CNet [8] both use the raw LiDAR scan as input. Both also propose to use a lower resolution scene representation internally which is subsequently upsampled into the full output voxel resolution. JS3CNet proposes a two-stage approach: First, a semantic segmentation of the input LiDAR scan is inferred. Second, a neural network fuses the voxelized semantic segmentation and point-wise feature vectors into the voxelized representation of the completed scene. S3CNet augments the input LiDAR scan with a calculation of normal surface vectors from the depth-completed range image and TSDF values. These are stored in a sparse tensor. A semantic 2D BEV map and a 3D semantic sparse tensor are predicted in parallel. These are then subsequently fused into a full 3D tensor. The final scene completion is obtained after a second semantically-based post-processing. The authors conduct ablations and attribute a large share of the final results to the post-processing.

3 Proposed Approach

3.1 Overview

Our method takes as input a LiDAR scan and outputs the corresponding scene completion function $\mathbf{f}^{\mathbf{c}}_{\text{LDIF}} : \mathbb{R}^3 \rightarrow [0,1]^{N+1}$. This function maps every 3D position $\mathbf{p}$ within the scene to a probability vector that we define to represent the semantic class of the position $\mathbf{p}$. The dependence of the completion function $\mathbf{f}^{\mathbf{c}}_{\text{LDIF}}$ on the input data is expressed by the superscript vector $\mathbf{c}$. Positions belonging to objects in the scene are categorized into N semantic classes. The additional class *free space* represents positions that are not occupied by any object (instead they are occupied by air). The resulting total of $N+1$ classes is able to describe every position within the scene. Hence the $\mathbf{f}^{\mathbf{c}}_{\text{LDIF}}$ function uniformly represents the geometric and semantic segmentation of space instead of only the physical boundaries of objects. The global $\mathbf{f}^{\mathbf{c}}_{\text{LDIF}}$ function is built from many local functions $\mathbf{f}_{\text{L}}$. Every local function has two distinctive inputs: The coordinate of interest $\Delta\mathbf{p}$ and a parameterization vector $\mathbf{c}_V$. In

the context of DIFs, producing an output function $\mathbf{f}_{\mathrm{L}}^{\mathbf{c}}$ means generating a parameterization (*conditioning*) vector $\mathbf{c}_V$. When the parameterization vector $\mathbf{c}_V$ is fixed we obtain the conditioned function $\mathbf{f}_{\mathrm{L}}^{\mathbf{c}}$ which is only dependent on the remaining input coordinate $\Delta\mathbf{p}$. Our approach to the composition of the $\mathbf{f}_{\mathrm{LDIF}}^{\mathbf{c}}$ function is designed to encode large outdoor scenes. While related works on single object shape representation encode geometry information in a fixed size conditioning vector, we add spatial structure to the latent space through the use of a 2D feature grid. Each grid entry is a conditioning vector for a local function. The grid is chosen to be two-dimensional, uniform and represents the xy-coordinates of a flattened scene that omits the vertical dimension. We use three grids, each with its own feature resolution. An illustration is given in Fig. 2. As a consequence of the grid approach, the amount of conditioning information is tied to the spatial extent of the scene. The intuition is that each individual conditioning vector now describes only a small part of the complete scene in the vicinity of its own position. Each grid entry always encodes a volume of the same size, regardless of the overall scene extent.

We propose a convolutional encoder to generate the feature maps that make up the conditioning grid. Outdoor scenes are mainly composed from objects at different locations on the ground plane (xy). Therefore the configuration of outdoor scenes is assumed to be translation-invariant in x and y direction. Intuitively, the encoding of the front of a car or a part of a tree can be the same regardless of the absolute position of the object within the scene. For this reason we consider the implementation of the encoder as a convolutional neural network as appropriate. Fig. 2 gives a schematic overview over the point cloud encoding stage, feature selection, and decoding a position into a coordinate classification.

The next section describes the details of the composition of the global completion function $\mathbf{f}_{\mathrm{LDIF}}^{\mathbf{c}}$ from multiple conditioning vectors and grid resolutions. A sampling-based supervised training method from real-world LiDAR data is proposed and details on the used network architecture and inference procedure follow.

3.2 Spatial Structure of Latent Feature Grid

Composition of $\mathbf{f}_{\mathrm{LDIF}}^{\mathbf{c}}$. Centerpiece of our method is the formulation of latent conditioning vectors that are spatially arranged in a grid and generated by a convolutional encoder network on LiDAR point clouds. Each individual conditioning vector $\mathbf{c}_V$ parameterizes a *local segmentation function* $\mathbf{f}_{\mathrm{L}}^{\mathbf{c}}(\Delta\mathbf{p}_V)$ to classify a position of interest $\mathbf{p}$. Even though the domain of individual local functions is $\mathbb{R}^3$ and therefore infinite, the classification will only be meaningful for positions that are close to the conditioning vector's position within the scene.

It is necessary to define how a conditioning vector is selected for a given query coordinate $\mathbf{p}$. It is straightforward to use the single vector of the grid cell that contains the coordinate $\mathbf{p}$ when projected onto the ground plane. But with this approach the resulting global function would exhibit discontinuities between grid cells. Instead, we select the four grid cells with the closest center coordinates for the query coordinate $\mathbf{p}$. Thus we obtain four individual classifications for $\mathbf{p}$ and perform bilinear interpolation according to $\mathbf{p}$'s position within the square of the surrounding grid cell center points. We denote the set of the four closest conditioning cells the *support region* $\mathcal{V}_{\mathbf{p}}$ of the coordinate $\mathbf{p}$ and the corresponding coefficients for bilinear interpolation w. This yields the global classification function

$$\mathbf{f}_{\mathrm{LDIF}}^{\mathbf{c}}(\mathbf{p}) = \sum_{V \in \mathcal{V}_{\mathbf{p}}} w(\Delta\mathbf{p}_V)\mathbf{f}_{\mathrm{L}}(\mathbf{c}_V, \Delta\mathbf{p}_V) \tag{1}$$

$$\text{with} \quad \Delta\mathbf{p}_V = \mathbf{p} - \mathbf{o}_V, \tag{2}$$

for a coordinate $\mathbf{p}$. $\mathbf{o}_V$ is the center position of a cell V and $\mathbf{c}_V$ is the conditioning vector at cell V. The coefficients for bilinear weighing $w(\Delta\mathbf{p}_V)$ sum to 1. Intuitively, the spatial extent of a scene can be thought of as covered by overlapping function patches $\mathbf{f}_{\mathrm{L}}$. Each function $\mathbf{f}_{\mathrm{L}}$ has its own coordinate origin $\mathbf{o}_V$ at the center of its grid cell V. Eq. (2) conveys the translation of scene coordinates $\mathbf{p}$ into the coordinate system of the conditioning vector's grid cell that shall describe $\mathbf{p}$.

Multi-Resolution Scene Representation. An important aspect of the composition of $\mathbf{f}_{\mathrm{LDIF}}^{\mathbf{c}}$ is the use of three individual conditioning vectors from three different resolutions levels. The intuition behind this is that the geometric structure of a scene is composed of different levels of detail. There is the coarse positioning of the ground level and large structures as well as more fine-grained details like curbstones, small objects and poles. We reproduce this range in the network structure to facilitate learning of a smooth representation with more details and more consistency over cell boundaries. The conditioning information for a single local function f_L is composed from three resolution-specific feature vectors. We opt for features $\mathbf{c}_V = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ from the resolution ratios 1:16, 1:4, and 1:1 that originate from a U-net-structured [49] convolutional feature encoder, as illustrated in Fig. 2. The resolution ratios correspond to grid cells with 5.12 m, 1.28 m, and 0.32 m edge length respectively.

