General Panoptics: Combining Semantic Segmentation and Classical Methods for a Fast LiDAR Panoptic Segmentation

Lukas Beer and Hans-Joachim Wünsche*

Abstract: Detailed knowledge about the environment is a prerequisite for autonomous driving. One main question in this context is: Which objects are around me? The answer to this question contains two main parts: getting semantic and advanced geometric information. Thus, we need classified, individual instances with a unique ID, together with further information of the surrounding, such as a classification of the ground. This combination is generally known as panoptic segmentation. In this paper, we present a novel general approach for this task. Instead of focusing on an end-to-end network or a combination of object detection and semantic segmentation, our method combines semantic segmentation with a non-learning-based clustering. Both information sources are combined using a Neighborhood-Related Activation Function (NeRAF). This allows a general panoptic segmentation of everything in the surrounding - no matter of its class. We show that the proposed method can run with up to 45 fps on a mobile computer.

Keywords: Clustering, LiDAR, Panoptic Segmentation, Semantic Segmentation

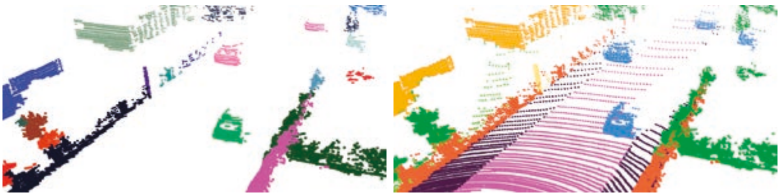
1 Introduction

Since [1] recently introduced the task of panoptic segmentation (PS), it got more and more into focus of current research. Originally defined for images, PS combines semantic segmentation (assign a class to each pixel) and instance segmentation (detect and segment each instance). PS splits classes in two categories: countable objects (*things*) and amorphous regions (*stuff*). Not only limited to images, this task is also a hot topic with 3D point clouds. As a result, we receive individual, classified LiDAR instances.

These instances have several applications: besides the awareness of other traffic participants for motion planning, many of the current localization and SLAM algorithm need semantic landmarks for a precise positioning [2–4]. Nevertheless, most of the *thing* classes are considered to be traffic participants in recent works. Static objects such as trees, bushes, buildings and poles are ignored [5; 6] or only non-natural static objects (buildings, poles, signs) are seen as *things* [7]. To the best of our knowledge, static but natural objects, such as trunks, are not present in so far published LiDAR PS datasets.

In this work, we want to tackle this issue. Contrary to many other approaches [5; 6; 8–11], our method does not fully rely on deep learning. Instead, we combine the non-learning

*This research paper is funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr [project MORE – Munich Mobility Research Campus].
The authors are with the Institute for Autonomous Systems Technology (TAS), Department of Aerospace Engineering, Universität der Bundeswehr München, Neubiberg, Germany. `Lukas.Beer@unibw.de`



(a) Instance output, each instance has a different color.

(b) Semantic output, each class has a different color.

Figure 1: Result of our approach: LiDAR scan from the KITTI dataset with semantic information (1a) and instance information (1b). A video of our approach can be seen on `https://www.mucar3.de/fas2022-general-panoptics`.

based clustering approach from [12] with the semantic segmentation network from [13]. Therefore, we do only need training data for the semantic segmentation.

Both parts are combined using a novel Neighborhood-Related Activation Function (NeRAF) in combination with a consensus-driven fusion as proposed in [8]. The activation function takes instance ID changes, label changes and the depth difference of two neighboring points into account and decides whether they belong to the same semantically or geometrically consistent object or not. In combination with a look-up table for defining the relations, we can simply tune the weights of each component in the activation function, depending on the classes of two neighboring points. With the help of NeRAF, we receive separated instances. For classifying them, we perform the consensus-driven fusion. The output of our approach can be seen in Fig. 1. Even geometrically close objects like fences and bushes can be split robustly.

Due to the high efficiency of all three parts (clustering, semantic segmentation and fusion) our approach runs with up to 45 fps on a consumer graphics card (NVIDIA GeForce GTX 1060).

Compared to previous approaches, the main advantages of this method are the generalized PS, which does not need dedicated PS training data together with the high efficiency while retaining adequate accuracy.

After we introduced PS in Section 1, we start in Section 2 with a short summary of related work in the field of PS. In Section 3 we explain our approach in detail. A quantitative and a qualitative evaluation in Section 4 is followed by a conclusion in Section 5.

