GndNet: Fast Ground Plane Estimation and Point Cloud Segmentation for Autonomous Vehicles

Anshul Paigwar, Özgür Erkent, David Sierra González, Christian Laugier

GndNet: Fast Ground Plane Estimation and Point Cloud Segmentation for Autonomous Vehicles

Anshul Paigwar, Özgür Erkent, David Sierra-Gonzalez, Christian Laugier

***Abstract*— Ground plane estimation and ground point segmentation is a crucial precursor for many applications in robotics and intelligent vehicles like navigable space detection and occupancy grid generation, 3D object detection, point cloud matching for localization and registration for mapping. In this paper, we present GndNet, a novel end-to-end approach that estimates the ground plane elevation information in a grid-based representation and segments the ground points simultaneously in real-time. GndNet uses PointNet and Pillar Feature Encoding network to extract features and regresses ground height for each cell of the grid. We augment the SemanticKITTI dataset to train our network. We demonstrate qualitative and quantitative evaluation of our results for ground elevation estimation and semantic segmentation of point cloud. GndNet establishes a new state-of-the-art, achieves a run-time of 55Hz for ground plane estimation and ground point segmentation.**

I. Introduction

Fully autonomous driving is an important but challenging goal, for which a reliable perception of the local environment is crucial [1]. 3D-LiDARs are popular and widely used sensors in robotics and intelligent vehicles. LiDARs generate high-resolution 3D point clouds of the environment while remaining unaffected by varying illumination. Point clouds are used for several applications including object recognition, navigation, and path planning. Navigation for intelligent vehicles requires accurate detection of the navigable space and classification of obstacles. For path planning and to estimate the risks on the path, occupancy grids are often used [2]. Occupancy grids are 2D spatial maps of the environment around the vehicle which can be constructed by processing the LiDAR point cloud data. Typical steps that precede the occupancy grid generation are the estimation of the ground plane and the segmentation of the ground points [3].

The goal of the ground plane estimation is to find out the height of the ground for each of the grid cells. Due to the sparse nature of the point clouds, this is a challenging task. Furthermore, the occupation of some of the cells by temporary obstacles such as vehicles or pedestrians, lack of points in some of the cells due to occlusions, and unevenness in the surface of the ground are some of the other challenges for estimating the ground height of the grid cells. The benefits of a correct estimation of the ground plane go beyond the computation of occupancy grid maps. For instance, knowing the height of each LiDAR point in relation to the ground can benefit the plethora of LiDAR-based 3D object detection approaches [4], [5].

[1] Univ. Grenoble Alpes, Inria, 38000, Grenoble, France; e-mail: firstname.lastname@inria.fr

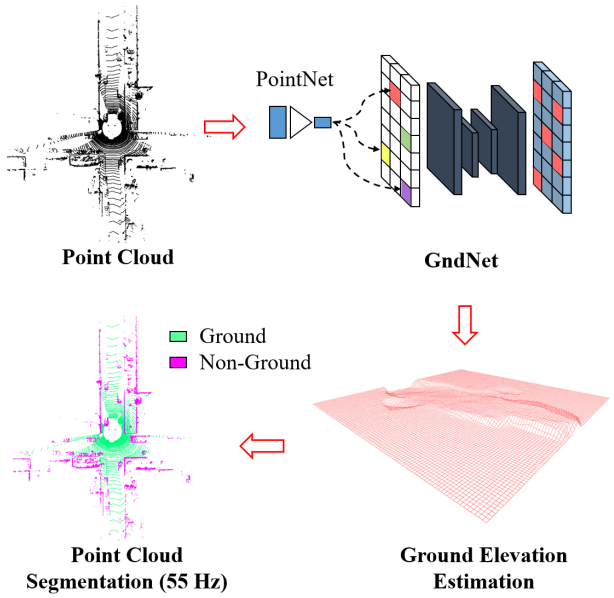

Fig. 1. **Overview of GndNet architecture:** GndNet takes point cloud from 3D LiDAR as input, estimates ground plane elevation, and segments point cloud into the ground and non-ground categories at 55Hz.

The segmentation of the points belonging to the ground is also an important task. If these points can be segmented, other tasks such as object detection and classification or localization and mapping would be improved both in accuracy and speed.

Therefore, we formulate our problem as finding a method that would estimate the ground plane and segment the ground points simultaneously in real-time for an autonomous vehicle setting where the computational efficiency of the method is a critical aspect.

In this work, we present GndNet, a deep learning-based approach for ground plane estimation and point cloud segmentation. GndNet uses PointNet to extract point-wise features [6] and 2D convolutions on the grid to extract spatial features. By leveraging these features, the proposed network learns the appearances, analyzes the scene and estimates the ground elevation in real-time (Figure 1).

