SCSSnet: Learning Spatially-Conditioned Scene Segmentation on LiDAR Point Clouds

Christoph B. Rist[a,b], David Schmidt[a], Markus Enzweiler[a] and Dariu M. Gavrila[b]

***Abstract*— This work proposes a spatially-conditioned neural network to perform semantic segmentation and geometric scene completion in 3D on real-world LiDAR data. Spatially-conditioned scene segmentation (SCSSnet) is a representation suitable to encode properties of large 3D scenes at high resolution. A novel sampling strategy encodes free space information from LiDAR scans explicitly and is both simple and effective. We avoid the need for synthetically generated or volumetric ground truth data and are able to train and evaluate our method on semantically annotated LiDAR scans from the Semantic KITTI dataset. Ultimately, our method is able to predict scene geometry as well as a diverse set of semantic classes over a large spatial extent at arbitrary output resolution instead of a fixed discretization of space.**

Our experiments confirm that the learned scene representation is versatile and powerful and can be used for multiple downstream tasks. We perform point-wise semantic segmentation, point-of-view depth completion and ground plane segmentation. The semantic segmentation performance of our method surpasses the state of the art by a significant margin of 7 % mIoU.

I. Introduction

Perception systems need to reason about their environment given partial observations from sensors. For many applications, including autonomous driving, a decision which area is occupied by an object and an understanding of its semantic meaning is required. While today's LiDAR sensors and RGB-D cameras already gather accurate 3D information of their surroundings from a sensor-view perspective, we are interested in an occupancy and semantic estimation of the full Euclidean space around the robot. Point-of-view sensor data projected into 3D space is inherently sparse and creates incomplete object geometries because of occlusions. This leads to the ill-posed problem of scene completion as a composite of dense reconstruction from sparse measurements and estimation of hidden geometry. Obtaining ground truth data that includes the full scene geometry on a large scale or for complex outdoor areas is especially tedious.

Existing learning-based 3D completion approaches can broadly be categorized into sensor-view depth completion [1], single object shape completion [2], [3], and scene completion on synthetic [4], [5] or small-scale tabletop [6] indoor datasets. Unlike image-space completion (e.g. [7], [8]) the scene and object completion task requires an encoding of 3D space where existing works focus most commonly on voxelization [4], [6], [9], [5], [3]. However, this results in satisfactory output resolutions only for volumes of limited extent.

[a]Mercedes-Benz AG, Stuttgart
[b]Intelligent Vehicles Group, TU Delft
Primary contact: christoph_bernd.rist@daimler.com

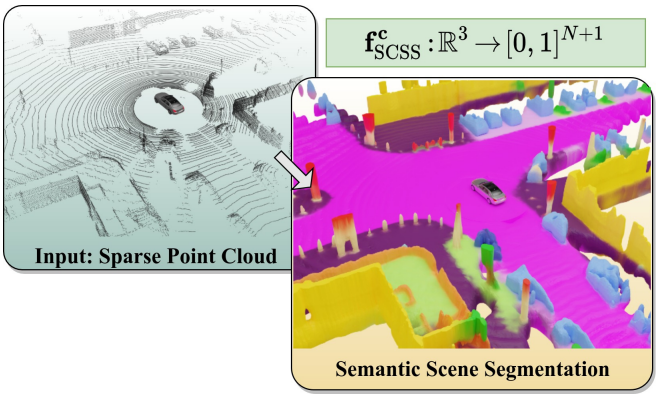

Fig. 1: **Visualizations of the scene segmentation function $\mathbf{f}_{\text{SCSS}}$ created from a single LiDAR frame**: Our method completes sparse LiDAR measurements over a large outdoor area and predicts from 19 semantic classes. The underlying representation can be applied to large spatial extents and is not tied to any fixed output resolution. The generated function $\mathbf{f}^{\mathbf{c}}_{\text{SCSS}} : \mathbb{R}^3 \rightarrow [0,1]^{N+1}$ maps any point in $\mathbb{R}^3$ to a probability vector over N semantic classes and free space.

We see two reasons that have impeded the successful transfer of semantic scene completion into robotics and autonomous driving: The lack of real-world road scene data and a suitable representation of large areas of 3D space that handles different levels of detail well. In this paper both issues are addressed by proposing an end-to-end trainable approach for semantic scene segmentation from LiDAR data. Our method does not require any synthetic data for pretraining or a generated prior distribution of object shapes to choose from. Instead, we implement single-point targets as supervision signal that conveys information about objects and free space of a scene.

Recent work [10], [11], [12], [13] proposes to encode geometry implicitly as a function over 3D space, mapping any point to a measure for occupation or signed distance to an object. We base our method on this idea and name our output function $\mathbf{f}^{\mathbf{c}}_{\text{SCSS}}(\mathbf{p})$. This function represents the desired geometric and semantic segmentation of 3D space implicitly. A point of interest $\mathbf{p} \in \mathbb{R}^3$ is the input to this function and gets classified into either free space or one of N semantic object categories by an output probability vector $\mathbf{f}^{\mathbf{c}}_{\text{SCSS}}(\mathbf{p}) \in [0,1]^{N+1}$. The dependence of $\mathbf{f}^{\mathbf{c}}_{\text{SCSS}}$ on the LiDAR scan of the

scene is expressed by the *conditioning* vector $\mathbf{c}$. The feature vector $\mathbf{c}$ is generated from the LiDAR scan and parameterizes the $\mathbf{f}^{\mathbf{c}}_{\text{SCSS}}$ function to represent a particular scene. Output function $\mathbf{f}_{\text{SCSS}}$ and encoder to generate $\mathbf{c}$ are each represented by a neural network and trained by backpropagation. A visualization of the resulting function $\mathbf{f}_{\text{SCSS}}$ is depicted in Fig. 1. We show that our method avoids voxel resolution trade-offs and is suitable to work on large outdoor areas with arbitrary resolution.

