Improving Image Segmentation using Lidar Data

INF573 - Image Analysis and Computer Vision: algorithms and applications

École polytechnique
Academic Year 2023/2024

Prof. M.Bredif
Dr. J. Lutzeyer

Luca Brodo - luca.brodo@polytechnique.edu
Virgilio Strozzi - virgilio.strozzi@polytechnique.edu

Contents

1 Motivation

In the rapidly advancing domain of autonomous driving, the seamless integration of various sensor technology has emerged as the winning strategy for enhancing the perceptual capabilities of intelligent vehicles. As a matter of fact, while some automakers are adopting radically different approaches, such as adopting exclusively camera-based solutions ([17], [18]), the state-of-the art revolves around the adoption of multiple sensors ([15]) with the focus on fusing their data in order to alleviate the potential shortcomings that those might have. In this complex technological landscape, one of the most promising solutions in autonomous driving technology lies in the adoption of LiDAR (Light Detection and Ranging) sensors. LiDAR, as a remote sensing technology, plays a crucial role in capturing detailed three-dimensional spatial information about the environment. While LiDAR has been extensively used for mapping and object detection, it is not without its shortcomings. LiDAR primarily provides depth information, lacking the ability to distinguish color and texture details. Moreover, while LiDAR is proficient in detecting object boundaries and generating point clouds, the data it creates may not be indicated to extrapolate clear semantic information about the object in question. Furthermore, LiDAR performance can be affected by adverse weather conditions such as heavy rain, fog, or snow, which may hinder its ability to accurately detect objects. Interestingly, studies have demonstrated that the degradation in performance due to bad weather can even be attributed to interference by moisture in the air ([12], [5]). Furthermore, other studies have demonstrated that the performance of this type of sensor is also highly influenced by the material and the color of the objects. For example, Lee et al. in [10] conducted experiments revealing that LiDAR demonstrates superior performance when detecting colored objects, with its optimal efficiency observed in the presence of white-colored objects. Similarly, Park in [7] reached the same conclusion, also highlighting high performance when detecting retroreflective films. In addition, Roh et al. in [8] showed that LiDAR achieve the highest performances with materials such as aluminum, steel, plastic, and wood. In order to alleviate short this shortcomings, in this work, we aim at fusing RGB information captured by a vision sensor to the LiDAR data.
This sensor fusion strategy aims to leverage the strengths of both LiDAR and vision technologies, mitigating the limitations associated with individual sensors. By combining the information provided by LiDAR with the color and texture details captured by RGB sensors, the system aspires to create a more robust and adaptable perception framework, capable of overcoming the challenges presented by diverse environmental conditions that autonomous driving system must face to provide a satisfying service to the customers.
As stated before, while LiDAR excels in capturing precise depth information, it lacks the ability to discern color, making it challenging to distinguish objects based on visual characteristics alone. RGB images, on the other hand, provide a wealth of color details, enabling a more thorough understanding of the environment. This fusion of depth and color information enhances object recognition, allowing the system to identify objects based not only on their shapes but also in combination with their visual appearance. Moreover, the combination of LiDAR and RGB images addresses the limitation of LiDAR in generating clear semantic information about objects. LiDAR, while proficient in detecting object boundaries and creating point clouds, may struggle with semantic segmentation—assigning specific labels to different parts of an image. RGB images excel in capturing fine details and differentiating between object classes, but struggles more in detecting depth information. By merging RGB data with LiDAR, the system should

gain the ability to extrapolate clearer semantic information, contributing to more accurate scene understanding.
It is well known that semantic information are essential for learning driving policies and that can be effectively used as an intermediate representation for the same purpose ([2]), therefore in this work, we will focus on this specific problem. Namely, we will leverage the demonstrated capability of Deep Neural Networks (DNNs) in sensor fusion to combine LiDAR information to RGB information in order to improve performances in the image segmentation task, both in terms of flexibility and in terms of accuracy.