One issue with learning-based approaches is the requirement of a large amount of annotated data for training. Currently, no public dataset is available with an annotated ground elevation map. To overcome this difficulty, we propose two approaches: morphological operations and a CRF-based [3] approach to obtain the ground-truth elevation map from the SemanticKITTI dataset [7], [8].

The key contributions of this study can be summarized as follows:

- We present a novel deep neural network architecture called GndNet for real-time ground elevation estimation and point cloud segmentation. The network directly operates on sparse 3D points and is end-to-end trainable.
- We show the network's ability to analyze the scene, learn the appearances, and distinguish the points belonging to the ground and non-ground.
- We demonstrate how to generate ground truth datasets for ground elevation from the SemanticKITTI dataset. We perform real experiments and quantitative comparisons to prove the integrity of our learning-based approach.

II. Related Work

In this section, we first discuss methods related to ground plane estimation and segmentation. Next, we discuss methods that solely focus on ground points segmentation. Finally, we cover deep learning methods for feature extraction from point clouds.

One of the initial methods that appeared for ground estimation was based on elevation maps [9]. This method was used in the DARPA Urban Challenge and is based on projecting 3D points as a 2.5D grid and then using a Min-Max elevation map. However, this approach suffers from large errors in the case of bridges or treetops. Another group of studies relies on a 2D-line-extraction-based fast algorithm [10]. However, these algorithm-based methods have problems with scalability to a large set of cases, for example, curved terrains. Several other methods use the gradient information of the terrain to model the ground plane using a Markov Random Field (MRF) or a Conditional Random Field (CRF) [11], [12], [13], [3]. However, due to limited learning capacity the results in cases of occlusions and sparse points are not satisfactory and the segmentation of ground points require thus additional processing time.

Narksri et al, propose a two-stage method to estimate the ground points and then fit a plane by using RANSAC, which makes the approach slow [14]. Liu et al integrate Gaussian Process Regression (GPR) and Robust Locally Weighted Regression (RLWR) to form a hybrid regression model for the ground plane, which is also not real-time due to the computational complexity of the GPs [15]. In another study, Velas et al use a discriminative method to segment the ground points by using a neural network that performs in real-time [16]. However, the network is capable only of ground point segmentation. A ground plane is not obtained for occluded regions which would be necessary for tasks related to autonomous vehicles.

One of the common points in most of the discussed methods is the use of handcrafted features. They generally fail to scale, are complex to implement, and are computationally expensive. The sparsity, occlusions and the roughness of the terrain are not considered.

Recently, Deep Learning methods has emerged as the prominent alternative for manual feature engineering for learning the representation of point clouds. However, in contrast to images, point clouds lack the detailed texture information. Pointclouds obtained from 3D LiDARs are unordered, sparse and have a variable point density.

Recent work proposes novel types of network architectures, dealing with challenges associated with point cloud processing. Among these, PointNet has shown encouraging results for single object classification and semantic segmentation [6]. PointNet can learn point features from a set of points but its applicability to entire large scale point cloud has not been studied yet. 3D object detection approaches like VoxelNet discretize the point cloud into a voxel grid [4]. PointPillars takes a similar route but uses pillars instead of voxels [5]. PointNet is then applied to the points inside each voxel/pillar to learn voxel/pillar wise features. This voxel/pillar feature encoding network is then used with convolution filters for region proposals and object detection.

Inspired by VoxelNet and Pointpillars we choose to use PointNet together with a pillar feature encoding network as the backbone for our model.

III. GndNet Architecture

GndNet accepts raw point clouds as input and produces a grid-based ground elevation estimation and a point cloud segmentation into two categories (ground and non-ground). It consists of three main stages as shown in Figure 2: (1) Discretization of the point cloud into a 2D grid; (2) Pillar feature encoding network that converts a point cloud to a sparse pseudo image; and (3) 2D convolutional encoder-decoder network to process the pseudo-image and produce a high-level representation and a regression of the ground elevation per cell.