II. Related Work

We categorize related work into completion methods on the sensor-view plane and completion in 3D Euclidean space. The amount of information that needs to be inferred or estimated from given measurements is generally larger in the latter case. In addition, we distinguish works on implicit isosurface space representation and the state of the art in semantic segmentation.

Depth Super-Resolution and Completion: Learning-based depth super-resolution and LiDAR upsampling approaches concern themselves with increasing the resolution of data samples that are expected to be evenly distributed over the image plane [14], [15], [16]. These methods are generally not categorized under the term *completion* in a narrow sense. In contrast, many learning-based methods for completion of a sparse LiDAR depth map employ various possibilities to use RGB images as guidance: One group of such methods is using multi-stage setups using intermediate surface normals from RGB [17], [1], occlusion boundary cues [18], [1], or semantic information [18]. Others use end-to-end trained deep neural networks (DNNs) proposing 2D-3D feature fusion [19] or confidence weights [20]. Yang et al.[21] formulates the dense depth completion as a maximum a-posteriori estimate over the given sparse depth and a previously observed image dataset.

Methods for LiDAR-only depth completion learn dense depth prediction with convolutional neural networks (CNNs)[22], [7], [8] or alternating direction neural networks (ADNNs)[23]. They operate on sparse LiDAR points projected onto the image plane. To handle such sparse data a sparsity invariant CNN formulation is proposed by [8]. Classical image processing methods are utilized by [24] for the depth completion task.

Object and Scene Completion: It creates a substantial new challenge to shift the completion task from an image or sensor-view space to Euclidean space. Hence methods need to reason about the structure of space in occluded or otherwise unseen areas. 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 [25]. [6] is a pioneering work to infer full scene geometry from a single depth image in an output space of voxelized signed distance functions (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.

[4], [5] predict a volumetric occupancy grid with semantic information end-to-end from voxelized SDFs as input. They apply their methods to synthetic indoor data from the SUNCG dataset. While [5] works on single RGB-D images only [4] extends this 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. Explicit fusion of single depth images with RGB data to infer voxelized SDFs and semantic segmentation is performed by [9]. [3] implements single object shape completion on real-world LiDAR data with shape priors from synthetic models and learns to predict a complete shape from partial LiDAR scans by sampling the maximum likelihood shape from this prior.

Geometry and Surface Representation: The choice of output representation for scene completion falls most commonly on a voxel occupancy grid [5], voxelized SDFs [3], [4], [6], [9], or interpolation and CRFs [26] for sub-voxel accuracy. To skip the intermediate SDF and train surface representation end-to-end, [27] introduces the a differentiable deep marching cubes algorithm which is still constrained by the underlying voxel resolution. In 3D representation the general trade-off between output resolution and computation resources is an issue [4]. Octree-based CNNs [28], [29], [30], [31] have been proposed to represent space at different resolutions and to perform gradual shape refinements.

Recent works [10], [11], [12], [13] propose to represent 3D shape implicitly as isosurface of an output function supported by 3D space that is represented as a DNN. They either use oriented surfaces [12] or watertight meshes [11] from ShapeNet [2] as synthetic full-supervision training targets. The scope of these approaches has so far been limited to single object reconstruction.

Semantic Segmentation: Numerous prior works focus on semantic classification of all observed data points meaning pixel-wise or point-wise classification. 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 [32], [33], [34] have defined a standard of relevant semantic classes for autonomous driving while simultaneously advancing the state of the art. CNN-architectures on RGB-Images for segmentation and detection [35], [36] have inspired sensor-view based approaches in the more recent LiDAR-based segmentation task [37], [38]. Neural network architectures adjust to the three dimensional nature of a segmentation or detection problem through voxelization of input data [39], [40], [41] or use of surface geometry [37]. The semantic segmentation problem on real-world data has only recently been advanced by the large-scale Semantic KITTI dataset [33] featuring point-wise semantic annotations on LiDAR together with a private test set and segmentation benchmark.