2 Related Work

One of the first main contributions to LiDAR PS was [5]: the authors present a dataset for PS in addition to two baseline methods. Each of their approaches uses a state-of-the-art semantic segmentation (KPConv [14] and RangeNet++ [15]) in combination with a state-of-the-art object detector (PointPillar [16]). Having one detection and one segmentation part, they fuse both information afterward. All points inside one bounding box of the object detector are seen as one instance. Using oriented bounding boxes minimizes the error of misclassifying nearby objects. For detecting *things*, they need one object-detector per class.

The Dynamic Shifting Network (DS-Net) [8] combines semantic segmentation with a learning-based clustering. Grid-level features are extracted with the help of cylinder convolutions. Based on those features, two branches perform different tasks: the semantic branch connects Multi-Layer Perceptrons to the cylinder convolution to perform semantic segmentation, while the instance branch uses center regressions for preparing the *things'* points for further clustering. A so-called dynamic shifting (DS) shifts the regressed center to the correct cluster center. In the end, a consensus-driven fusion is applied to merge instances with the semantic segmentation. This consensus-driven fusion is a majority voting – for each predicted instance, the most appeared semantic label of its points is used for the whole instance.

In contrast to proposal-based approaches, the Panoptic-PolarNet of [11] is based on a single network. This deep network consists of four main components: a so-called Polar BEV encoder that encodes the raw point cloud into a 2D representation, a shared encoder-decoder network, two heads for semantic and instance segmentation and finally a fusion step.

One of the fastest PS algorithms is Panoster [9]. The authors use an end-to-end network that consists of a shared encoder and two decoupled, symmetric decoders - one for the instance segmentation and one for the semantic segmentation. They fuse the semantic and the instance information using a mask. Their approach is highly efficient and can run with 58 fps.

3 Proposed Approach

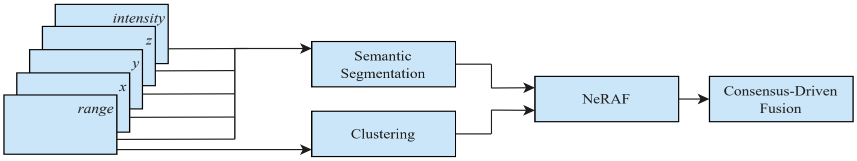


Figure 2: Overview of our approach. With an image-projection as input, a clustering and a semantic segmentation are performed. Afterwards, the activation function generates object instances which are classified by the consensus-driven fusion.

Assuming already preprocessed data, the current approach basically consists of three parts:

1. Semantic Segmentation
2. Clustering
3. Label Fusion

The label fusion itself consists again of two parts: generating instances with the NeRAF and classifying those instances using the consensus-driven fusion. The workflow of our approach can be seen in Fig. 2.

3.1 Preprocessing

Having a 3D point cloud as input, we need to preprocess the incoming data first. Thus, all 3D points (x, y, z) with their euclidean norm $d = \sqrt{x^2 + y^2 + z^2}$ are projected to a set of 2D image coordinates (u, v):

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} W \left[\frac{1}{2}\left[1 - \arctan(y, x)\pi^{-1}\right]\right] \\ H \left[1 - \frac{\left(\arcsin\left(\frac{z}{d}\right) + \mathrm{fov}_{\mathrm{down}}\right)}{\mathrm{fov}_{\mathrm{up}} + \mathrm{fov}_{\mathrm{down}}}\right] \end{pmatrix}. \tag{1}$$

H and W represent the height and the width of the resulting image. $\mathrm{fov}_{\mathrm{up}}$ and $\mathrm{fov}_{\mathrm{down}}$ represent the field of view of the sensor. We can see the resulting output as an image with five channels: $\{d, x, y, z, \mathit{intensity}\}$. If two 3D points lie on the same image coordinate, we choose the one with the smaller euclidean norm (range).

3.2 Semantic Segmentation

Generally, any semantic segmentation could be used for our approach. Nevertheless, we focus on the capability of a high-speed PS. That is why we choose the more efficient variant (”tiny”) of the 3D-MiniNet [13] as the segmentation part of our PS approach. We do not perform the K-Nearest-Neighbors post-processing. Further, we keep the proposed settings for training and data augmentation. For a detailed description of the 3D-MiniNet, we refer to [13].