2 Related Work

Sensor fusion for LiDAR and vision sensors is a subject that has been study thoroughly in literature. While this approach has not been developed with any particular implementation as source of inspiration, multiple solutions can be found in literature with similar objectives or methodologies. Recently, Candan et al. in [3] proposed a similar architecture to the one in this work for road segmentation. In their study, they categorize the fusion technique used in this article as ”early fusion”. Alternatively, they also proposed a second architecture, categorized as ”late fusion”, which trains two parallel U-Net models to then fuse the last layer. The latter approach, however, not only is outperformed by the ”early fusion” technique in their benchmarks, but also requires double the processing power to be applied, both during training and during deployment, since two models are used to achieve the same effect. Since we are aiming at a deployment in the autonomous vehicle domain, where processing power and efficiency are of the highest importance, we are motivated to focus mainly on an ”early fusion” solution.
Liu et al. in [11] also propose a solution for road segmentation using RGB and LiDAR fusion. Similarly to the work in this paper, authors of [11] propose a model with a decoder-encoder architecture to achieve their segmentation. Additionally, they also study the performance using different architectures for the encoders, such as Resnet and VGG. In order to fuse the LiDAR data, however, contrary to out solution, they perform ”stage-wise fusion”, i.e. performing a fusion procedure at each stage, or step, in the model and then passing the fused data to the next block. Interestingly, the authors decide to apply fusion at the decoding stage, not encoding. They are motivated by the fact that the encoder does not clearly balance RGB data with LiDAR information. Furthermore, they point out that the amount of training data for this specific task is scarce and encoders require a lot of it, hence making over-fitting a likely possibility. While the second claim is partly true and stage-wise fusion is of particularly interest, considering also the approach they propose, it should be tested whether the scarcity of data and the unclear balance of information is deleterious for model performance and efficiency and, more importantly, whether a richer dataset could avoid this problem entirely.
The inherent sparsity of the LiDAR data is not extensively dealt with in this approach nor in the solutions mentioned above. Jaritz et al. in [6] deal with the sparsity of the data by using a different encoder architecture. Namely, they utilize a late fusion approach, therefore using two NASNet encoders, to obtain a semantic segmentation of the scene. While the need of two encoders, as mentioned above, will require more computational power, the flexibility of the model and the performance achieved demonstrate the efficacy of their approach.

3 Data Used

For the purpose of this work, we will represent the LiDAR point clouds as two-dimensional images. While other approaches might achieve highly promising solutions with other representation, we adopted 2D images mainly for convenience reasons.

3.1 Dataset

The KITTI dataset ([1]), acronym for "Karlsruhe Institute of Technology and Toyota Technological Institute", serves as a cornerstone in the realms of autonomous driving and computer vision research. Collected in 2011, this dataset encapsulates a diverse array of real-world data, including high-resolution images, Velodyne LiDAR point clouds, and GPS/IMU information, all captured from a sensor-equipped vehicle. The dataset's sensor suite comprises a high-resolution color camera, a grayscale Velodyne 3D lidar sensor, and an inertial measurement unit (IMU), providing a comprehensive view of the driving environment. Noteworthy for its manually annotated object categories—such as cars, pedestrians, and cyclists—the KITTI dataset enables the rigorous evaluation of algorithms related to object detection, tracking, stereo depth estimation, and 3D scene understanding. Organized into sequences that span urban, rural, and highway scenarios, KITTI has become a staple for benchmarking challenges, fostering the development of accurate and robust algorithms. With its availability in various data formats and widespread usage in academia and industry, KITTI continues to significantly impact advancements in computer vision and autonomous driving research, contributing to the ongoing evolution of these critical fields.
The information and data collected in this dataset will be used for the development adn test of the solutions proposed in this work. We will specifically leverage the data from the KITTI semantic instance segmentation benchmark, which focuses on 200 semantically annotated training images and an additional 200 test images. These images align with the KITTI Stereo and Flow Benchmark 2015, providing a specialized dataset for evaluating algorithms and models in the context of semantic instance segmentation. This subset of the KITTI dataset offers a curated collection of scenes, each meticulously annotated with semantic information, enabling researchers and practitioners to address the challenges of accurately delineating object instances within the visual field.

3.2 Coordinate Transformation

The right integration of LiDAR point cloud data with camera imagery is crucial for enhancing segmentation accuracy. The two sources of information must be coherent in order to achieve the highest degree of accuracy. This involves transforming the coordinates of one into the coordinates of the other. This transformation from LiDAR coordinates to the camera's frame of reference involves a series of mathematical operations. Initially, the LiDAR data is loaded and adjusted for calibration. Then, it undergoes a transformation from the LiDAR coordinate system to the camera's coordinate system. This transformation typically involves operations such as rotation, translation, and sometimes scaling to align the two coordinate systems properly.
The transformation from one coordinate system to another is often represented using matrices. The transformation matrix combines rotation, translation, and scaling operations into a single matrix that can be applied to the original data. Let's denote a point

in the LiDAR coordinate system as $P_L = [X_L, Y_L, Z_L, 1]$, where X_L, Y_L, and Z_L are the coordinates of the point in LiDAR space, and the fourth element 1 is added to make it a 4-dimensional homogeneous coordinate. The transformation from LiDAR to camera coordinates can be expressed as a matrix multiplication:

$$P_C = T \cdot P_L \tag{1}$$

Here, P_C is the transformed point in the camera coordinate system, and T is the transformation matrix. The transformation matrix T typically has the following structure:

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{2}$$

Where:

- R is a 3x3 rotation matrix.
- t is a 3x1 translation vector.
- The bottom row is $[0, 0, 0, 1]$ to make it a 4x4 matrix.

The rotation matrix R represents the rotation operations around the x, y, and z axes. For example, if θ_x, θ_y, and θ_z are the rotation angles around the x, y, and z axes, respectively, then the rotation matrix R would be:

$$R = R_z(\theta_z) \cdot R_y(\theta_y) \cdot R_x(\theta_x)$$

Each $R_i(\theta_i)$ is a rotation matrix for a specific axis rotation.

The translation vector t is a 3x1 vector representing the translation along the x, y, and z axes.

Once you have the rotation matrix R and translation vector t, you can combine them into the transformation matrix T as shown above. The transformation matrix is then applied to each point in the LiDAR data to convert it to the camera coordinate system. In summary, the transformation involves combining rotation, translation, and sometimes scaling operations into a single 4x4 matrix, which is then applied to each point in the LiDAR data. The use of homogeneous coordinates allows the representation of both rotation and translation in a unified manner. The mathematical process involves leveraging calibration parameters between the LiDAR and the camera, often obtained through calibration files or procedures. These parameters define the intrinsic and extrinsic properties of both devices, enabling the conversion of LiDAR points from their original space to the camera's frame of reference. This transformation ensures accurate alignment, allowing LiDAR data to be accurately projected onto the camera image. The resulting output merges LiDAR positional data with camera intensity information. This augmented dataset includes LiDAR points projected onto the camera image, providing both depth information (from LiDAR) and visual context (from the camera) for each point in the combined dataset.

4 Solution Proposed

4.1 Attempted Approach - Clustering

In the first approach we tried to increase segmentation accuracy involved clustering techniques. Clustering has been proven to achieve promising results in segmenting point

clouds ([19][20]), therefore it was trivial for us to make an attempt utilizing such techniques. While we realized image segmentation using clustering is not the state-of-the-art and other algorithm achieved better and more flexible results, our initial goal was to set up a pipeline around clustering algorithms. With the use of the transformation described in Section 3.2, we managed to obtain a dense map representation from the LiDAR data. In other words, we used the transformation in Section 3.2 to transform the point cloud coordinates into the camera coordinates and then use the distance information to create a dense map. This is generated using a kernel-based interpolation technique and an example can be seen in Fig. 1b (original can be seen in Fig. 1a).
As a result of the pipeline revolving around clustering, we tried to achieve a similar result to the depth map. We tried cluster points based on their measured distance. Intuitively, points having similar measured distances to each other belong to the same objects.
Unfortunately, we can conclude that this result did not bring the desired effects and we infered it would bring an additional unpredictable element in the overall approach. We, therefore, discarded the idea in favour of a solution completely based on Deep Learning techniques.

4.2 Final Approach - Deep Learning

As mentioned in previous parts of the reports, the approach which we focused on mostly during our work is based on Deep Learning techniques. The final idea is to create an ablation studies on different architectures, employing both RGB and LiDAR inputs, where we divide the models in 4 typologies, depending on their input: **RGB**, **LiDAR**, **LiDAR&RGB**, **LiDAR+RGB**.

4.2.1 Dataset

The final data used for this project consists of 200 320×1216 pixels RGB images from the KITTI dataset we described in Section 3.1, alongside with 200 320×1216 depth map generated as indicated in Section 3.2. We also employ for each of the 200 inputs a 320×1216 ground truth segmentation-mask with 4 different classes, divided in people, vehicles, bikes, others. This mask assign a value v $\in$ {0,1,2,3} to each pixel, representing the class of belonging of each pixel in the image.
This dataset is then split in two parts, with a proportion of 15% and 85% for the validation set and the training set respectively. Regarding pre-processing, the only technique we employ is a resize of both the RGB and LiDAR input to a size of 160x608 pixels and a normalization of the RGB images with the following values from the Imagenet dataset[4]: mean $= [0.485, 0.456, 0.406]$, std $= [0.229, 0.224, 0.225]$). The reason for the above is to allow bigger batch size due to the smaller image size for the first one and to allow the usage of pre-trained encoders on the Imagenet dataset for the second one. We do not employ further classical techniques of augmentation, such as random-tilting, jittering, color-shifting, due to the fixed camera used to gather all the images of the dataset. Employing one of the previous technique resulted in worst performances across all the models, as we expected.