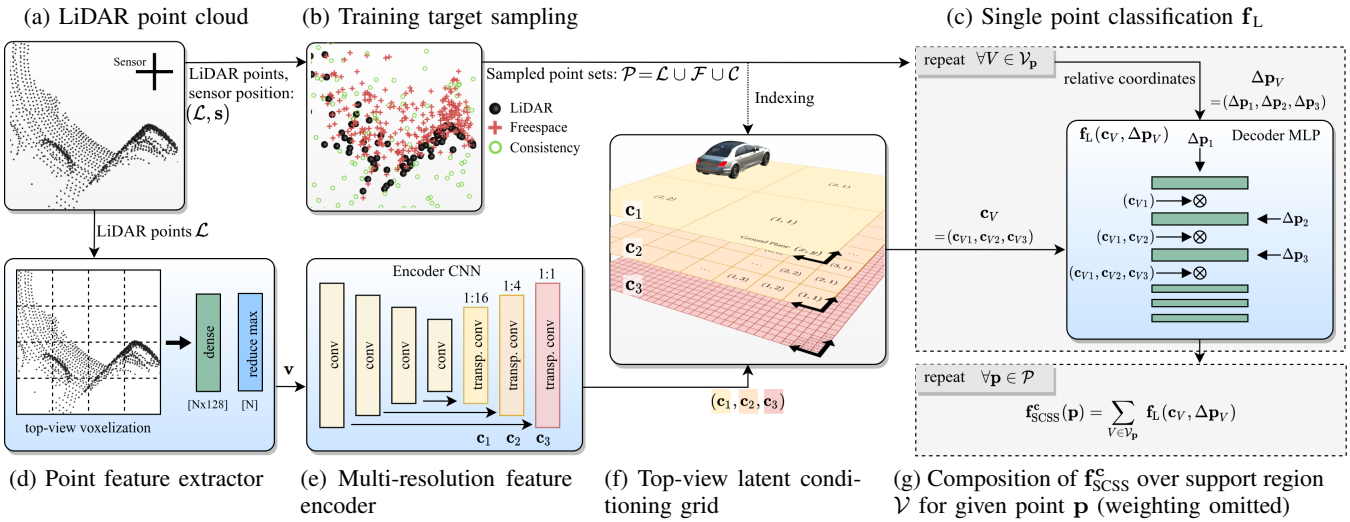

Fig. 2: **Composition of the segmentation function $\mathbf{f}^\mathbf{c}_\mathrm{SCSS}$**: (a): Input LiDAR scan consisting of points $\mathcal{L}$ and sensor position $\mathbf{s}$. (b): Sampling of additional free space points $\mathcal{F}$ between sensor position $\mathbf{s}$ and LiDAR points $\mathcal{L}$ for supervised training of scene geometry. (c): Point classification head for a single point using coordinates $\Delta\mathbf{p}_V$ relative to the position of a specific conditioning vector $\mathbf{c}_V$. (d): Top-view pseudo-image from LiDAR scan through voxelization. (e): U-net like multi-resolution feature encoder CNN. (f): CNN feature maps at three output resolutions are arranged spatially as top-view latent grid of conditioning vectors with direct correspondence to points in the scene through ground plane x-y-coordinates (offset in height between $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$ only for visualization, ego-vehicle for scale). (g): Each point $\mathbf{p}$ falls into a conditioning grid cell V. However, the point $\mathbf{p}$ is classified by all of the nearby conditioning vectors within the support region $\mathcal{V}_\mathbf{p}$ around this grid cell. The $\mathbf{f}^\mathbf{c}_\mathrm{SCSS}$ value for a point $\mathbf{p}$ is the weighted summation over these single point classifications $\mathbf{f}_\mathrm{L}$.

III. Proposed Approach

A. Overview

Our method takes as input a point cloud and outputs the corresponding Spatially Conditioned Scene Segmentation (SCSS) function $\mathbf{f}^\mathbf{c}_\mathrm{SCSS} : \mathbb{R}^3 \rightarrow [0,1]^{N+1}$. This function maps every 3D coordinate of a point $\mathbf{p}$ within the scene to a probability vector that we define to represent the semantic class of the point $\mathbf{p}$. Points 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 for a total of $N+1$ classes. Hence the $\mathbf{f}^\mathbf{c}_\mathrm{SCSS}$ function uniformly represents the geometric and semantic segmentation of space instead of only the physical boundaries of objects.

The components of the $\mathbf{f}^\mathbf{c}_\mathrm{SCSS}$ function are represented by a neural network $\mathbf{f}_\mathrm{L}$ with two distinctive inputs: The coordinate of interest $\Delta\mathbf{p}$ and a parameterization vector $\mathbf{c}_V$. In this context, *generating* an output function $\mathbf{f}^\mathbf{c}_\mathrm{L}$ means generating a parameterization (*conditioning*) vector $\mathbf{c}_V$ and therefore obtaining the conditioned function $\mathbf{f}^\mathbf{c}_\mathrm{L}$ that is just dependent on the remaining input coordinate $\Delta\mathbf{p}$.

Our approach to the composition of the $\mathbf{f}^\mathbf{c}_\mathrm{SCSS}$ 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 propose a convolutional encoding where the amount of conditioning information is tied to the spatial extent. To achieve this, multiple conditioning vectors are spatially distributed over a two-dimensional uniform grid in the xy-ground plane at a fixed resolution (Fig. 2f). 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. The fixed grid layout ensures that every location of the scene is evenly covered by conditioning vectors. The configuration of objects in outdoor scenes is assumed to be translation-invariant in x and y direction. Such as the encoding of the front of a car or a part of a tree is the same regardless of the absolute position of the object. On this account an implementation of the generation of the conditioning vectors by a convolutional neural network suggests itself. A schematic diagram of the point cloud encoding into the latent conditioning grid and composition of the output function is given in Fig. 2.

In the next chapter the details of the composition of the $\mathbf{f}^\mathbf{c}_\mathrm{SCSS}$ function from multiple conditioning vectors and grid resolutions are described. A sampling-based supervised training method from real-world LiDAR data is proposed and details on the used network architecture and inference procedure follow. In summary, our main contributions are:

- We formulate conditioning vectors that are distributed in a hierarchical spatial manner to parametrize an output function to classify positions in 3D space.
- We propose a point-based training scheme on LiDAR data for the 3D scene segmentation task.
- We validate that the 3D representation generalizes to occluded areas and is suitable for geometric reconstruction and semantic segmentation of space.