3.3 Clustering

Due to the focus on efficiency, we choose the clustering algorithm from [12]. This approach takes a range image as input and computes instances based on an angle criterion. Therefore, we can use the same range image as in our semantic segmentation part to compute separated instances. We do not need to perform any other preprocessing, due to the shared image projection. Contrary to object detection algorithms which are based on deep learning and run on the GPU, this algorithm runs efficiently on the CPU. While the GPU is busy performing the semantic segmentation, the clusters can be calculated on the CPU. For further information about the clustering, we refer to [12].

3.4 Label Fusion

The core of our approach is the NeRAF. Neither the semantic segmentation nor the clustering result in 3D points yet. So far, the outcome of each part is a matrix which contains class IDs and instance IDs. These two matrices share the same image projection, and thus also the underlying 3D points. We combine these matrices using the NeRAF: In combination with a breadth-first-search (BFS), such as it is used in the labeling part of [12], we iterate through the matrices and compare the range, the class and the instance ID of two pixels. Further, we ignore ground points – we fully trust on the semantic segmentation for classifying ground. We start with a new instance, and NeRAF decides if two neighboring pixels belong to the same instance. Having two neighboring pixels p_x, p_{x+1} with the classes c_x, c_{x+1}, the instance IDs i_x, i_{x+1} and the ranges d_x, d_{x+1}, we define the varying of the class as v_c, the varying of the instance IDs as v_i and the difference between two depths as δd. For improving the robustness, we do not compare the direct classes c_x and c_{x+1}. Instead we compare the predicted class of the current instance c_i with c_{x+1}. We define NeRAF as

$$\text{NeRAF}(p_x, p_{x+1}) = \frac{w_c v_c + w_i v_i + w_\delta (1 - \frac{min(|\delta d|, \delta d_{max})}{\delta d_{max}})}{w_c + w_i + w_\delta} \tag{2}$$

with w_c, w_i, w_δ as the three weights for v_c, v_i and δd. v_c and v_i have two possible states: 0 if there is a class / instance ID change and 1 if they are equal.

NeRAF results in a value between 0 and 1. We set the decision boundary to 0.5. If the output is greater than that, we add pixel p_{x+1} to the instance. As it can be seen in Eq. (2), we use weights to control each component of NeRAF. Therefore, we can add common a-priori knowledge about possible neighborhood relations. We define three set of weights (w_c, w_i, w_δ):
Set 1. The main focus lies on v_i and δd.
Set 2. This is the counterpart to set 1: v_c is dominant.
Set 3. Here, we set equal weights. Due to the close relation between δd and v_i, we decrease w_δ.

Now, those three sets can handle several different scenarios, depending on the prior knowledge. The scenarios are:

1. very close, but different classes,
2. same class,
3. normal distance between objects.

As an example: It is common sense, that vegetation and trunks are very close or even overlapping, and thus would fall into scenario 1. Contrary to the pure clustering, the semantic segmentation can handle the separation of those two classes. That would be an ideal use case for set 2. Having two neighboring pixel with the same class, we focus v_i and δd (set 1). Neighborhood relations of objects which should not be too close to each other (e.g. vegetation and cars, buildings and trunks) will be handled in set 3. The scenarios are stored in a look-up table. Each combination of classes receives one integer value, which represents the corresponding scenario. Depending on the scenario, NeRAF can switch between the three different weight sets.

Nevertheless, NeRAF just helps to decide if two neighboring points belong to the same object. The class of those instances is still ambiguous: Each point of each instance can have a different class. As proposed in [8], we use a consensus-driven fusion module. This simple fusion strategy is a majority vote: the whole new instance gets that class ID, which occurs most. As a result, we receive a set of pixels of the projected objects which are either geometrically and / or semantically separated.

3.5 Backprojection

The backprojection from images to point clouds is kept simple. For each point in the 3D point cloud, we get the information on which pixel the point should be projected. Due to the discretization, two or more points might share one pixel. In Section 3.1, we projected only the closest points. Thus, for receiving the class and instance information per point, we add a distance threshold. If the range differs more than 30 cm between the current point and the depth on its position on the range image, the class and the instance ID of this point remain undefined.