A. Point cloud discretization

To extract features from the point cloud, we first discretize it into an evenly spaced grid in the $x-y$ plane. Unlike VoxelNet [4], which discretizes the point cloud into a 3D voxel map, the PointPillars authors argued that there is no need for binning the z dimension, as it does not affect the accuracy of the detection and is significantly more computationally efficient [5]. This is equivalent to creating a set of pillars P. We denote by l a point in a point cloud with coordinates x, y, z, and reflectance r. Similar to Pointpillars, the points in each pillar are then augmented with $(x_c, y_c, z_c, x_p, y_p)$ where the c subscript denotes distance to the arithmetic mean of all points in the pillar and the p subscript denotes the offset from the pillar (x, y) center. Four-dimensional lidar point l is now augmented to a $D = 9$ dimension. A large set of pillars will be mostly empty due to the sparsity of the point cloud, and the non-empty pillars will, in general, have only a few points in them. We fixed the number of points N per pillar to create a dense tensor of size (D, P, N), where P is the number of non-empty pillars. If a pillar holds more than N points then we randomly sample the points to fit the tensor. Conversely, if a pillar has too few data points then a zero padding is applied to populate the tensor.

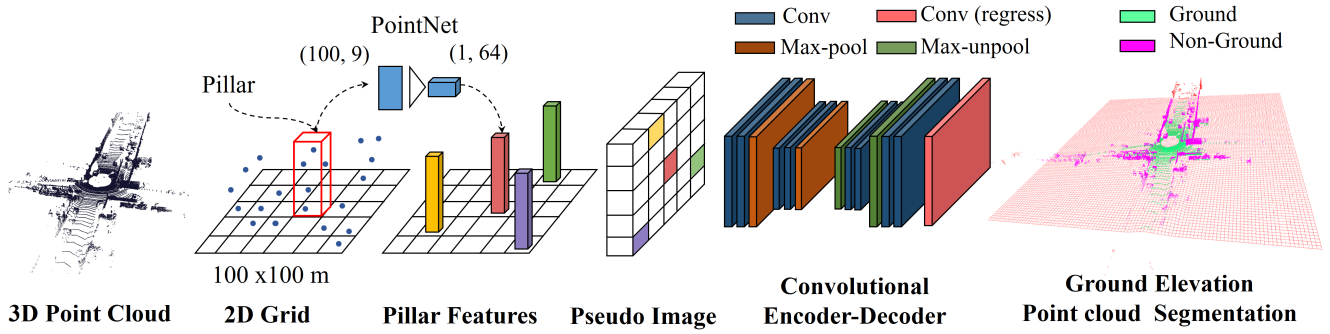

Fig. 2. **GndNet architecture:** We use a pillar feature network with a fully convolutional encoder-decoder network to regress local ground elevation values. The raw point cloud is discretized in a 2D grid forming a set of pillars. PointNet uses points in the pillars to learn pillar wise features that can be scattered back to a 2D pseudo image. A convolutional encoder-decoder network uses the pseudo image to learn spatial features and regresses ground elevation values per cell in the grid. The point cloud is then segmented in the ground and non-ground category using elevation values with a threshold.

B. Pillar Feature Encoding and Pseudo Image

Next, we extract features per non-empty pillar using a simplified version of PointNet. Our simplified PointNet consists of a linear layer that is applied to each point followed by Batchnorm and ReLU to generate a (C, P, N) sized tensor. Then, a max-pooling operation over the channels creates an output tensor of size (C, P). These encoded pillar features are then placed to their original locations in the grid to create a pseudo-image of size (C, H, W) where H and W indicate the height and width of the grid.

C. Encoder-Decoder Network

To extract spatial features we use a convolutional encoder-decoder network in a similar fashion as SegNet [17]. The encoder part of the network consists of 4 convolutional layers. Max-pooling with a 2×2 window and stride 2 (non-overlapping window) is performed after every two convolution layers. The resulting output of the encoder is sub-sampled by a factor of 4. We selected the number of layers such that the ground elevation of a cell is only dependent upon its close neighboring cells. Each encoder layer has a corresponding decoder layer and hence the decoder network has 4 convolution layers and max-unpooling after every two convolution layers. The final decoder output is of the same shape as the input pseudo image (grid), and is fed to 3x3 convolution filter which regresses the ground elevation value for each pixel in the pseudo image (cell in the grid).

D. Point cloud segmentation

We segment the point cloud into two categories as ground or obstacle points. In each cell, the points above threshold T of predicted ground elevation are segmented as obstacles and points below are segmented as ground points. These ground points then can be removed to generate an occupancy grid or to facilitate object detection. Ground elevation itself can be used to find navigable and non-navigable space for path planning.

IV. Training and Experiments

Supervised learning using deep neural networks requires, in general, a large amount of annotated data to train. Absence of annotated ground elevation and point cloud data is the reason learning-based approaches have not been explored for this task. Also because of the lack of annotated data, none of the existing ground detection approaches carry-out exhaustive quantitative evaluation and cross-comparison. Generating annotated ground elevation data is a crucial and important challenge we try to solve in this work.