For a scene position $\mathbf{p}$, we select the four closest feature vectors within the highest resolution feature map as support region. This 2×2 square of feature vectors is used for bi-linear interpolation. For each of the lower resolutions, the single closest feature vector and associated cell is selected to complete the conditioning of the local function. The resulting four local segmentation functions all describe the single position $\mathbf{p}$ in the scene. All four need to be evaluated to obtain the final interpolated classification result.

Each conditioning vector $\mathbf{c}_i, i \in \{1, 2, 3\}$ belongs to a grid cell V_i at resolution i defining a coordinate system relative to its own position through its origin $\mathbf{o}_{V_i}$. Due to the hierarchical set of vectors $(\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ at different resolutions, we also obtain a corresponding 3-tuple of relative coordinates $\Delta\mathbf{p}_V = (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ with $\mathbf{p}_i = \mathbf{p} - \mathbf{o}_{V_i}$ as input for $\mathbf{f}_{\mathrm{L}}$.

3.3 Training on LiDAR Point Clouds

Sampling Targets for Supervised Training. The decoder neural network and feature encoder are trained end-to-end using individual coordinates within the scene and their associated training labels. This set of coordinate-label tuples is generated from different data sources. The large number of time-

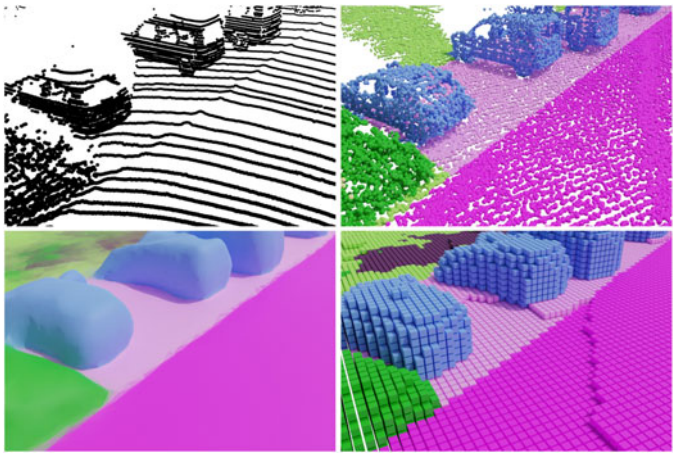


Fig. 3. Left to right, top to bottom: Input points, ground truth accumulated points, mesh visualization of continuous output function, derived voxelization at 20cm edge length. Geometric details can be represented more accurately by our continuous output function as compared to the voxelization resolution of the Semantic KITTI dataset. Our method does not cause artifacts on slanted surfaces (e.g., road plane) or edges between objects.

accumulated LiDAR measurements is used as primary training target. Each LiDAR point has a position in the reference coordinate frame and an associated semantic label. Together, these positions make up all training targets for the occupied classes. The top row of Fig. 3 shows the single input point cloud and the accumulated training targets with semantic annotations.

Next, we need to obtain positions that are of the free space class, so not occupied by any object. The pre-processing that accumulates LiDAR points keeps track of all voxels that are observed at least once, but empty. In every such empty voxel we sample a free space position target uniformly at random. This ensures that the scene extent is evenly covered with free space information.

We use the input point cloud as a second source of free space positions. The straight line between a LiDAR measurement and the sensor's position at time of measurement is empty, meaning not occupied by any object. We exploit this reality for self-supervised training of object geometry. The goal of our scene completion function is to resemble physical boundaries. Wherever surfaces are scanned by the LiDAR sensor we would like to have a sharp transition of the completion function from the prediction of an occupied class to a free space prediction. Therefore we sample free space positions on the straight lines between LiDAR measurement and sensor position. We use an exponential decaying probability distribution to sample the free space positions close to the surfaces of objects. The approach of close surface sampling of free space targets and the combination of surface sampled and global training positions is similar to [26].

Loss Function. Training the classifier involves three separate loss terms: semantic L_S, geometric L_G, and consistency L_C loss. Semantics and geometry could also be covered by a single cross-entropy classification problem. However, the formulation with individual losses allows to include positions that are known to be occupied by an object without information about an object class, e.g., unlabeled LiDAR points. Moreover, geometric and semantic loss terms can be weighted more easily against each other. The overall loss

$$L = \lambda_S \sum_{\mathcal{P}} L_S + \lambda_G \sum_{\mathcal{P}} L_G + \lambda_C \sum_{\mathcal{P}} L_C, \tag{3}$$

is the weighted sum of the individual losses that are each in turn summed over all training targets. We write the predicted probability vector at position $\mathbf{p}$ as $[f_1, \ldots, f_N, f_{N+1}]^\intercal = \mathbf{f}^{\mathbf{c}}_{\mathrm{LDIF}}(\mathbf{p})$. The scalar f_{N+1} is the predicted probability of the free space class.

The semantic loss L_S is a cross-entropy loss between the classification output vector $[f_1, \ldots, f_N]^\intercal$ and semantic ground truth. The ground truth free space probability for LiDAR targets is always zero as LiDAR measurements $\mathcal{L}$ are assumed to be located on objects. This loss is not evaluated for free space targets.

The geometric reconstruction loss

$$L_G = H\left(\left[l_{\mathrm{occupied}}, l_{\mathrm{free}} \right]^\intercal, \left[\sum_{i=1}^{N} f_i, f_{N+1} \right]^\intercal \right), \tag{4}$$

is the binary cross-entropy H between the sum of the semantic class probabilities $[f_1, \ldots, f_N]$ for all objects and the remaining free space probability f_{N+1}. It is available for all free space points with $[l_{\mathrm{occupied}}, l_{\mathrm{free}}]^\intercal = [0, 1]^\intercal$ and all LiDAR points with $[l_{\mathrm{occupied}}, l_{\mathrm{free}}]^\intercal = [1, 0]^\intercal$.

The consistency loss

$$L_C = \mathrm{JSD}(\mathbf{f}_{\mathrm{L},0}(\mathbf{p}), \ldots, \mathbf{f}_{\mathrm{L},m}(\mathbf{p})) \tag{5}$$

$$\begin{aligned} &= H\left(\frac{1}{m} \sum_{V \in \mathcal{V}_{\mathbf{p}}} \mathbf{f}_{\mathrm{L}}(\mathbf{c}_V, \Delta\mathbf{p}_V) \right) \\ &\quad - \frac{1}{m} \sum_{V \in \mathcal{V}_{\mathbf{p}}} H(\mathbf{f}^{\mathbf{c}}_{\mathrm{L}}(\Delta\mathbf{p}_V)), \end{aligned} \tag{6}$$

for a given coordinate $\mathbf{p}$ is the Jensen-Shannon divergence (JSD) between $m = |\mathcal{V}_{\mathbf{p}}|$ probability distributions predicted by the local segmentation functions $\mathbf{f}_{\mathrm{L}}$ on the support region $\mathcal{V}_{\mathbf{p}}$ of a consistency point $\mathbf{p}$. $\mathrm{H}(\mathbf{P})$ denotes the entropy of distribution $\mathbf{P}$. The JSD is symmetric and always bounded. Multiple local functions f_L make a prediction for the same position in the scene. The unweighted output of these local functions f_L exhibit grid artifacts between neighboring cells. The consistency loss acts as a regularizer by penalizing divergence between the grid cells without the need to specify any particular semantic or free space target label. Thereby, this loss term is available at any position within the scene, not only at regions where training targets from LiDAR points or sampled free space targets are occurring. We provide our numerically stable formulations of the geometric and consistency loss terms in the supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2021.3095302.