4 Experiments

We evaluate our approach using different settings to observe the dependencies between our PS and the semantic input:

- $512i$: Projection size: 64×512 px.
- 2048: Projection size: 64×2048 px.
- $2048f$: Projection size: 64×2048 px, "full" 3D-MiniNet.

Note that $2048f$ refers to the non-tiny version of the 3D-MiniNet.

For each setting, we train a network for 500 epochs on the KITTI data for semantic segmentation [17; 18]. As proposed in [17], sequences 00-10 were used for training, except sequence 08 which is used as validation set. For testing and evaluating, a notebook was used with an NVIDIA GTX 1060 as GPU and an i7-8750H CPU with 2.20 GHz. The clustering and the semantic segmentation run in parallel, everything else runs on a single core. We implement our approach in C++ inside the Robot Operating System (ROS) framework [19] and use the TensorRT [20] library to speed up the inference.

4.1 Evaluation Metric

As proposed in [1], we evaluate our approach using the Panoptic Quality (PQ), the Segmentation Qualitiy (SQ) and the Recognition Quality (RQ) which are calculated across all classes. The scores are between 0 and 100%, whereby 100% denotes a perfect PS. We further evaluate those three metrics separately on *things* PQ^{th}, SQ^{th}, RQ^{th} and *stuff* PQ^{st}, SQ^{st}, RQ^{st}. Moreover, $PQ^{\dagger}$ defines the PQ metric, while swapping the PQ of *stuff* classes with its IoU. Even though our approach itself does not distinguish between *stuff* and *things* by its class, we keep the *stuff*/*things* setting from [5] due to the lack of static or natural *things* in the evaluation data. Setting the estimated instance ID of those classes would cause a misleading result. Therefore, we follow [5] and set the instance ID of the *stuff* classes (for the evaluation) to zero.

4.2 Quantitative Results

4.2.1 Panoptic Quality

We evaluate our approach using the SemanticKITTI Benchmark for PS. We compare our results to the results of the two baseline methods in [17], LPSAD [10], the Multi-Object Panoptic Tracking of [21], Panoster [9], the Panoptic-PolarNet [11] and the DS-Net [8]. The results can be found in Table 1.

Generally, our approach receives a PQ score between 24.5 and 39.5%, depending on the network. Therefore, our method reaches similar results to [10] and [15] in combination with [16]. Nevertheless, the accuracy of our approach lies under most of the state-of-the-art methods. One main reason is the trade-off between performance and quality: The better the overall performance of the semantic segmentation (e.g. $2048f$), the better the result of the PS. Nevertheless, this also leads to an extended computation time. Moreover, our approach is the only general approach: while others do need the strict definition of *stuff* and *things* and dedicated training data for the instance segmentation, our algorithm results in PS of each object – no matter of its class.

Table 1: Comparison of LiDAR panoptic segmentation performance on the SemanticKITTI test set. All scores are in [%]. Note different GPUs:
†NVIDIA RTX 2080ti, *NVIDIA Quadro P6000, +NVIDIA GTX 1060, ♯NVIDIA GTX 1080, ᵒunknown

Method	PQ	PQ†	SQ	RQ	PQ^{Th}	SQ^{Th}	RQ^{Th}	PQ^{St}	SQ^{St}	RQ^{St}	mIoU	t [
R.N. [15] + P.P [16]†	37.1	45.9	75.9	47.0	20.2	75.2	25.2	49.3	76.5	62.8	52.4	4
KPC [14] + P.P. [16]†	44.5	52.5	80.0	54.4	32.7	81.5	38.7	53.1	**79.0**	65.9	58.8	5
LPSAD [10]*	38.0	47.0	76.5	48.2	25.6	76.8	31.8	47.1	76.2	60.1	50.9	8
Panoster [9]♯	52.7	59.9	**80.7**	64.1	49.4	**83.3**	58.5	**55.1**	78.8	68.2	59.9	**1**
Panoptic-PolarNet [11]+	54.1	60.7	65.0	**81.4**	53.3	60.6	**87.2**	54.8	68.1	**77.2**	59.5	2
DS-Net [8]ᵒ	**57.7**	**63.4**	68.0	77.6	**61.8**	68.8	78.2	54.8	67.3	77.1	**63.5**	5
ours (512)+	27.0	36.9	72.0	36.5	15.9	74.3	20.9	35.0	70.3	47.8	40.8	2
ours (2048)+	34.3	42.8	74.1	45.4	23.4	76.7	29.8	42.2	75.1	56.7	45.1	1
ours (2048*f*)+	39.6	47.6	74.8	52.0	30.9	78.4	38.5	45.9	72.1	61.8	51.1	1