A. Dataset generation

To generate an annotated ground elevation dataset we use the SemanticKITTI [7] dataset, which is based on the KITTI Vision benchmark and uses sequences from the odometry task [8]. SemanticKITTI provides dense annotations for each scan of sequences 00-10. The dataset contains 28 classes including classes distinguishing non-moving and moving objects (which covers traffic participants), and also functional classes for ground, such as parking areas and sidewalks.

Definition of ground: We define ground elevation as the height in the LiDAR sensor frame of reference at which object boundaries start or objects can be placed. For example, the ground can be any terrain where a vehicle can traverse (does not imply that it is legal to traverse). In the case of non-moving objects like walls and trees, the ground elevation is at the bottom where the actual object boundary starts as shown in Figure 3.

To generate ground elevation dataset from the SemanticKITTI dataset, we first remove all the points that do not belong to the ground, and keep only those belonging to the categories road, sidewalk, parking, other ground, and terrain as shown in Figure 4. We divide the entire environment into a 2D grid of shape $(100, 100)$ with cell resolution of 1m x 1m. We then calculate the ground-truth ground elevation per cell. One approach for

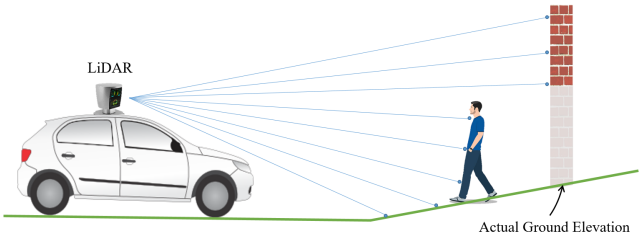

Fig. 3. Definition of the ground plane as defined and utilized for this work.

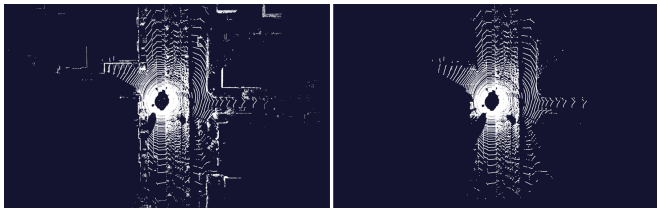

Fig. 4. *Left*: 3D point cloud from SemanticKITTI dataset; *Right*: point cloud with only ground points.

this task would be to align the point clouds and estimate global elevation field for the whole driving sequence. We observe that there are still many cells without the ground points, mainly due to the occlusions. We explore two different methods to tackle this issue:

Morphological method: We average the z values of all the ground points in the corresponding cells and create an elevation map as shown in Figure 5 (d). Many cells belonging to the ground do not have any points in them resulting in the elevation of the cell as zero. Through this work, we aim to achieve a smooth, uniform ground plane elevation even in regions with sparse points and in occluded regions. To fill the holes in the elevation map we use an image processing technique of in-painting. The procedure is as follows:

1) The first step is to create a mask for the cells that need to be inpainted. We create an occupancy map using the ground points as in Figure 5 (a). We use dilation to fill the holes in occupancy map, Figure 5 (b). We subtract occupancy map from dilated occupancy map resulting in the desired mask, Figure 5 (c).

2) Inpainting is the process of reconstructing lost or deteriorated parts of images using the information from neighboring pixels. We use a biharmonic function based inpainting and surface completion method [18] implemented in scikit-image [19] to fill the holes in the elevation map. Inpainting resulted in the far superior interpolation of ground elevation than just averaging over neighboring cells—see Figure 5 (e). Finally, to further smoothen small crests and troughs we apply an average filter over the entire inpainted elevation map, Figure 5 (f).

CRF-based Method: Another more mathematically consistent approach for surface reconstruction is using CRFs. We used a similar CRF model as proposed in [3] to model the ground plane. The CRF based method uses temporal

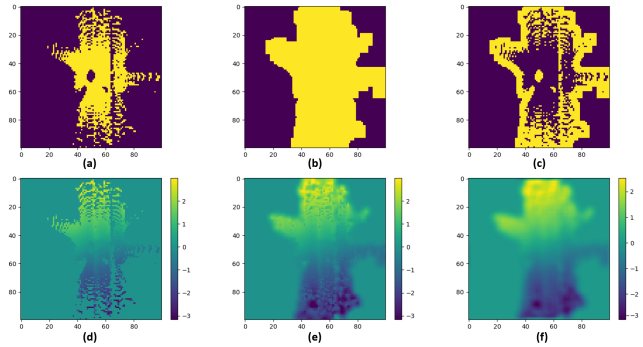

Fig. 5. **Morphological Operations:** (a) Occupancy map generated using only ground points. (b) Dilated occupancy map to fill the void cells. (c) Occupancy map subtracted from dilated occupancy map resulting into a mask for void cells. (d) Elevation map. (e) Inpainted elevation map. (f) Final elevation map after applying average filter. The colors represent elevation of the cells and the side bar shows the scale in meters.