B. Spatially-distributed conditioning vectors

Composition of $\mathbf{f}^{\mathbf{c}}_{\text{SCSS}}$: Centerpiece of our method is the formulation of latent conditioning vectors that are spatially distributed and generated by a convolutional encoder network on LiDAR point clouds. Each individual conditioning vector $\mathbf{c}_V$ parameterizes a *local segmentation function* $\mathbf{f}_{\text{L}}(\mathbf{c}_V, \Delta\mathbf{p}_V)$ (see Fig. 2c) to classify a point of interest $\mathbf{p}$. While the extent of the support of individual local functions is $\mathbb{R}^3$ and therefore infinite, the classification will only be meaningful for points that are close to the conditioning vector's position. Hence, for a given point $\mathbf{p}$ we define a set of grid cells in its vicinity as *support region* $\mathcal{V}_\mathbf{p}$ of the point $\mathbf{p}$. Determining the support region for $\mathbf{p}$ means finding the grid cell that contains $\mathbf{p}$ and selecting this center cell together with a number of neighboring cells. The classification $\mathbf{f}^{\mathbf{c}}_{\text{SCSS}}$ of point $\mathbf{p}$ is the weighted mean of the individual function outputs based on the conditioning vectors at the grid cells in the support region:

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

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

$\mathbf{o}_V$ is the center position of a cell V and $\mathbf{c}_V$ is the conditioning vector at cell V. The weights w are introduced to value cells linearly according to their distance to $\mathbf{p}$. Naturally, further away conditioning vectors get assigned a lower weight. Intuitively, the spatial extent of a scene can be thought of as covered by overlapping function patches $\mathbf{f}_{\text{L}}$. Each function $\mathbf{f}_{\text{L}}$ 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 additional aspect of the composition of $\mathbf{f}^{\mathbf{c}}_{\text{SCSS}}$ is the use of three individual conditioning vectors from three different resolutions levels. The geometric structure of a scene is composed of different levels of detail: from the coarse position of the ground level and walls to more fine-grained details like curbstones, small objects and poles. We reproduce this hierarchy in the network structure to facilitate learning of a smooth representation with more details and without value jumps over cell boundaries. That is why a single conditioning vector $\mathbf{c}$ gets defined as the concatenation of three resolution-specific conditioning 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 like convolutional feature encoder, see Fig. 2e. The resolution ratios correspond to grid cells with $5.12\,\text{m}$, $1.28\,\text{m}$, and $0.32\,\text{m}$ edge length respectively.

The size of the support regions differs between the resolution levels. At lowest resolution we choose to only use the cell the point resides in ($|\mathcal{V}_1| = 1$). At $1{:}4$ resolution a centered 3×3 square is selected ($|\mathcal{V}_2| = 9$), and at $1{:}1$ resolution a 5×5 square ($|\mathcal{V}_3| = 25$). This results in a support region $\mathcal{V}_\mathbf{p}$ for point $\mathbf{p}$ that consists of all possible combinations of conditioning vectors $(\mathbf{c}_{V1}, \mathbf{c}_{V2}, \mathbf{c}_{V3})$ within the resolution-specific regions $\mathcal{V}_1$, $\mathcal{V}_2$, and $\mathcal{V}_3$. In total there are $1 \times 9 \times 25$ combinations for $\mathbf{c}_V$ and just as many conditioned local segmentation functions that are able to describe the single point $\mathbf{p}$ in the scene. When training, only two of these combinations of each point are drawn at random.

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 $\mathbf{f}_{\text{L}}$.

Point type	Source	Loss available		
		Semantic	Reconstr.	Consistency
LiDAR $\mathcal{L}$	Input data	✓	✓	✓
Free space $\mathcal{F}$	Gen. from input	✗	✓	✓
Consistency $\mathcal{C}$	Random uniform	✗	✗	✓

TABLE I: **Training targets:** LiDAR measurements and sampled free space points define the binary reconstruction target (occupied, free) used in the geometric reconstruction loss. The consistency loss is well-defined for all points in 3D space, regardless of semantic label.

C. Training on LiDAR data

Training target sampling: The point-classification neural network and feature encoder are trained using individual points and their associated training labels. We use three different kinds of training targets: LiDAR $\mathcal{L}$, free space $\mathcal{F}$, and consistency points $\mathcal{C}$. The set of 3D points of a LiDAR point cloud is denoted as $\mathcal{L}$. Naturally, the coordinates in $\mathcal{L}$ must all refer to the same sensor coordinate system so that the position of the LiDAR sensor itself is known for every measured point in $\mathcal{L}$.

A natural insight is that 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 model this explicitly for supervised training of scene geometry. Points that are randomly sampled on these straight lines form the set of free space points $\mathcal{F}$. Accordingly, these points get assigned the semantic class *free space*. This additional class allows to unambiguously define a classification label for every point within the scene.

To learn a realistic representation over a larger scene extent it is crucial to enforce classification consistency over the support region $\mathcal{V}_\mathbf{p}$ of a point $\mathbf{p}$. The idea is to bias the individual local segmentation functions $\mathbf{f}_{\text{L}}$ within $\mathcal{V}_\mathbf{p}$ in Eq. (1) to predict the same probability distribution at point $\mathbf{p}$. Therefore, we introduce the additional training target type *consistency points* together with an additional loss term. The consistency loss penalizes divergence without specifying any particular target label. Thereby, this loss term is available at any coordinate within the scene, not only at regions where LiDAR $\mathcal{L}$ or free space points $\mathcal{F}$ are occurring. A set of additional consistency points $\mathcal{C}$ is sampled uniformly within the scene boundaries to cause the network to make meaningful predictions even in completely occluded areas.