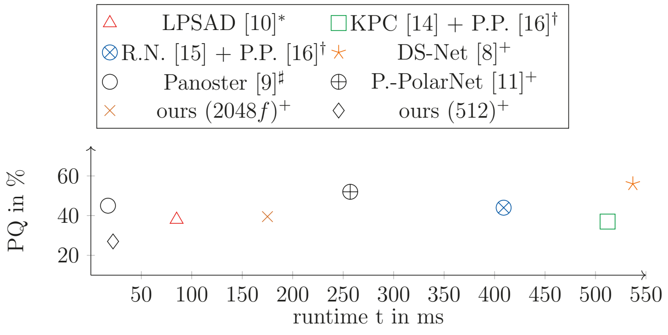
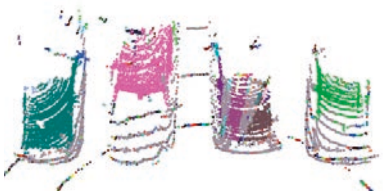


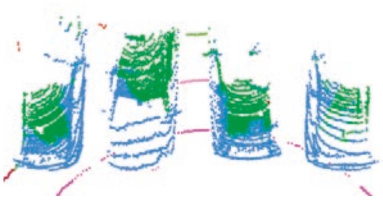
Figure 3: Runtime of several approaches. As already in Table 1, note the different GPUs.

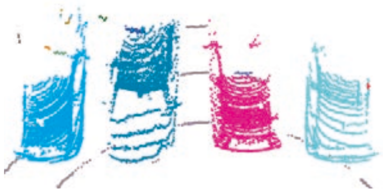
4.2.2 Runtime Analysis

The runtime of the different approaches can be seen in Table 1. In our calculation, we exclude the spherical projection and the backprojection: Everything else, including further preprocessing, copying, and moving data, is included.

Generally, our approach reaches a mean runtime of 22 to 175 ms, depending on the setting. As one would expect, a four times smaller input (64×512 vs. 64×2048) results in a roughly four times lower computation time. Comparing our approach with others is rather difficult due to different hardware or non-public code. Nevertheless, in Fig. 3 we can see the runtime and PQ of several methods and their hardware used. Even though a low-level consumer graphics card was used, we can clearly see that our approach is up to 26 times faster than other approaches. Nevertheless, we can not reach the 17 ms from [9], conducted on an NVIDIA GTX 1080. Comparing the full point cloud (64×2048), the approach from [10] beats our runtime, too. According to the authors, they used an NVIDIA Quadro P6000.
Even though we use a highly efficient network, the segmentation still has the highest impact on the computation time: With a full range-image (64×2048), the fusion step takes 15 ms. Copying the data in combination with further preprocessing takes 40 ms. The clustering does not impact the computation time because it is always faster than the semantic segmentation. 42 to 69 % of the computation time is used for the inference.
In summary, even with a low-level graphics card, our approach runs faster than most of the other approaches, despite their superior hardware.


(a) Cluster input. Grey points denote ground.


(b) Semantic input.


(c) Cluster output. Grey points denote ground.

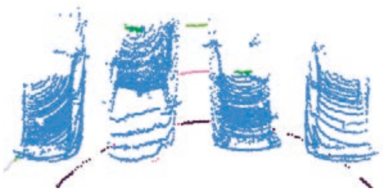

(d) Semantic output.

Figure 4: In- and outputs of our method. While Fig. 4a shows the cluster input and Fig. 4b shows the semantic input, Fig. 4c shows the resulting cluster and Fig. 4d shows the resulting class.

4.3 Qualitative Results

Even though other purely deep learning based methods perform well in their evaluation sequences, our approach has one major advantage: through the combination of learning and non-learning based methods, it takes the benefits of both worlds. Hence, the output of our PS is generally stable in unknown areas and different sensors. Fig. 4 shows an example using a Velodyne VLS-128 LiDAR. Further, we added extra noise to the point cloud in order to enlarge the error. Even though neither the semantic segmentation nor the clustering result in a good solution, the outcome of the combination becomes more coherent.