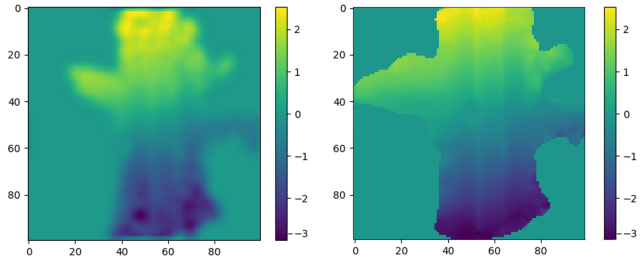

Fig. 6. Qualitative comparison of ground truth elevation map generated using morphological operations vs CRF-method. The colors represent elevation of the cells and the side bar shows the scale in meters.

dependence and the reconstruction resulted in more structurally defined and consistent output than a morphology-based approach. A comparison between outputs from both the method is shown in Figure 6.

Finally, we choose to use the CRF-based method to generate the ground elevation dataset for the training of our network. For each scan, we save the complete point cloud (including ground and non-ground points) and the ground elevation map generated using only ground points. To help our network generalize better on challenging terrains with steeper slopes than those present in the SemanticKITTI dataset, we augment the point cloud by randomly applying 3D rotations in the range of $(-10, 10)$ degrees with respect to the x and y axes. The SemanticKITTI dataset has 11 annotated sequences; we use 7 sequences for the training set and 4 sequences for the validation set. The final dataset consists of 6584 frames for training and 3040 frames for validation.

B. Loss Function

We perform end-to-end training of the full model including pillar feature encoding layer and encoder-decoder module using a combination of regression loss L_{reg} and spatial-smoothing loss L_{smooth}:

$$L(I, \hat{I}) = \alpha L_{reg}(I, \hat{I}) + \beta L_{smooth}(\hat{I}) \qquad (1)$$

where I denotes the ground truth elevation map of the grid with shape (H, W), and $\hat{I}$ denotes the predicted elevation map. The hyper-parameters α and β are used for balancing the two losses. For the regression of elevation values, we use the Huber loss. For spatial smoothness, we minimize the L1 norm of the second-order gradients for the predicted elevation maps (similar to [20]). We penalize the norm of second-order gradients across adjacent cells to encourage not constant but rather smoothly changing elevation values:

$$L_{smooth}(\hat{I}) = \nabla_x^2 \hat{I} + \nabla_x \nabla_y \hat{I} + \nabla_y \nabla_x \hat{I} + \nabla_y^2 \hat{I} \quad (2)$$

where ∇_x and ∇_y are, respectively, the gradients in the x and y direction of the elevation map.

V. Network and Training Details

A. Network

We do not pre-train our networks, all weights were initialized randomly using a uniform distribution. Our simplified PointNet in the pillar feature encoding network has one fully connected layer with C = (9, 64) input-output features. In the encoder-decoder architecture, all the convolutional layers have kernel size 3, and stride and padding 1. Each convolutional layer is followed by batch normalization and element-wise rectified-linear non-linearity (ReLU) except for the last regression layer. Max-pooling and max-unpooling are performed with a 2×2 window and stride 2. The specifics of the encoder-decoder network design with input-output channels for the convolutional layers are: Conv1(64, 128), Conv2(128, 128), Max-pool, Conv3(128, 256), Conv4(256, 256), Max-pool, Max-unpool, Conv5(256, 256), Conv6(256, 128), Max-unpool, Conv7(128, 64), Conv8(64, 64), and regression layer $\text{Conv}_{reg}(64, 1)$.

B. Training Settings

GndNet takes a raw point cloud with approximately 100,000 points as input. We discretize the environment into a 2D grid of size $(100, 100)$ meters. The x, y, z range is $[(-50, 50), (-50, 50), (-4, 4)]$ meters, and we remove all the points outside that range. The maximum number of points per pillar (N) is kept at 100. We keep the cell resolution of 1m x 1m as it provides us better trade-off between accuracy and run-time. Compared to the PointPillars, which keeps the cell resolution of 0.2 m is very small our task as it results in very few ground points per cell, and most of the cells being empty. The hyper-parameters α and β were set to 0.9 and 0.1 respectively. We use the stochastic gradient descent (SGD) optimizer to train the model with a momentum of 0.9, weight decay of 0.0005 and batch size 2. The learning rate is kept 0.01 for the first 40 epochs and then lowered it to 0.001 for further epochs. We observed that the network converges in roughly 120 epochs. Training on our custom ground elevation dataset takes 6 to 8 hours to converge using an Nvidia GTX 1080 GPU. We use the PyTorch [21] machine learning framework for the development of our model.