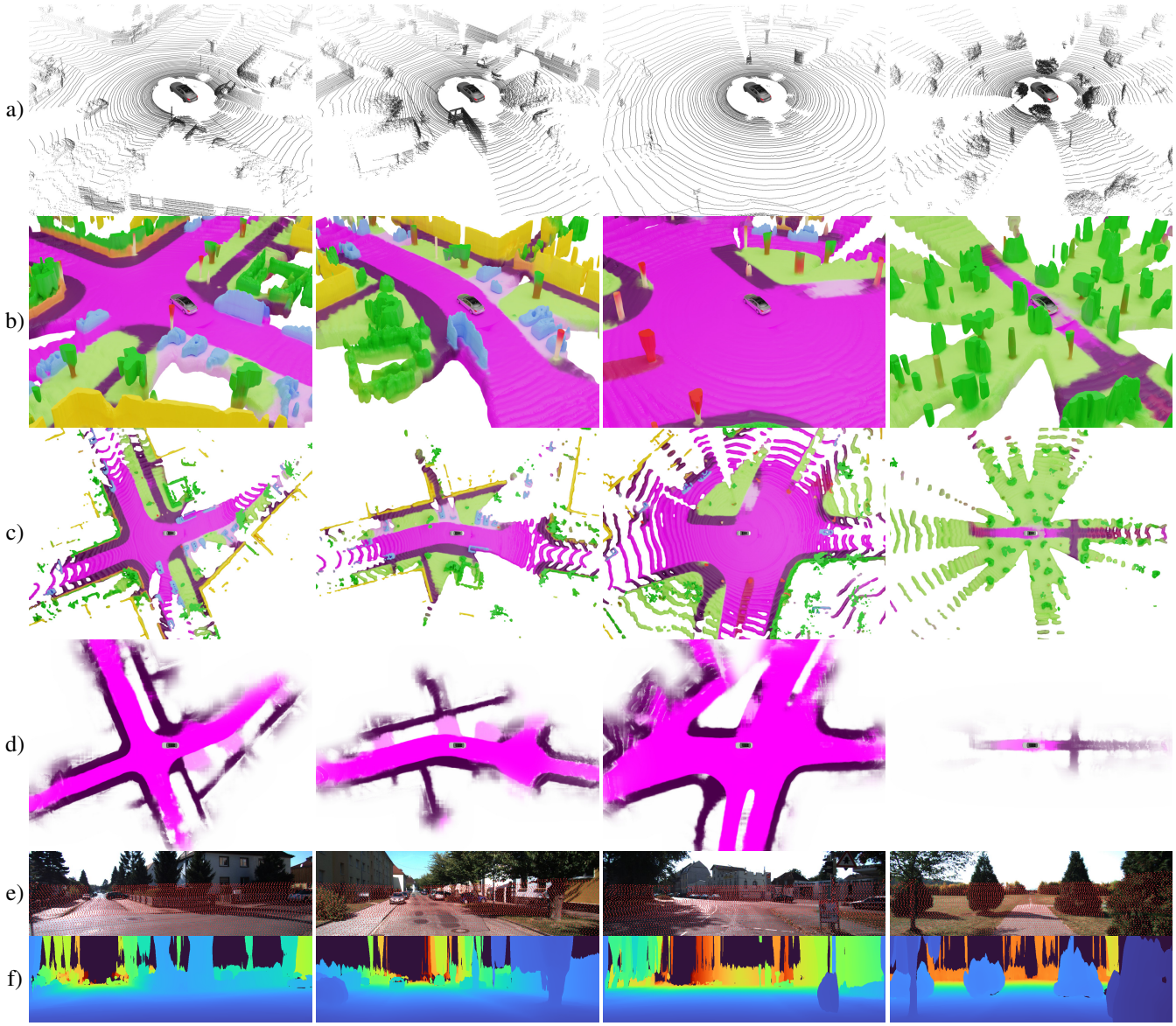

Fig. 3: **Qualitative results on KITTI Semantic test set:** Renders of the isosurface of the free space probability as semantically colored mesh [b, c] give an intuition how the learned function behaves for different scenes. The ground plane is estimated from semantic predictions and segmented into classes road, sidewalk, and parking space [d]. The top-view images cover an extent of $-45\,\mathrm{m}$ to $45\,\mathrm{m}$ behind and in front of the ego-vehicle. We create sensor-view dense depth images [f] from the $\mathbf{f}^{\mathbf{c}}_{\mathrm{SCSS}}$ representation by projection of the free space surface onto the image plane. Only sparse LiDAR data has been used [a]. The RGB image is given as a reference only (not used by our method) [e].

Loss function: Training the classifier involves the three different loss terms semantic, geometric reconstruction, and consistency loss. The availability of these losses at specific coordinates in the scene differs between the three different kind of sampled training targets as summarized in Table I. The overall loss

$$L = \lambda_S \sum_{\mathbf{p}\in\mathcal{L}} L_S + \lambda_R \sum_{\mathbf{p}\in\mathcal{L}\cup\mathcal{F}} L_R + \lambda_C \sum_{\mathbf{p}\in\mathcal{L}\cup\mathcal{F}\cup\mathcal{C}} L_C \quad (3)$$

is the weighted sum of the individual losses that are each summed over their respective training point sets. We write the predicted probability vector at position $\mathbf{p}$ as $[f_1, \ldots, f_N, f_{N+1}]^\intercal = \mathbf{f}^{\mathbf{c}}_{\mathrm{SCSS}}(\mathbf{p})$. 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 full classification output vector $[f_1, \ldots, f_{N+1}]^\intercal$ 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.