3.4 Implementation

All DNN network details and the hyperparameters are listed in the supplemental material in Tables 1 and 2, available online.

LiDAR Point Cloud Encoding. At the base layer we use a voxel-wise point cloud feature encoder from recent literature [42], [44]. The encoder transforms the raw input point set into a fixed-size bird's-eye view feature representation (Fig. 2, top-left) that corresponds to the spatial extent of the scene and is a suitable input for a convolutional feature extractor. Note that the encoder input feature space is in principle unrelated to the $\mathbb{R}^3$ domain of the generated completion function. This means that the point cloud encoder can make use of additional information of the sensor. We supply the reflectivity value of every LiDAR point as an extra feature. The positions of LiDAR points are encoded as separate coordinates relative to the mean position of the points within the voxel and the voxel center.

Decoder for Batch-Norm Conditioned Classification. Spatial encoding is implicitly modeled with a local output function $\mathbf{f}_{\mathrm{L}}$ that needs to be conditioned on the latent vector $\mathbf{c}_V$ of the feature extractor. This single-position classification function is implemented as a Multi-layer perceptron (MLP) that uses conditioned batch normalization (CBN) layers to express its dependency on the latent vectors [50]. Hereby, the resulting mean and variance of feature maps are generated by an affine transformation of the respective conditioning vectors. Our method divides the latent coding $\mathbf{c}_V$ into resolution-specific latent vectors $\mathbf{c}_V = (\mathbf{c}_{V1}, \mathbf{c}_{V2}, \mathbf{c}_{V3})$ and their associated relative positions $\Delta\mathbf{p}_V = (\Delta\mathbf{p}_1, \Delta\mathbf{p}_2, \Delta\mathbf{p}_3)$. This information then conditions the output function from coarse to fine: Thus beginning with the lowest resolution latent vector and adding more fine-grained information in the later layers of the MLP. The decoder diagram on the right of Fig. 2 illustrates this setup.

Training Details. Training the architecture involves common spatial augmentations of the input LiDAR point clouds in sensor coordinates. We use random uniform rotation over full 360°, random uniform scaling between $\pm 5\%$, random uniform translations between ± 5 cm. When training we use a top-view input grid with 256×256 voxels which results in a square with edge length of 40.96m within the scene. The grid is initially centered over the area where the accumulated training targets have been generated. The voxel grid is shifted off-center using normally-distributed offsets with standard deviation $\sigma = 8$ m. We sample a single free space point for each point in the input LiDAR point cloud and a single free space point within each empty voxel. Additionally, 2500 random scene locations are sampled and contribute to the consistency loss term, but do not have any other annotations. When training, only two out of the four nearest local functions f_L are evaluated for each query point to be able to include almost twice as many query training targets in a single batch. The two selected weighting coefficients w are scaled up accordingly. Depending on available VRAM and desired batch size the total number of training targets is clipped to a maximum value. For a KITTI scan with around 120 000 points and GPUs with 16 GB VRAM we selected a batch size of two and 400 000 training targets per GPU. Training on four Tesla-V100-GPUs with an effective batch size of eight took around four days to complete.

3.5 Inference and Visualization

We use latent conditioning vectors to define a function $\mathbf{f}^{\mathbf{c}}_{\mathrm{LDIF}}$ over $\mathbb{R}^3$ to represent geometry and semantics in a single classification vector. Depending on the task at hand this implicit representation necessitates different procedures to obtain explicit results. In any case, the completion function is evaluated for an arbitrary number of query coordinates at test time.

Semantic Scene Completion. We query the completion function for all corner points of all voxels within a dense voxel grid. Every corner point is shared by eight voxels. A voxel is marked as occupied when at least a single corner of the voxel is assigned any occupied class. The semantic label is averaged from all corners which are predicted as occupied. A threshold $\theta_{\text{empty voxel}} \in (0, 1)$ declares the free space probability under which a coordinate is considered occupied. This hyper-parameter controls the position on the precision-recall curve for the occupied class and is tuned on the training set to reach the maximum IoU of the occupied class.

LiDAR Semantic Segmentation. The positions of the LiDAR points themselves are used as query points at test time to obtain semantic predictions for a LiDAR point cloud. In this mode, it is previously known that none of the query positions can accurately be classified as *free space*. Therefore, the predicted class value is just the argmax over all non-free-space semantic classes.

Visualization. The $\mathbf{f}^{\mathbf{c}}_{\mathrm{LDIF}}$ function can be visualized by 3D meshes which represent the isosurface of the scalar free space function as close as possible (see Fig. 4, left column). From the $N+1$ semantic classes of the vector-valued $\mathbf{f}^{\mathbf{c}}_{\mathrm{LDIF}}$ function we extract the free space probability isosurface at a threshold $\theta_{\text{free space}} \in (0, 1)$. This isosurface $\{\mathbf{p} \in \mathbb{R}^3 | \mathbf{f}^{\mathbf{c}}_{\mathrm{LDIF}}(\mathbf{p})_{N+1} = \theta_{\text{free space}}\}$ resembles the estimated boundaries of all objects in the scene and therefore gives an idea of the learned scene representation. To extract the mesh, we use multiresolution IsoSurface Extraction (MISE) [11]. MISE evaluates points in an equally spaced grid from coarse to fine. By only evaluating the points of interest close to the isosurface the number of calculations is reduced considerably. Subsequently, the marching cubes algorithm is applied and the resulting mesh is refined by minimizing a loss term for each vertex using the proximity to the desired threshold value and the gradient information for faces of the mesh. This approach removes artifacts from the marching cubes algorithm and requires that gradients w.r.t. the position of input points are available. We query the $\mathbf{f}^{\mathbf{c}}_{\mathrm{LDIF}}$ function for all face-center positions of the resulting mesh and color the mesh based on these semantic predictions. Fig. 3 compares the mesh visualization and voxelized output that is obtained from the completion function.

We create a ground segmentation image to inspect the completion function at positions which are hidden in the scene. First, semantic segmentation is applied to the input point cloud. The LiDAR points that are identified to belong to one of the ground classes are selected. Then, the positions of the selected ground points are used for a bi-variate spline interpolation of all ground positions. A dense regular top-view grid of predicted ground positions is extracted. We query the completion function and display the predictions for the previously selected ground classes as image.