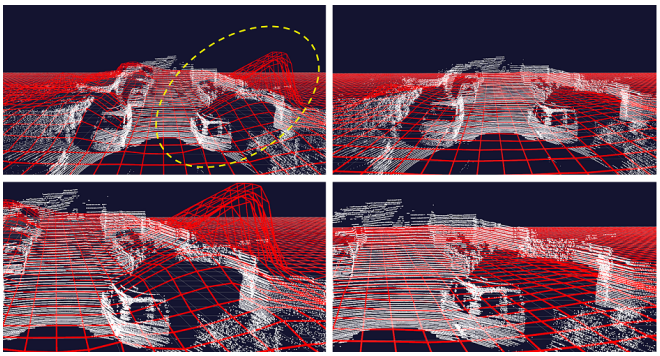

Fig. 7. Qualitative comparison of ground elevation estimation in case of occluded areas (yellow). (*Left*) Output from CRF-method [3]; (*Right*) Output from the proposed method GndNet.

VI. Results

Real-time accurate ground plane estimation and semantic segmentation of point clouds is a challenging task. We compare the proposed method with several top-performing approaches, including a CRF-based method [3], Hybrid-regression [15] and a CNN-method [16]. At the time of writing this work, no ground elevation dataset was available publicly. Most previous approaches only reported qualitative results, while others evaluated their approach on small custom datasets, providing no common basis for quantitative comparison.

In this work, we first deal with the challenge of establishing a common basis for a quantitative evaluation of the ground estimation method. We use the publicly available SemanticKITTI dataset to generate a ground elevation dataset as described in subsection IV-A. We perform a qualitative and quantitative evaluation on this dataset and use the CRF-based method [3] as our baseline. For ease of comparison, we follow the same color schematic and representation of ground elevation as in [3]. We perform two different quantitative evaluations to show the effectiveness of our approach:

A. Quantitative Evaluation

Ground plane estimation: To evaluate the ground plane estimation we use the Root Mean Squared Error (RMSE) metric. We calculate the RMSE for all the cells that contain ground points. We use the elevation map in Figure 5 (d) as ground truth for elevation values. We only qualitatively evaluate the cells that do not have ground points as only interpolated elevation values are available for these regions. Ground elevation estimation results for GndNet are shown in Table I.

TABLE I
Evaluation of Ground Plane Estimation

Method	Dataset	Frames	RMSE (m)
Hybrid-reg [15]	custom	10	0.182
CRF-method[3]	SemKITTI	3040	0.201
GndNet	SemKITTI	3040	0.195

Segmentation of ground points: To evaluate the segmentation of the point clouds we use the mean Jaccard or

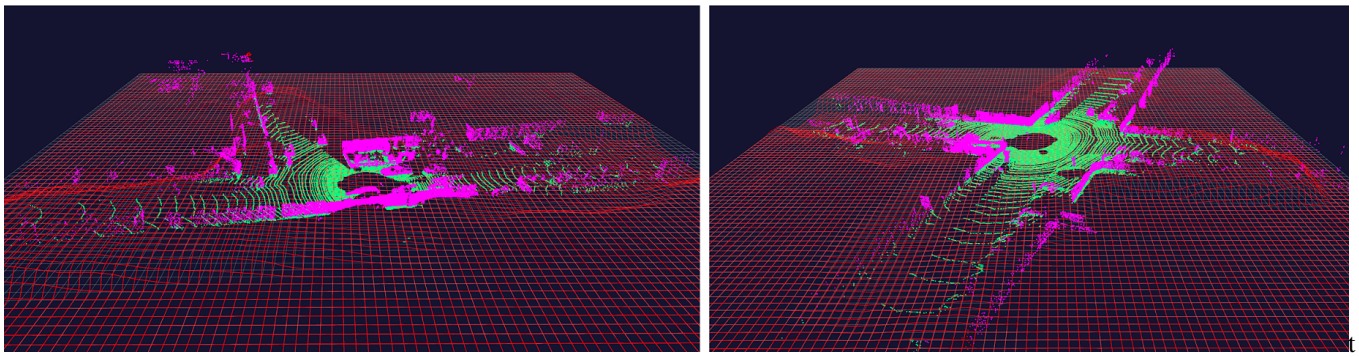

Fig. 8. **Ground elevation estimation by GndNet**: Two diverse scenes with uphill and downhill slopes. GndNet outputs smooth and uniform ground elevation even in the regions with very sparse points up to the distance of 50m. GndNet equally perform well in occluded regions as the network not only uses local features for regressing elevation values but also the spatial features from the points in nearby areas.