The geometric reconstruction loss

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

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 $[l_{\text{occupied}}, l_{\text{free}}]^\intercal = [0, 1]^\intercal$ and all LiDAR points with $[l_{\text{occupied}}, l_{\text{free}}]^\intercal = [1, 0]^\intercal$. The differentiation between semantic and reconstruction loss is only introduced to cover LiDAR points that do not carry a semantic label.

The consistency loss

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

$$= H\left(\frac{1}{m}\sum_{V \in \mathcal{V}_\mathbf{p}} \mathbf{f}_\text{L}(\mathbf{c}_V, \Delta\mathbf{p}_V)\right) - \frac{1}{m}\sum_{V \in \mathcal{V}_\mathbf{p}} H\left(\mathbf{f}_\text{L}(\mathbf{c}_V, \Delta\mathbf{p}_V)\right) \tag{6}$$

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 $\mathbf{f}_\text{L}$ on the support region $\mathcal{V}_\mathbf{p}$ of a consistency point $\mathbf{p}$. $\text{H}(\mathbf{P})$ denotes the entropy of distribution $\mathbf{P}$. The JSD is symmetric and always bounded.

D. Implementation

Point cloud encoding: At the base layer we use a voxel-wise point cloud feature encoder from recent literature [39], [41]. The encoder transforms the raw input point set into a fixed-size pseudo-image feature representation (Fig. 2d) 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 $\mathbb{R}^3$ support of the generated 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 extra feature.

Batch-norm conditioned classification: Spatial encoding is implicitly modeled with a local output function $\mathbf{f}_\text{L}$ that needs to be conditioned on the latent vector $\mathbf{c}_V$ of the feature extractor. This point classification function is implemented as an Multi-layer perceptron (MLP) that uses conditioned batch-normalization layers to express its dependency on the latent vectors[42]. 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 (see Fig. 2c).

Training details: Training the architecture involves common spatial augmentations of the input LiDAR point clouds themselves in sensor coordinates: random uniform rotation over the full 360°, random uniform scaling between $\pm 5\,\%$, random uniform translations between $\pm 5\,\text{cm}$. In addition, the voxelization grid is globally shifted off-center using normal-distributed offsets with $\sigma = 15\,\text{m}$. We sample two free space points for each LiDAR point in a scan and 2500 consistency locations per scan. Depending on available VRAM and desired batch size the number of training targets of each category is reduced by random drop-out. For a scan with 120k points, 16 GB VRAM and batch size of 4 we dropped $10\,\%$ of training targets. We found that a batch size of 4 was the minimum required for stable training.

E. Inference and Visualization

We use latent conditioning vectors to define a function $\mathbf{f}^\mathbf{c}_\text{SCSS}$ 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 function is evaluated for an arbitrary number of query coordinates at test time.

For point-wise semantic segmentation it is straightforward to use the positions of the set of LiDAR points $\mathcal{L}$ as test time query points to obtain semantic predictions for the point cloud itself. The predicted class value is the maximum probability over all non-free-space semantic classes $\text{argmax}\ (\mathbf{f}^\mathbf{c}_\text{SCSS}(\mathbf{p})_i)_{i \in [1,N]}$.

Sensor-view depth completion images as in Fig. 3f can be generated given the projection matrix from LiDAR coordinates to an image plane. Each image pixel corresponds to a down-projected straight line in LiDAR 3D coordinates. Tracing the free space probability of the output function $\mathbf{f}^\mathbf{c}_\text{SCSS}$ on these lines for every image pixel starting at the image plane at $d = 0$ until encountering a free space probability below a threshold value θ_thres yields the depth values for each pixel.

To visualize the $\mathbf{f}^\mathbf{c}_\text{SCSS}$ function we generate 3D meshes to represent the isosurface of a scalar function over $\mathbb{R}^3$ as close as possible (see Fig. 3b). From the $N+1$ semantic classes of the vector-valued $\mathbf{f}^\mathbf{c}_\text{SCSS}$ function we choose to visualize the free space probability isosurface at a certain threshold $\theta_\text{free space} \in (0, 1)$. This isosurface $\{\mathbf{p} \in \mathbb{R}^3 | \left(\mathbf{f}^\mathbf{c}_\text{SCSS}(\mathbf{p})\right)_{N+1} = \theta_\text{free space}\}$ corresponds to the 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]. We query the $\mathbf{f}^\mathbf{c}_\text{SCSS}$ function for all vertex positions of the resulting mesh and color the mesh based on these semantic predictions.

IV. Experiments

We conduct different types of experiments to validate our proposed training approach and output representation. Experiments are split into point-wise semantic segmentation and ground plane segmentation. Additionally we visualize the predicted free space isosurface. An overview is given in Fig. 3.