4 Experiments

In this section, we first describe the details of our training dataset and how it differs from the published Semantic KITTI

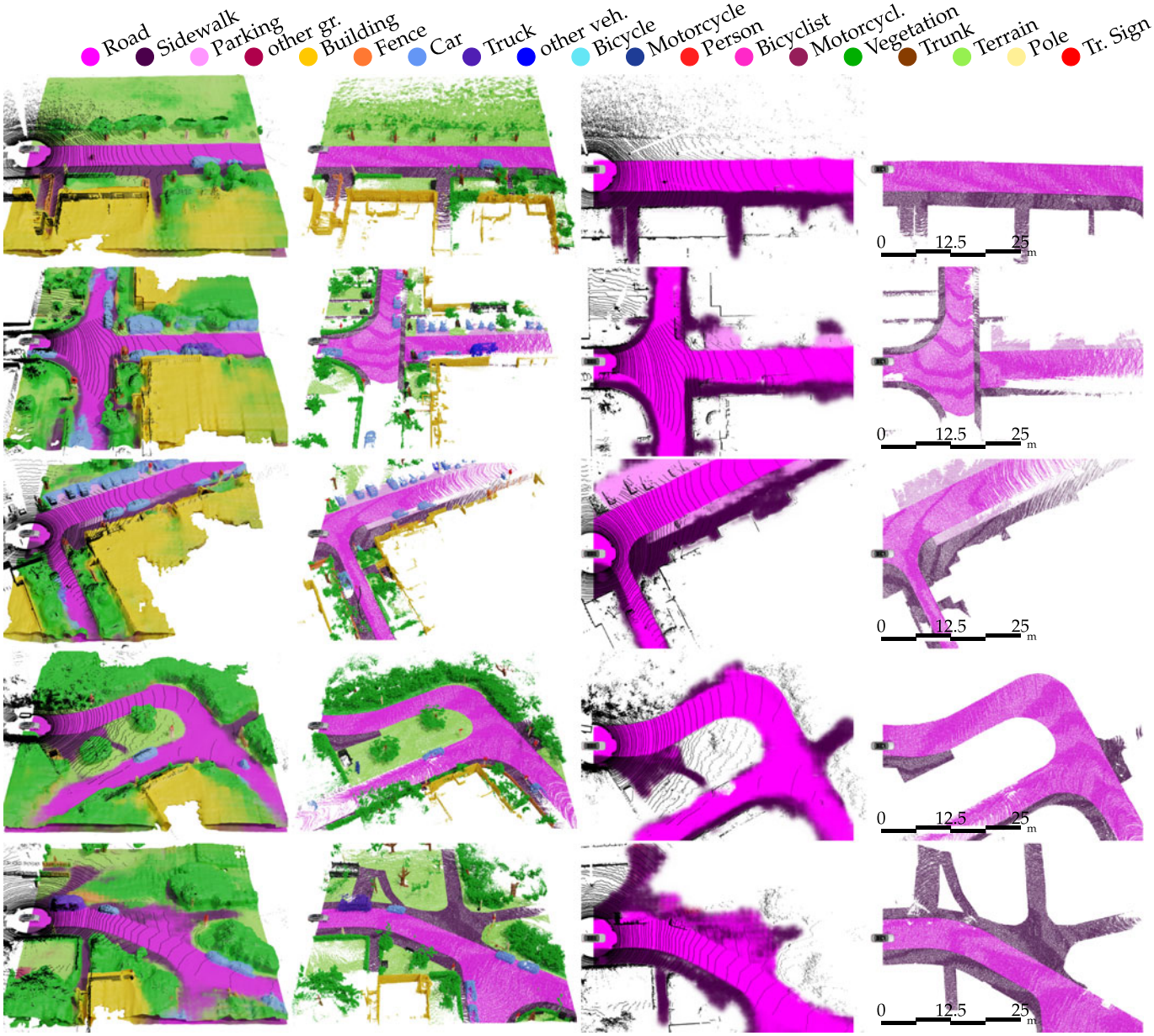


Fig. 4. Columns from left to right: Completed scene, accumulated LiDAR as ground truth, ground segmentation, and corresponding ground truth. Each row displays qualitative results and ground truth for a single scene on the Semantic KITTI validation set. The single LiDAR scan used as input for our method is depicted as an overlay of black points. The far-right section in each scene view demonstrates that our approach is able to operate on areas that include hardly any LiDAR measurements anymore. The method is data-based and takes advantage of experience from the training dataset to facilitate predictions based on the larger context of the scene. This is particularly visible from the completed courses of streets and sidewalks. We provide more qualitative results from diverse scenes of the test set in the supplemental material, available online.

scene completion dataset. Next, we introduce other published methods for real-world outdoor semantic scene completion and compare the quantitative results on the closed test set through the public benchmark. Finally, we perform an ablation study about the upsampling architecture, hyperparameter choices and semantic supervision signal.

4.1 Dataset and LiDAR Accumulation

The Semantic KITTI authors construct the semantic scene completion task from LiDAR scans of the KITTI Odometry dataset [51] and their corresponding semantic annotations [6]. The LiDAR sensor is a Velodyne HDL-64 that rotates with a frequency of 10Hz. The continuously measured LiDAR points from a full revolution are bundled into a LiDAR *scan*. The cut between scans is the negative x-axis in sensor coordinates so that every scan begins and ends looking backwards. LiDAR points are annotated with their respective semantic class.

The recordings are made up of 21 sequences in total. The data is split on a per-sequence basis: Ten sequences for training (19130 point clouds), one sequence for validation (4071 point clouds) and eleven sequences for testing (20351 point clouds). In the KITTI Odometry dataset the LiDAR scans are already ego-motion corrected. All points within a single 360° scan are transformed into the coordinate system located at the sensor's position in the moment the sensor was looking in the direction of the vehicle's front. In addition, the Semantic KITTI authors provide a frame-by-frame

Fig. 5. LiDAR scan (green) projected into reference RGB image. The vertical field of view of the KITTI LiDAR sensor only covers a range up to a few degrees over the horizon. Nevertheless, the resulting scene completion training targets cover objects at more than 2m over the ground since they are accumulated from more distant ego-positions.

point cloud registration. Sequences and registration are crucial as they allow to accumulate LiDAR measurements of a longer time span into a single fixed *reference coordinate system*. This process creates the annotations of the semantic scene completion task without requiring any additional manual annotations. The Semantic KITTI completion task combines a sequence of future LiDAR scans to generate the completion target of the scene at the time of the input LiDAR scan. This accumulation naturally includes future pathways of dynamic objects and therefore requires to predict object motion to solve the task in full. Section 4.2 details how we deviate from this handling of dynamic objects and explains the *static scene* accumulation targets that we propose instead.

The Semantic KITTI scene completion task uses a voxelized scene as output representation. A voxelized input LiDAR scan is also provided next to the raw LiDAR scan from the KITTI Odometry dataset. However, we do not use the provided voxelized scene to train our method as it is designed to classify individual positions. Instead of creating a labeled voxel grid from accumulated LiDAR measurements we use all of the individual points as training targets. The accumulated point clouds are sub-sampled to include only a maximum of 10 points within each original Semantic KITTI voxel. This reduces the overall dataset size and eliminates a large part of the redundancy in regions that are scanned by the sensor in multiple frames. The second column of Fig. 4 shows examples of the accumulation result. The input LiDAR point cloud is shown as on overlay over the prediction in the first column (left). We use the same extent for accumulation as the Semantic KITTI scene completion dataset: A square with 51.2m edge length where the ego-vehicle is located in the middle of an edge facing the center of the square.

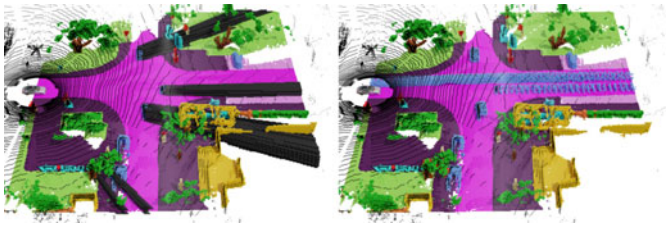
The difficulty of the scene completion task gets apparent when looking at the pronounced sparsity of the input point cloud in a distance of around 50 m from the sensor. In sparse regions most geometric details have to be inferred from scene context. It is apparent that there are geometric and semantic ambiguities within the 3D scenes which cannot be decided with high confidence from the single input LiDAR scan. Fig. 5 shows a projection of the LiDAR point cloud into the camera view of the ego-vehicle. The Velodyne HDL-64 sensor features a vertical field of view that at the top only covers a few degrees over the level horizon. Thus, in the vicinity of the ego-vehicle the LiDAR only covers a height of about 2m over ground. The scene completion target does however include geometry further up because it includes LiDAR points that were recorded from a greater distance of the ego-vehicle. This is another prominent ambiguity of the training data that requires a method to guess e.g., if there is a traffic sign attached to a pole without actual evidence from the sensor.