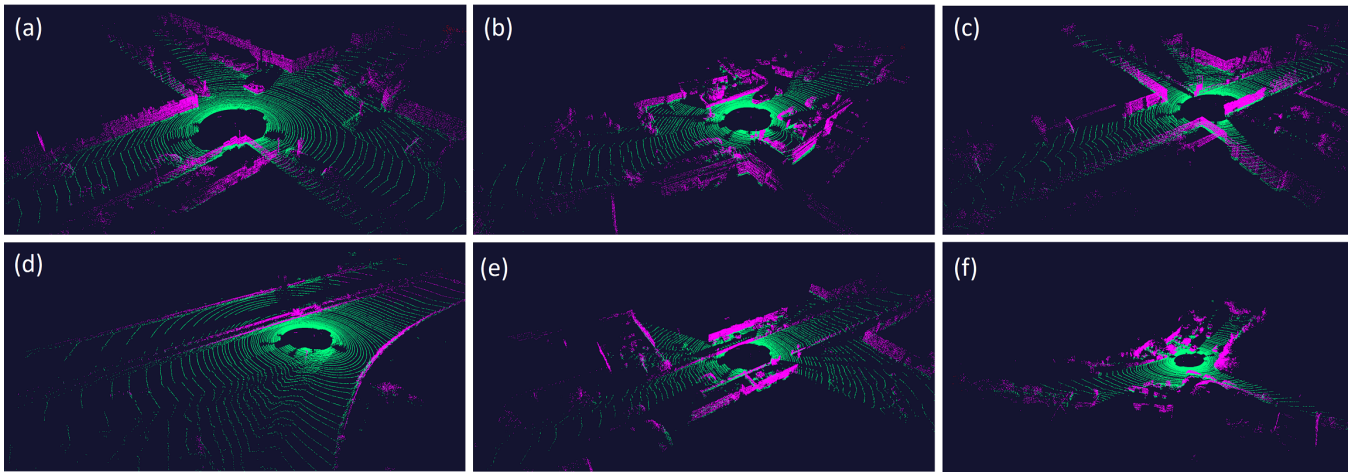

Fig. 9. **Point cloud segmentation using GndNet**: Diverse scenes with crossroads, urban environment, slopes, vehicles, and pedestrians. GndNet learns to distinguish points that are relevant for ground elevation estimation. We segment point cloud in two categories, the points below the estimated elevation values as ground points, those above as non-ground points. The first column shows failure cases with inconsistencies with ground segmentation, some points on the car in (a) are wrongly segmented as the ground while in (d) some road points are segmented as non-ground.

intersection-over-union (mIoU) over two classes: ground and non-ground.

$$mIoU = \frac{1}{C}\sum_{c=1}^{C}\frac{TP_c}{TP_c + FP_c + FN_c} \tag{3}$$

where, TP_c, FP_c and FN_c correspond to the number of true positive, false positive, and false-negative predictions for class c, and C is the number of classes. We also provide precision and recall value and compare those with other approaches in Table II. GndNet shows comparable results with other approaches. Hybrid-reg [15] and CNN-method [16] report very high precision but their approaches are evaluated on a limited set of data.

Inference: The GndNet is a lightweight network and achieves runtime performance of 55 Hz compared to CRF-method [3] of 9.8 Hz for processing $\approx 100,000$ points in the grid size of 100x100m. The Hybrid-reg [15] approach is not real-time, while the CNN-method [16] reports a runtime of 140Hz but only performs the segmentation task (the ground elevation is not estimated). Detailed analysis of the inference time for GndNet is given in Table III. Note that the pre-processing and point segmentation task is performed on CPU using Numpy and Numba which could be further accelerated using GPU.