A. Dataset

We train our method on single LiDAR frames from the Semantic KITTI dataset that provide semantic annotations in 19 different classes for each LiDAR measurement. The point clouds of this dataset are taken from the ego-motion corrected KITTI Odometry dataset that is recorded using a

Method / IoU [%]	mIoU	Road	Sidewalk	Parking	other gr.	Building	Car	Truck	Bicycle	Motorcycle	other veh.	Vegetation	Trunk	Terrain	Person	Bicyclist	Motorcycl.	Fence	Pole	Tr. Sign
TangentConv [37]	40.9	83.9	63.9	33.4	15.4	83.4	90.8	15.2	2.7	16.5	12.1	79.5	49.3	58.1	23.0	28.4	8.1	49.0	35.8	28.5
DarkNet21Seg [33], [38]	47.7	91.4	74.0	57.0	26.4	81.9	85.4	18.6	26.2	26.5	15.6	77.6	48.4	63.6	31.8	33.6	4.0	52.3	36.0	50.0
DarkNet53Seg [33], [38]	49.9	**91.8**	74.6	64.8	27.9	84.1	86.4	25.5	24.5	**32.7**	22.6	78.3	50.1	64.0	36.2	33.6	4.7	55.0	38.9	**52.2**
SCSSnet (ours)	**57.6**	90.8	**75.5**	**68.0**	**31.9**	**89.8**	**95.9**	**49.6**	**36.3**	26.1	**42.4**	**83.5**	**61.5**	**66.7**	**47.9**	**50.8**	**22.2**	**62.3**	**44.5**	47.9

TABLE II: **Point-wise semantic segmentation performance** (higher is better): We compare our method to the three best-performing methods from [33]. These are based on sensor-view CNNs. We outperform all other methods on nearly all semantic categories.

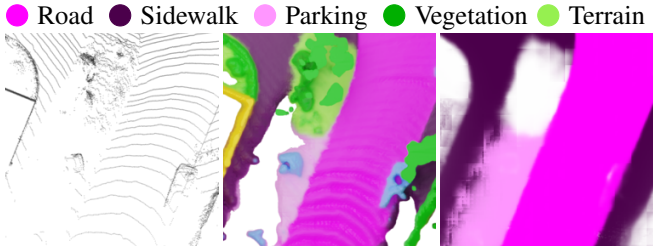

Fig. 4: **Ground plane estimation and segmentation**: Top-view, from left to right: zoom into sparse point cloud, corresponding geometric reconstruction, and segmentation of ground plane into road, parking, sidewalk and remaining classes. Our approach makes meaningful semantic predictions in partly or completely occluded areas from the ego-vehicle's point of view. Segmentation and extent of parking spaces are predicted beneath or behind occupying objects as well as continued over areas that are very sparse (bottom left and right corners).

single Velodyne HDL-64 sensor. We split the part containing public ground truth annotations into 22 001 frames for training and 1200 frames for validation.

B. Semantic Segmentation

To evaluate semantic segmentation performance we submit our results to the Semantic KITTI segmentation benchmark on the single 360° LiDAR frames category. The benchmark uses the widely-used mean intersection-over-union (mIoU) or Jaccard index to assess performance. The semantic classes itself and results are listed in Table II. Our method is compared to the three best performing LiDAR semantic segmentation methods from [33] and outperforms them by a significant margin.

C. Ground plane estimation and segmentation

We execute a simple ground plane estimate and query the output segmentation function for semantic labels on just this surface. The semantic classes of primary interest for a car on the ground plane are *Road*, *Parking*, and *Sidewalk*. All other classes are summed up and displayed as white background. The height value at positions that are not predicted as of any ground class are interpolated to generate a dense ground plane height grid for semantic evaluation. See Fig. 4 for the color coding and a close-up example of ground plane scene completion. More qualitative examples are listed in Fig. 3d. The learned representation is able to complete a meaningful segmentation in metric space of the scene even in great distances from the recording vehicle where LiDAR points are very sparse. We see some artefacts from the voxelized conditioning vector hierarchy only in entirely invisible regions. The ground plane segmentation exists as continuous function over space. Consequently the road surface can be extracted from the scene as a single object. This adds much value over individual LiDAR points or voxels of reduced resolution. The representation evades all sparsity issues that are inherently existent in LiDAR data.

V. Conclusion

This paper introduces spatially-conditioned scene segmentation networks (SCSSnet) as convolutional-generated representation for 3D geometry and semantic segmentation. In a series of experiments we show that the representation is versatile and powerful: It is suitable for scenes of a large extent without sacrificing resolution and precision. Inferred semantic and geometric decision boundaries are smooth and meaningful. Using this 3D scene representation we reach a semantic segmentation mIoU score of 57.6 % on the Semantic KITTI Benchmark significantly outperforming all other sensor-view based methods. An apparent next step is to train our method on time-accumulated point clouds and volumetric ground truth scene completion data to adapt to the full scene completion task.

References

[1] Y. Zhang and T. Funkhouser, "Deep Depth Completion of a Single RGB-D Image," in *CVPR*, 2018, pp. 175–185.

[2] A. X. Chang, T. A. Funkhouser, *et al.*, "ShapeNet: An Information-Rich 3D Model Repository," *ArXiv*, vol. abs/1512.03012, 2015.

[3] D. Stutz and A. Geiger, "Learning 3D Shape Completion From Laser Scan Data With Weak Supervision," in *CVPR*, 2018, pp. 1955–1964.

[4] A. Dai, D. Ritchie, *et al.*, "ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans," in *CVPR*, 2018, pp. 4578–4587.

[5] S. Song, F. Yu, *et al.*, "Semantic Scene Completion From a Single Depth Image," in *CVPR*, 2017.

[6] M. Firman, O. Mac Aodha, *et al.*, "Structured Prediction of Unobserved Voxels From a Single Depth Image," in *CVPR*, 2016, pp. 5431–5440.

[7] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-Supervised Sparse-to-Dense: Self-Supervised Depth Completion from LiDAR and Monocular Camera," in *ICRA*, 2019.

[8] J. Uhrig, N. Schneider, *et al.*, "Sparsity Invariant CNNs," in *3DV*, 2017.

[9] M. Garbade, Y.-T. Chen, *et al.*, "Two Stream 3D Semantic Scene Completion," in *CVPR Workshops*, 2019.