4.2 Handling of Dynamic Objects

We use *static* training and evaluation data for the semantic scene completion task. We regard this variant as more suitable for a meaningful evaluation of performance compared to the handling of dynamic objects in the original scene completion annotations. The KITTI Odometry scenes contain dynamic objects such as moving cars and pedestrians. These objects are additionally annotated with a *dynamic* flag. The original Semantic KITTI scene completion data accumulates the occupied voxels from dynamic objects in the reference frame just as the voxels of any other static object. Effectively this creates *spatio-temporal tubes* of moving traffic participants along their respective path. Therefore, fully solving the Semantic KITTI scene completion task requires predicting the future trajectories of traffic participants.

As we focus on geometric reconstruction of the scene in the instant of the input LiDAR scan, we take a different approach to ground truth targets for dynamic objects. Measurements on dynamic objects are omitted from the accumulation while the input LiDAR scan is kept unmodified. When accumulating LiDAR measurements, we only keep the single current scan on dynamic objects. By omitting the following scans over dynamic objects no trajectory *tube* is created. Next, it is necessary to ensure that no free space points get sampled within the extent of a dynamic object. As the object potentially moves from its initial position, the following LiDAR scans will record the initial position as free space. So to prevent free space targets within the actual object we record the shadow cast by the object in the first frame and treat the occluded regions as unseen regions where no free space points are sampled (see black regions in Fig. 6a). These two measures make the replicated geometry consistent in the presence of dynamic objects. The resulting set of training targets reflects the true scene at the moment of the input LiDAR scan. Areas where we cannot obtain consistent targets from future frames are ignored in the training.

In Fig. 6 we compare the two approaches for dynamic objects and show an example. We quantitatively measure the difference in performance when using the different dataset targets for evaluation. Note, that in this comparison, our method is trained on our *static* version of the data in both cases. This allows us to better judge the performance reported by the benchmark on the private test set. We see that there is almost no quantitative difference for the geometric completion evaluation (*Occupied IoU*) because static objects are prevalent over dynamic objects. However, for semantic scene completion we expect a significant difference. Object classes with a large proportion of dynamic voxels perform much worse if a method does not predict the object's movement. By not requiring our method to predict complicated object trajectories of even completely invisible objects we generate a consistent supervision signal.



(a) Static scene (b) Spatio-temporal tubes

Dataset variant	Occupied IoU	Semantic mIoU	Car IoU	Person IoU	Bicyclist IoU
(a) Static scene	57.8	**26.1**	**51.3**	**15.7**	**24.7**
(b) Spatio-temporal object tubes	57.6	24.0	45.6	3.3	0.9

Fig. 6. Our dataset ((a) static scene) and the official benchmark ((b) spatio-temporal tubes) handle dynamic objects differently. We remove all free space targets within the shadows of dynamic objects (marked as black regions) to obtain a consistent static scene. We evaluate the same model on both variants to measure this difference quantitatively. The impact on overall reconstruction performance in terms of IoU for *occupied* and *free space* class is marginal because of the prevalence of voxels belonging to static objects. However, the impact on IoU of small object classes that are primarily dynamic (e.g., *Person*, *Bicyclist*) is significant and leads to an increase in mIoU over all classes of about 2.1%. The comparison highlights that our method is in fact able to recognize smaller traffic participants. But an additional requirement to predict their motion will hide this ability.

Qualitative results of other methods [6], [8], [9] on Semantic KITTI show that they do not predict tubes as well, but instead also complete dynamic objects as if they were static. Having said this, the benchmark metric of course penalizes all methods equally for not predicting spatio-temporal objects tubes for dynamic objects.

4.3 Scene Completion Evaluation

In accordance with the scene completion benchmark [6] we use the mean intersection-over-union (mIoU) metric to assess both geometric completion performance and semantic segmentation accuracy. This metric is calculated on a per-voxel basis for the semantic scene completion task and on a per-point basis for single-scan LiDAR semantic segmentation. The semantic scene completion task is ranked by the mIoU value over all semantic classes including the free space class. The mere geometric completion performance is rated by the IoU value over all occupied classes combined, that is all classes except for free space.

The threshold $\theta_{\text{empty voxel}} \in (0, 1)$ is selected individually for each network variant based on the training set. This ensures that precision and recall values are balanced out, resulting in the respective maximum value for completion IoU and semantic mIoU. Fig. 7 plots the precision-recall-curve for the occupied class on the validation set together with IoU values for our best performing network.

We apply test-time augmentation (TTA) to our best performing approach for better comparison to the concurrent work JS3CNet. The regular predictions and predictions with TTA are submitted separately to the benchmark. TTA is implemented by augmenting the input point cloud at test time and averaging over the lattice grid predictions before generating the final voxel grid.

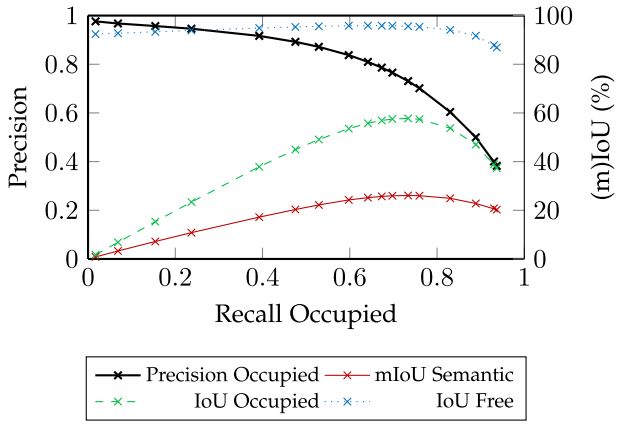


Fig. 7. *Precision-recall curve for the occupied class.* We plot the (m)IoU values for occupied, free and semantic classes of the baseline network variant. Markers are at the free space thresholds that are evaluated, interpolation in between.

4.4 Semantic Scene Completion Benchmark Results

We compare our approach against four recently published deep-learning-based methods on the challenging outdoor LiDAR semantic scene completion task. Quantitative results are reported by their respective authors on the benchmark and are compared in Table 1.

The performance of our method surpasses all other methods in pure geometric completion performance (57.7 percent). Here we exceed the second-best performing method LMSCNet-singlescale [7] by a margin of 1.0 percent. The authors of JS3CNet [8] only report benchmark results with TTA, so we use TTA as well for comparison. JS3CNet achieves a marginally higher mIoU (+0.2%) than our method with TTA, while being considerable inferior in geometric completion (-2.3% IoU). JS3CNet is more accurate on small object classes and less accurate on the larger ground classes. S3CNet [9] outperforms all other methods by a large margin on the semantics of small object classes, resulting in the best mIoU value. For the other object classes, it does however perform comparably or even worse to our method. Overall, when it comes to geometric accuracy, S3CNet underperforms significantly. This might be a result of the semantic post-processing steps.