5 Conclusion

In this work, we show a general approach for PS. Contrary to other methods, it is not fully based on deep learning and combines semantic segmentation with clustering using a neighborhood-related activation function (NeRAF). Hence, it generates classified instances, no matter of its class. Depending on the incoming semantic segmentation, our approach can reach the accuracy of current state-of-the-art. Moreover, every single part of our approach is highly efficient. This makes the PS run with up to 45 fps on a consumer graphics card. Future work should keep a focus on the combination of non-learning based clustering and semantic segmentation for PS and evaluate different methods for fusing both parts. Further, NeRAF should be evaluated further, using advanced networks and different clustering methods.

References

[1] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, “Panoptic Segmentation,” in *Proc. IEEE Conf. Comput. Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, Jan. 2019.

[2] B. Cao, C.-N. Ritter, D. Göhring, and R. Rojas, “Accurate Localization of Autonomous Vehicles Based on Pattern Matching and Graph-Based Optimization in Urban Environments,” in *Proc. IEEE Intelligent Transportation Syst. Conf. (ITSC)*, 2020.

[3] A. Schaefer, D. Buescher, J. Vertens, L. Luft, and W. Burgard, “Long-Term Urban Vehicle Localization Using Pole Landmarks Extracted from 3-D Lidar Scans,” in *European Conference on Mobile Robots (ECMR)*, Prague, Czech Republic, Sep. 2019.

[4] P. Burger, B. Naujoks, and H.-J. Wuensche, “Map-Aware SLAM with Sparse Map Features,” in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, Macau, China, 2019.

[5] J. Behley, A. Milioto, and C. Stachniss, “A Benchmark for LiDAR-based Panoptic Segmentation based on KITTI,” *arXiv preprint arXiv:2003.02371*, 2020.

[6] K. Sirohi, R. Mohan, D. Büscher, W. Burgard, and A. Valada, “EfficientLPS: Efficient LiDAR Panoptic Segmentation,” *arXiv preprint arXiv:2102.08009*, 2021.

[7] J. Xie, M. Kiefel, M.-T. Sun, and A. Geiger, "Semantic Instance Annotation of Street Scenes by 3D to 2D Label Transfer," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognition (CVPR)*, 2016.

[8] F. Hong, H. Zhou, X. Zhu, H. Li, and Z. Liu, "LiDAR-Based Panoptic Segmentation via Dynamic Shifting Network," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognition (CVPR)*, Jun. 2021.

[9] S. Gasperini, M.-A. N. Mahani, A. Marcos-Ramiro, N. Navab, and F. Tombari, "Panoster: End-to-end Panoptic Segmentation of LiDAR Point Clouds," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, 2021.

[10] A. Milioto, J. Behley, C. McCool, and C. Stachniss, "LiDAR Panoptic Segmentation for Autonomous Driving," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, 2020.

[11] Z. Zhou, Y. Zhang, and H. Foroosh, "Panoptic-PolarNet: Proposal-Free LiDAR Point Cloud Panoptic Segmentation," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognition (CVPR)*, Jun. 2021.

[12] I. Bogoslavskyi and C. Stachniss, "Fast Range Image-Based Segmentation of Sparse 3D Laser Scans for Online Operation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, 2016.

[13] I. Alonso, L. Riazuelo, L. Montesano, and A. C. Murillo, "3D-MiniNet: Learning a 2D Representation from Point Clouds for Fast and Efficient 3D LIDAR Semantic Segmentation," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, 2020.

[14] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "KPConv: Flexible and Deformable Convolution for Point Clouds," in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, 2019.

[15] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet++: Fast and Accurate LiDAR Semantic Segmentation ," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, 2019.

[16] 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. Vision and Pattern Recognition (CVPR)*, 2019.

[17] 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. IEEE Int. Conf. Comput. Vision (ICCV)*, 2019.

[18] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognition (CVPR)*, 2012.

[19] Stanford Artificial Intelligence Laboratory et al., "Robotic Operating System." [Online]. Available: https://www.ros.org

[20] NVIDIA, "TensorRT." [Online]. Available: https://developer.nvidia.com/tensorrt

[21] J. V. Hurtado, R. Mohan, W. Burgard, and A. Valada, "MOPT: Multi-Object Panoptic Tracking," *Proc. IEEE Conf. Comput. Vision and Pattern Recognition Workshops (CVPRW)*, 2020.