[10] Z. Chen and H. Zhang, "Learning Implicit Fields for Generative Shape Modeling," *CVPR*, pp. 5939–5948, 2019. [Online]. Available: http://arxiv.org/abs/1812.02822

[11] L. Mescheder, M. Oechsle, *et al.*, "Occupancy Networks: Learning 3D Reconstruction in Function Space," in *CVPR*, 2019, pp. 4460–4470. [Online]. Available: http://arxiv.org/abs/1812.03828

[12] M. Michalkiewicz, J. K. Pontes, *et al.*, "Deep Level Sets: Implicit Surface Representations for 3D Shape Inference," *ArXiv*, vol. abs/1901.06802, 2019.

[13] J. J. Park, P. Florence, *et al.*, "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation," *CVPR*, pp. 165–174, 2019. [Online]. Available: http://arxiv.org/abs/1901.05103

[14] T.-W. Hui, C. C. Loy, and X. Tang, "Depth Map Super-Resolution by Deep Multi-Scale Guidance," in *ECCV*, 2016.

[15] L. Triess, D. Peter, *et al.*, "CNN-based synthesis of realistic high-resolution LiDAR data," in *IV*, 2019, pp. 1512–1519.

[16] L. Yu, X. Li, *et al.*, "PU-Net: Point Cloud Upsampling Network," in *CVPR*, 2018, pp. 2790–2799.

[17] J. Qiu, Z. Cui, *et al.*, "Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image," in *CVPR*, June 2019.

[18] N. Schneider, L. Schneider, *et al.*, "Semantically guided depth upsampling," in *GCPR*, 2016.

[19] Y. Chen, B. Yang, *et al.*, "Learning Joint 2D-3D Representations for Depth Completion," in *ICCV*, 2019.

[20] W. Van Gansbeke, D. Neven, *et al.*, "Sparse and Noisy LiDAR Completion with RGB Guidance and Uncertainty," in *MVA*, 2019.

[21] Y. Yang, A. Wong, and S. Soatto, "Dense Depth Posterior (DDP) From Single Image and Sparse Range," in *CVPR*, 2019, pp. 3353–3362.

[22] H. Hekmatian, J. Jin, and S. Al-Stouhi, "Conf-Net: Toward High-Confidence Dense 3D Point-Cloud with Error-Map Prediction," 2019. [Online]. Available: http://arxiv.org/abs/1907.10148

[23] N. Chodosh, C. Wang, and S. Lucey, "Deep Convolutional Compressed Sensing for LiDAR Depth Completion," in *ACCV*, 2018, pp. 499–513.

[24] J. Ku, A. Harakeh, and S. L. Waslander, "In Defense of Classical Image Processing: Fast Depth Completion on the CPU," in *2018 15th Conference on Computer and Robot Vision (CRV)*, 2018, pp. 16–22.

[25] N. Silberman, D. Hoiem, *et al.*, "Indoor Segmentation and Support Inference from RGBD Images," in *ECCV*, 2012, pp. 746–760.

[26] L. P. Tchapmi, C. B. Choy, *et al.*, "SEGCloud: Semantic Segmentation of 3D Point Clouds," in *3DV*, 2017.

[27] Y. Liao, S. Donné, and A. Geiger, "Deep marching cubes: Learning explicit surface representations," in *CVPR*, 2018, pp. 2916–2925.

[28] C. Häne, S. Tulsiani, and J. Malik, "Hierarchical Surface Prediction for 3D Object Reconstruction," in *3DV*, 2017, pp. 412–420.

[29] G. Riegler, A. Osman Ulusoy, and A. Geiger, "OctNet: Learning Deep 3D Representations at High Resolutions," in *CVPR*, 2017, pp. 3577–3586.

[30] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree Generating Networks: Efficient Convolutional Architectures for High-Resolution 3D Outputs," in *ICCV*, 2017, pp. 2088–2096.

[31] P.-S. Wang, Y. Liu, *et al.*, "O-cnn: Octree-based convolutional neural networks for 3d shape analysis," *ACM Trans. on Graphics*, vol. 36, no. 4, 2017.

[32] H. Alhaija, S. Mustikovela, *et al.*, "Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes," *IJCV*, 2018.

[33] J. Behley, M. Garbade, *et al.*, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *ICCV*, 2019. [Online]. Available: http://arxiv.org/abs/1904.01416

[34] M. Cordts, M. Omran, *et al.*, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *CVPR*, 2016, pp. 3213–3223.

[35] L.-C. Chen, Y. Zhu, *et al.*, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," in *ECCV*, 2018.

[36] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[37] M. Tatarchenko, J. Park, *et al.*, "Tangent Convolutions for Dense Prediction in 3D," in *CVPR*, June 2018.

[38] B. Wu, A. Wan, *et al.*, "SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud," in *ICRA*, 2018, pp. 1887–1893.

[39] A. H. Lang, S. Vora, *et al.*, "PointPillars: Fast Encoders for Object Detection From Point Clouds," in *CVPR*, 2019, pp. 12 697–12 705.

[40] C. Rist, M. Enzweiler, and D. Gavrila, "Cross-Sensor Deep Domain Adaptation for LiDAR Detection and Segmentation," in *IV*, 2019, pp. 1535–1542.

[41] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," in *CVPR*, 2018, pp. 4490–4499.

[42] H. de Vries, F. Strub, *et al.*, "Modulating early visual processing by language," in *NIPS*, 2017, pp. 6594–6604.