4.5 Ablation Study

We use the Semantic KITTI validation split and the static scene data variant for evaluation of the ablation study. All ablation results are listed in Table 3.

Multi-Resolution Upsampling and Decoder Variants (Table 3, Architecture). The individual local functions are arranged in a grid where each cell has an edge length of 0.32m. The encoder uses a number of pooling layers and generally produces feature maps at lower resolutions of up to 16 times the output grid size. Our baseline Local-DIFs variant achieves a high resolution output grid by two independent upsample approaches. The first is upsampling and concatenating the lower resolution feature maps progressively in the encoder. The second is to supply pairs of relative coordinates and conditioning vectors for different resolutions. The decoder then handles the fusion of multiple feature maps. The conditioned-batch normalization (CBN)

TABLE 1
Quantitative Scene Completion Results for Our Method and Recently Published Approaches on the Semantic KITTI Scene Completion Benchmark (in Intersection-Over-Union, Higher is Better)

	Geometric Completion	Semantic Completion																		
Method / IoU [%]	Occ.	mIoU	Road	Sidewalk	Parking	other gr.	Building	Fence	Car	Truck	other veh.	Bicycle	Motorcycle	Person	Bicyclist	Motorcycl.	Vegetation	Trunk	Terrain	Pole	Tr. Sign
TS3D [3], [6]	50.6	17.7	62.2	31.6	23.3	6.5	34.1	24.1	30.7	4.9	0.1	0.0	0.0	0.0	0.0	0.0	40.1	21.9	33.1	16.9	6.9
LMSCNet-singlescale [7]	56.7	17.6	64.8	34.7	29.0	4.6	38.1	21.3	30.9	1.5	0.8	0.0	0.0	0.0	0.0	0.0	41.3	19.9	32.1	15.0	0.8
JS3CNet † [8]	56.6	23.8	64.7	39.9	34.9	**14.1**	39.4	30.4	33.3	**7.2**	12.7	14.4	8.8	8.0	5.1	0.4	43.1	19.6	40.5	18.9	15.9
S3CNet [9]	45.6	**29.5**	42.0	22.5	17.0	7.9	**52.2**	**31.3**	31.2	6.7	**16.1**	**41.5**	**45.0**	**45.9**	**35.8**	**16.0**	39.5	**34.0**	21.2	**31.0**	**24.3**
Local-DIFs (ours)	57.7	22.7	67.9	42.9	40.1	11.4	40.4	29.0	34.8	4.4	4.8	3.6	2.4	2.5	1.1	0.0	42.2	26.5	39.1	21.3	17.5
Local-DIFs + TTA †	**58.9**	23.6	**69.6**	**44.5**	**41.8**	12.7	41.3	30.5	**35.4**	4.7	4.7	3.6	2.7	2.4	1.0	0.0	**43.8**	27.4	**40.9**	22.1	18.5

†: *Method uses test-time augmentation.*

works as an attention mechanism between latent vector and query position. This variant is unique to decoder architectures based on DIFs.

We drop one of the two upsample approaches at a time resulting in two model architecture variants: *Local-DIFs-CBN* does not have transposed convolutions for upsampling in the encoder. *Local-DIFs-c3* uses only the highest resolution feature map in the decoder. Both completion and semantic scene completion performance is highest when using the baseline model that can rely on both upsample pathways. Building the decoder only on the high resolution feature map in *Local-DIFs-c3* reduces performance to a lesser extent than removing the transposed convolutions in the encoder in *Local-DIFs-CBN*. In both cases the drop in semantic scene completion is more noticeable than the drop in pure geometric completion. *Local-DIFs-CBN* reduces the number of trainable parameters compared to the baseline to about 78 percent (Table 2). The transposed convolutions account for a considerable share of the total parameters of the encoder. This experiment indicates that a decoder based on coarse grid cells together with coordinates as an attention mechanism can reduce the number of network weights required for upsampling.

Inspired by [24], [25], we construct a continuous representation decoder without the use of CBN. This third architecture variant *feature interpolation* performs bilinear interpolation on each resolution of the 2D feature grid to obtain a latent feature vector corresponding directly to the query position. As this feature only contains information about the xy-position we also concatenate the z-position of the query position onto this *positioned* vector. The resulting decoder structure contains almost the same number of parameters. While the overall performance is comparable to the baseline, the accuracy in semantic mIoU declines.

Grid Cell Size (Table 3, Cell Size). We review the impact of the architecture's grid cell size by scaling the base cell size of 0.32m to {75.0%, 87.25%, 150%, 200%} of its original value. The lower resolution feature maps as well as the input voxelization resolution are scaled accordingly. Larger grid cells tend to have only a negligible impact on the large ground object classes. However, semantic mIoU drops due to overall lower accuracy over all classes.

Loss weighting (Table 3, Loss). The individual loss weights $\lambda_{\{S,G,C\}}$ of the baseline network are $\lambda_S = 7.5, \lambda_G = 2.0, \lambda_C = 1.0$. We vary this weighting towards a larger contribution of the semantic loss, a larger contribution of the geometric loss, and a disabled consistency loss. Reducing the semantic loss weight does help with geometric reconstruction accuracy. However, the semantic segmentation accuracy does not improve over the baseline level by a higher relative weighting.

Impact of Semantic Supervision Signal on Geometric Completion Quality (Table 3, Data). Previous work uses deep neural networks to perform geometric scene completion both with

TABLE 2
Network Parameter Count for Architecture Variants

Variant	Σ	Point feat.	Encoder		Decoder
			Convs	Upsample	
Local-DIFs	9 892 788	1280	7 123 648	1 656 192	1 111 668
Local-DIFs-CBN	7 712 308	1280	7 123 648	0	587 380
Local-DIFs-c3	9 556 340	1280	7 123 648	1 656 192	775 220
Feature interp.	9 897 364	1280	7 123 648	1 656 192	1 116 244

TABLE 3
Quantitative Results of Baseline and Ablations on the Validation Set (Higher is Better)

	Variation	Geometric Completion IoU Occ.	Geometric Completion Precision	Geometric Completion Recall	Semantic Completion mIoU
	Local-DIFs (Baseline)	57.8	73.1	73.4	**26.1**
	(Local-DIFs + TTA)	(58.5)	74.2	73.5	(26.9)
Arch.	Local-DIFs-CBN	55.4	71.9	70.8	23.8
	Local-DIFs-c3	57.1	72.7	72.6	24.2
	Feature interpolation	57.4	73.0	73.0	25.5
Cell size	Cell size 75.0 %	54.1	70.6	69.8	23.8
	Cell size 87.5 %	57.1	73.8	71.7	25.6
	Cell size 150 %	56.7	71.6	73.1	24.1
	Cell size 200 %	56.7	72.5	72.3	23.3
Loss	$\lambda_S = 15, \lambda_G = 1$	55.6	71.1	71.7	24.0
	$\lambda_S = 3.75, \lambda_G = 4$	**58.2**	74.5	72.7	24.7
	$\lambda_C = 0$	56.9	72.0	73.0	25.0
Data	Simplified sem.	57.8	74.1	72.3	(38.8)
	Without sem.	57.9	73.6	73.1	(57.9)

and without semantic understanding of objects or scenes. This choice primarily depends on the existence on semantic ground truth annotations. Previous experiments suggest a correlation between semantic classification of objects on the SUNGC dataset and the accuracy on geometric completion of the scene [4]. We investigate if the semantic supervision signal helps with understanding objects in the scene and therefore also with geometric reconstruction.