TABLE II

EVALUATION OF POINT CLOUD SEGMENTATION

Method	**Dataset**	**Frames**	**Prec**	**Recall**	**mIoU**
Hybrid-reg [15]	custom	10	**0.98**	-	-
CNN-method [16]	custom	252	0.929	0.993	-
CRF-method [3]	SemKITTI	**3040**	0.801	0.993	0.782
GndNet	SemKITTI	**3040**	0.841	0.993	0.836

TABLE III

ANALYSIS OF COMPUTATION TIME REQUIRED BY GNDNET

Task	**Time**	**Device**
Pre - processing	7.59 ms	CPU
Model forward pass	6.57 ms	GPU
Point segmentation	3.88 ms	CPU
Total	17.98 ms (55.61 Hz)	

B. Qualitative Evaluation

The illustrations in Figure 8 and Figure 9 show the ground elevation estimation and point cloud segmentation by GndNet in diverse and challenging scenarios. The main advantage of GndNet is in occluded regions with no data points and regions with sparse data. Figure 7 depicts a scenario where a vehicle in front occludes a section of the ground and part of a wall. The competing CRF-based method uses the lowest points in the cells to estimate ground elevation; for the cells that do not have any points, the ground elevation values are interpolated using values from the neighboring cells. As in Figure 7, the CRF-method incorrectly estimates ground elevation as a steep slope starting from the bottom of the vehicle to the visible part of the wall and continues further. GndNet rather learns the appearances, analyzes the scene to accurately estimate ground elevation.

VII. Conclusions

In this paper, we tackled the challenging problem of ground plane estimation and point cloud segmentation for intelligent vehicles. We presented a deep learning-based approach called GndNet, which uses PointNet and Pillar Feature encoding networks to learn point features and output ground elevation in a grid-based representation. We provide a qualitative and quantitative evaluation of our approach on a large dataset and compare our results with other competing approaches. GndNet achieves comparable results in terms of the accuracy and establishes a new state-of-the-art in terms of the run-time of 55Hz for ground plane estimation and ground points segmentation.

Currently, for training, we use a dataset derived from the SemanticKITTI dataset using a CRF-based method. This derived dataset can be inconsistent in certain scenarios and using a carefully annotated dataset by hand can improve the accuracy of the proposed network. Also, our proposed network uses only instantaneous data and spatial features. In the future, we plan to use temporal information and features collected from previous frames; this could potentially improve the ground elevation results in the cases of occluded areas and regions with sparse points. Another line of future work could be directly segmenting point cloud as a predicted output by the network rather than as a post-processing step.

Acknowledgment

This work was conducted at Inria, team Chroma. The authors would like to thank all the team members for their constant support on this research work. This work has been conducted within the scope of ES3CAP (Embedded Smart Safe Secure Computing Autonomous Platform) project.

References

[1] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, *et al.*, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727–774, 2008.

[2] L. Rummelhard, A. Nègre, and C. Laugier, "Conditional monte carlo dense occupancy tracker," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Sept 2015, pp. 2485–2490.

[3] L. Rummelhard, A. Paigwar, A. Nègre, and C. Laugier, "Ground estimation and point cloud segmentation using spatiotemporal conditional random field," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1105–1110.

[4] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[5] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[6] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, vol. 1, no. 2, p. 4, 2017.

[7] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[8] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.

[9] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[10] M. Himmelsbach, F. v. Hundelshausen, and H. J. Wuensche, "Fast segmentation of 3d point clouds for ground vehicles," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, June 2010, pp. 560–565.

[11] C. Guo, W. Sato, L. Han, S. Mita, and D. McAllester, "Graph-based 2d road representation of 3d point clouds for intelligent vehicles," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, June 2011, pp. 715–721.

[12] J. Byun, K.-i. Na, B.-s. Seo, and M. Roh, *Drivable Road Detection with 3D Point Clouds Based on the MRF for Intelligent Vehicle*. Cham: Springer International Publishing, 2015, pp. 49–60. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-07488-7_4

[13] M. Zhang, D. D. Morris, and R. Fu, "Ground segmentation based on loopy belief propagation for sparse 3d point clouds," in *3D Vision (3DV), 2015 International Conference on*, Oct 2015, pp. 615–622.

[14] P. Narksri, E. Takeuchi, Y. Ninomiya, Y. Morales, N. Akai, and N. Kawaguchi, "A slope-robust cascaded ground segmentation in 3d point cloud for autonomous vehicles," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 497–504.

[15] K. Liu, W. Wang, R. Tharmarasa, J. Wang, and Y. Zuo, "Ground surface filtering of 3d point clouds based on hybrid regression technique," *IEEE Access*, vol. 7, pp. 23 270–23 284, 2019.

[16] M. Velas, M. Spanel, M. Hradis, and A. Herout, "Cnn for very fast ground segmentation in velodyne lidar data," in *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2018, pp. 97–103.

[17] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[18] S. Damelin and N. Hoang, "On surface completion and image inpainting by biharmonic functions: Numerical aspects," *International Journal of Mathematics and Mathematical Sciences*, vol. 2018, 2018.

[19] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 6 2014. [Online]. Available: https://doi.org/10.7717/peerj.453

[20] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1851–1858.

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.