We compare our baseline model on the validation set with two models that are trained on variants of the training dataset. First, we map the 19 semantic classes of the semantic KITTI dataset to a simpler set of only 9 classes. For instance, similar object classes are pooled into categories for small and large traffic participants. Second, we omit semantic classes altogether and only differentiate between *occupied* and *free* while training. Quantitative results are listed in Table 3 grouped under *Data*. The performance on geometric completion is almost unaffected by semantic supervision: 57.6 percent for the baseline versus 57.9 percent without semantic predictions. It is still noteworthy that the seemingly more difficult task of semantic scene completion is solved by a network of the same size almost without a loss in geometric completion performance. This suggests that the semantic and geometric completion task are indeed related.

Impact of Scene Completion Training Data. We analyze the single frame segmentation performance measured by the mIoU over the segmentation of all input LiDAR points. For this purpose, we train our method on single LiDAR scans with free space sampling and compare it to the baseline trained for scene completion on accumulated data. The networks are identical and the accumulated scene completion targets are a super set of the semantic segmentation of a single LiDAR scan. The segmentation performance of the scene completion model with accumulated supervision is almost 6 percent lower than that of the model only trained on single frame segmentation. Smaller object classes see the strongest declines. The quantitative results of this comparison and the differences in IoU scores are listed in the supplemental material, available online.

5 Discussion

It is noteworthy that we can compete on a benchmark based on a voxelized representation even though we do not use voxels as input or training targets. The voxelized scene that we generate from a post-processing step is more accurate than the end-to-end learned voxelization of other methods. We believe that our continuous representation benefits the learning of a spatially accurate scene representation. Voxelization causes quantization noise in the input signal and the supervision signal which we can avoid entirely.

We have analyzed how the generated scene completion function behaves when confronted with sparse measurements. The ground segmentation images illustrate that the representation generalizes to areas that are never directly observed with the LiDAR sensor. Our method learns to interpolate the course of the road, sidewalks or parking areas between measurements. Fig. 8 highlights completion modes for areas that are highly predictable (top row) and areas where completion is based on a best guess from the prior data distribution obtained from the training dataset (bottom row). For the latter part we say that the DNN completes the scene from experience when presented with practically no evidence from measurements. We examine another aspect of the scene completion function and plot the results in the right column of Fig. 8. As before, we create a mesh that approximates the decision surface of the completion function at a certain threshold for the free space probability. In addition, we determine the gradient of the free space value w.r.t. to the surface normal. The magnitude of this gradient is now transformed into a pseudo-color of the mesh. With a larger magnitude the transition from free space to an occupied class gets sharper. It is clearly visible that the ground level has a sharp transition even in high distances as it is easier to predict. Smaller objects show generally smaller gradients at their surfaces. But it is also noteworthy that the invisible rear side of objects as well as the predicted clouds of parking-car-probabilities have a small magnitude. Meaning that there is a softer transition in the completion function. It appears that the free space gradient correlates with the certainty of the spatial position of a surface. However, it can not be considered a well-calibrated measure of uncertainty in the output, but probably more as an indication of such.

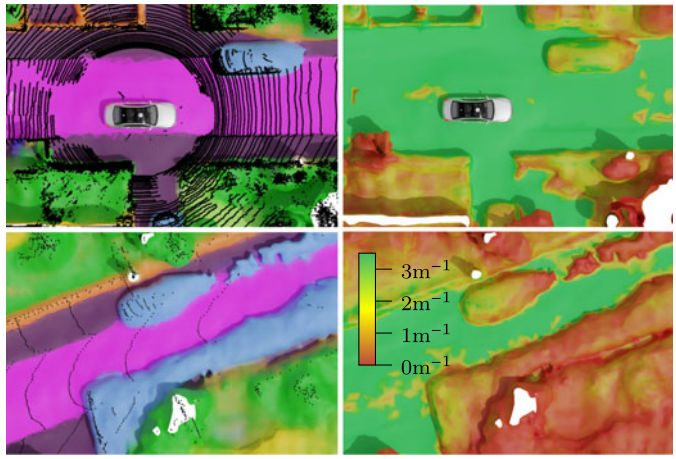


Fig. 8. Left: Top-view scene completion. Right: Magnitude of the gradient of the free space probability w.r.t. the surface normal. The top row demonstrates a highly predictable completion of road surface, the boundaries of the sidewalk, and a car in proximity to the ego-vehicle. Far away from the ego-vehicle, the bottom row shows how our method guesses the most likely classification of each individual scene coordinate in the absence of almost all evidence from actual measurements. The scene completion function is *softer* at these object boundaries (red surfaces).

We identify a failure mode of our method when it comes to the representation of fine geometric details and the drop of single frame segmentation performance as analyzed in the ablation about completion versus segmentation training data (Section 4.5, final paragraph). This loss in segmentation performance is significant given that the segmentation is derived from the exact same input point cloud. We do not have a definitive explanation for the magnitude of this circumstance. A possibility is to attribute the drop to the domination of the learning process by the completion task that leads to poorer performance on the segmentation task. An effect that can similarly be observed in multi-task learning setups. Another effect that contributes is that the completion task exhibits many ambiguities

in the areas where the input point cloud is sparse. There predictions are dominated by the dataset prior where small object classes are underrepresented. The convolutional architecture shares all weights over the spatial scene extent so that this kind of label noise contributes to the blurring of smaller object classes.

6 Conclusion and Future Work

We presented a novel approach to predict a semantically completed 3D scene from a single LiDAR scan. Our method is able to infer 3D geometry and semantics in sparsely measured areas from context and prior experience. In doing so we address two essential challenges: The first is to use LiDAR data and the included free space information as supervision signal. The second is being able to process large spatial extents for outdoor use while maintaining a high spatial resolution of the predicted completion at the same time. The key aspect is to encode LiDAR point clouds in a structured latent representation that is then decoded using local deep implicit functions at multiple resolutions. The output representation can be post-processed to obtain a voxel representation or 3D meshes for visualization purposes. We believe that we have set an important LiDAR-only baseline in the emerging field of large-extent outdoor scene completion.

Our approach surpasses all other methods on the challenging voxel-based Semantic KITTI scene completion benchmark in terms of geometric completion IoU (+1.0%). The ablation experiments demonstrated the advantage of the multi-resolution latent grid over a single resolution and verify the selected hyper-parameters. We showed that learning semantic classes along with geometry does not induce a performance penalty on the geometric completion performance. Uncertainty is inherent in the real-world scene completion task. As future work it will be rewarding to address this uncertainty by means of calibrating the network output or learning of a mapping to uncertainty from the input data. A well-calibrated uncertainty estimate will help to take full advantage of learning-based scene completion.

References

[1] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Niebner, "ScanComplete: Large-scale scene completion and semantic segmentation for 3D scans," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4578–4587.

[2] M. Firman, O. Mac Aodha, S. Julier, and G. J. Brostow, "Structured prediction of unobserved voxels from a single depth image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 5431–5440.

[3] M. Garbade, Y.-T. Chen, J. Sawatzky, and J. Gall, "Two stream 3D semantic scene completion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 416–425.

[4] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 190–198.

[5] D. Stutz and A. Geiger, "Learning 3D shape completion from laser scan data with weak supervision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1955–1964.

[6] J. Behley *et al.*, "SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9296–9306.

[7] L. Roldão, R. de Charette, and A. Verroust-Blondet, "LMSCNet: Lightweight multiscale 3D semantic completion," 2020. [Online]. Available: http://arxiv.org/abs/2008.10559

[8] X. Yan *et al.*, "Sparse single sweep LiDAR point cloud segmentation via learning contextual shape priors from scene completion," in *Proc. Conf. Artif. Intell.*, 2021, pp. 3101–3109.

[9] R. Cheng, C. Agia, Y. Ren, X. Li, and L. Bingbing, "S3CNet: A sparse semantic scene completion network for LiDAR point clouds," in *Proc. 4th Conf. Robot Learn.*, 2020.

[10] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5939–5948.

[11] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3D reconstruction in function space," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4460–4470.

[12] M. Michalkiewicz, J. K. Pontes, D. Jack, M. Baktashmotlagh, and A. Eriksson, "Implicit surface representations as layers in neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 4742–4751.

[13] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 165–174.

[14] C. Rist, D. Schmidt, M. Enzweiler, and D. Gavrila, "SCSSnet: Learning spatially-conditioned scene segmentation LiDAR point clouds," in *Proc. Intell. Vehicles Symp.*, 2020, pp. 1086–1093.

[15] A. Dai, C. Diller, and M. Niessner, "SG-NN: Sparse generative neural networks for self-supervised scene completion of RGB-D scans," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 846–855.

[16] L. P. Tchapmi, C. B. Choy, I. Armeni, J. Gwak, and S. Savarese, "SEGCloud: Semantic segmentation of 3D point clouds," in *Proc. Int. Conf. 3D Vis.*, 2017, pp. 537–547.

[17] Y. Liao, S. Donné, and A. Geiger, "Deep marching cubes: Learning explicit surface representations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2916–2925.

[18] C. Häne, S. Tulsiani, and J. Malik, "Hierarchical surface prediction for 3D object reconstruction," in *Proc. Int. Conf. 3D Vis.*, 2017, pp. 412–420.

[19] G. Riegler, A. Osman Ulusoy, and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3577–3586.

[20] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2088–2096.

[21] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-CNN: Octree-based convolutional neural networks for 3D shape analysis," *ACM Trans. Graph.*, vol. 36, no. 4, 2017, Art. no. 72.

[22] A. X. Chang *et al.* "ShapeNet: An information-rich 3D model repository," Stanford Univ. — Princeton Univ. — Toyota Technological Inst. Chicago, Tech. Rep. 2015.

[23] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Niessner, and T. Funkhouser, "Local implicit grid representations for 3D scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6000–6009.

[24] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 523–540.

[25] J. Chibane, T. Alldieck, and G. Pons-Moll, "Implicit functions in feature space for 3D shape reconstruction and completion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6968–6979.

[26] K. Genova, F. Cole, A. Sud, A. Sarna, and T. Funkhouser, "Local deep implicit functions for 3D shape," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 4856–4865.

[27] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proc. 4th Eurographics Symp. Geometry Process.*, 2006, pp. 61–70.

[28] D. Stutz and A. Geiger, "Learning 3D shape completion under weak supervision," *Int. J. Comput. Vis.*, vol. 128, pp. 1162–1181, 2020.

[29] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "PCN: Point completion network," in *Proc. Int. Conf. 3D Vis.*, 2018, pp. 728–737.

[30] J. Gu *et al.*, "Weakly-supervised 3D shape completion in the wild," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 283–299.

[31] L. Roldão, R. de Charette, and A. Verroust-Blondet, "3D semantic scene completion: A survey," 2021. [Online]. Available: https://arxiv.org/abs/2103.07466

[32] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 746–760.
[33] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2432–2443.
[34] A. Chang *et al.*, "Matterport3D: Learning from RGB-D data in indoor environments," in *Proc. Int. Conf. 3D Vis.*, 2017, pp. 667–676.
[35] H. Alhaija, S. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *Int. J. Comput. Vis.*, vol. 126, pp. 961–972, 2018.
[36] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3213–3223.
[37] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 833–851.
[38] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767
[39] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou, "Tangent convolutions for dense prediction in 3D," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3887–3896.
[40] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LiDAR point cloud," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1887–1893.
[41] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet++: Fast and accurate LiDAR semantic segmentation," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2019, pp. 4213–4220.
[42] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12 697–12 705.
[43] C. Rist, M. Enzweiler, and D. Gavrila, "Cross-sensor deep domain adaptation for LiDAR detection and segmentation," in *Proc. IEEE Intell. Vehicles Symp.*, 2019, pp. 1535–1542.
[44] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4490–4499.
[45] Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-voxel CNN for efficient 3D deep learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 963–973.
[46] M. Gerdzhev, R. Razani, E. Taghavi, and B. Liu, "TORNADO-Net: Multiview total variation semantic segmentation with diamond inception module," 2020. [Online]. Available: https://arxiv.org/abs/2008.10544
[47] H. Tang *et al.*, "Searching efficient 3D architectures with sparse point-voxel convolution," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 685–702.
[48] S. Liu *et al.*, "See and think: Disentangling semantic scene completion," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 263–274.
[49] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.
[50] H. de Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville, "Modulating early visual processing by language," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6594–6604.
[51] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

Christoph B. Rist received the bachelor's and master's degrees in electrical engineering and information technology from the Karlsruhe Institute of Technology, Karlsruhe, Germany, in 2014 and 2017, respectively. He is currently working toward the PhD degree with the Intelligent Vehicles Group, TU Delft, Delft, The Netherlands while working in the Corporate Research of Mercedes-Benz AG in Stuttgart (Germany). His current research focuses on LiDAR perception for autonomous driving.

David Emmerichs received the bachelor's and master's degrees in physics from the RWTH Aachen University, Aachen, Germany, in 2016 and 2018, respectively. He is currently working toward the PhD degree at IWR, Heidelberg University, Heidelberg, Germany while working in the Corporate Research of Mercedes-Benz AG, Stuttgart, Germany. His current research focuses on LiDAR perception for autonomous driving.

Markus Enzweiler received the PhD degree in computer science from the University of Heidelberg, Heidelberg, Germany, in 2011. From 2010, he was with Mercedes-Benz AG R&D in Stuttgart, Germany, most recently as a technical manager for LiDAR and camera. He co-developed the Daimler vision-based pedestrian detection system which is available in Mercedes-Benz cars. In 2021, he moved to the Esslingen University of Applied Sciences as a full professor for Autonomous Mobile Systems. His current research focuses on scene understanding for mobile robotics. In 2012, he received the IEEE Intelligent Transportation Systems Society Best PhD Dissertation Award and the Uni-DAS Research Award for his work on vision-based pedestrian recognition. He is a part of the team that won the 2014 IEEE Intelligent Transportation Systems Outstanding Application Award. In 2014, he was honored with a junior-fellowship of the Gesellschaft für Informatik.

Dariu M. Gavrila received the PhD degree in computer science from the University of Maryland at College Park, Maryland, in 1996. From 1997, he was with Daimler R&D, Ulm, Germany, where he became a distinguished scientist. He led the vision-based pedestrian detection research, which was commercialized 2013-2014 in various Mercedes-Benz models. In 2016, he moved to TU Delft (The Netherlands), where he since heads the Intelligent Vehicles Group as full professor. His research deals with sensor-based detection of humans and analysis of behavior, recently in the context of the self-driving cars in urban traffic. He received the Outstanding Application Award 2014 and the Outstanding Researcher Award 2019, both from the IEEE Intelligent Transportation Systems Society.