4.2.2 Input Modalities

The main idea of our study consists in trying to understand how the addition of information to the RGB images of the street provided by the LiDAR can help the segmentation

of an autonomous driving scene. Therefore, we try to use the same models with minimal modification to allow the usage of different inputs, while producing as output the same segmentation mask of the scene.
What we try to infer in our ablation study is the difference of the various inputs' techniques under different backbone models.
The different inputs modalities are the following:

- **LiDAR**: The idea of this model is to simply use the depth-map of each LiDAR information of a scene and feed it inside the backbone model.
  The main motivation behind this architecture is to check whether the LiDAR information is enough to provide meaningful prediction on the objects in the street.
- **RGB**: In this architecture we use, as in the one employing LiDAR, only one part of the input, the RGB image, and we feed it inside the backbone model.
- **LiDAR&RGB (concat)**: In this architecture, we combine the LiDAR depth-map to the RGB image as an additional layer, generating an input image with four channels. The main motivation is to verify whether the model is able to use this additional information to generate better predictions.
- **LiDAR+RGB (sum)**: In this architecture, we combine the LiDAR depth-map to the RGB image as a simple sum over all the channels (with weight for the LiDAR information of 0.05), hence generating an input image with 3 channels. The main motivation for this approach is to verify whether the model is able to use this additional information to generate better predictions. Moreover, by combining the information in this way, we think we can better benefit from some of the pretrained weights of a model trained on ImageNet[4].

4.2.3 Backbone Models

When deciding the backbone architecture to use in our work we first studied two different architectures, UNet++ ([21]) and Efficienet ([16]). As a result of our pre-study, we found UNet++ to achieve the most promising resutls. The Unet++ is a more powerful version of the UNet. The architecture is essentially a deeply-supervised encoder-decoder network where the encoder and decoder sub-networks are connected through a series of nested, dense skip pathways. These re-designed skip pathways aim at reducing the semantic gap between the feature maps of the encoder and decoder sub-networks. We then decided to perform an ablation study on the structure of the encoder of the Unet++ with the following pre-trained flavours of ResNet: 18, 34, 50 and 1011 ayers. We also employ the Spatial and Channel SE Blocks (scse) attention type inside our network[14]. The decoder part of the Unet++ consists of five layers of the following size 512, 256, 128, 64, 32.

4.2.4 Training

The hyperparameters employed in the following part are a result of a finetuning process. Each model is trained on the training set consisting of 170 inputs, between LiDAR and RGB images. Each model is then validated after each epoch on the 30 inputs from the validation set.
During both types of training, we minimize a combination of the *Dice Loss* and the *Cross Entropy* as:

$$(1-\alpha)\text{DiceLoss}(\text{pred, truth}) + \alpha\text{CrossEntropy}(\text{pred, truth}) \tag{3}$$

where $\alpha = 0.9$, on the prediction of each model. The cross entropy is also weighted based on the presence of the various classes inside the model with the following values:

$$\begin{aligned}\textbf{People} &: 895.888014\\ \textbf{Vehicles} &: 13.28713976,\\ \textbf{Bikes} &: 910.52040434,\\ \textbf{Others} &: 1.08398173\end{aligned}$$

We keep the model's checkpoint with the smallest loss on the validation dataset across the epochs of training for the inference stage.
To allow reproducibility, we fix the seed of each possible generator to 77. Moreover we make use of the Adam Optimizer [9] for all of the models and set the initial learning rate to 0.0002 for all the models. We fix a batchsize of size 8 for each model due to CUDA out of memory limit errors if increased further.
When training the models we use a total of 20 epochs for all the models and we keep as a result the checkpoint of the model achieving the smallest validation loss.
All the meaningful hyperparameters of the models are derived empirically and reported on the results.

4.2.5 Post Processing

As a post process for the output of each model, we simply take the logits output of the model per pixel and take the index corresponding to the highest value. This index indicates the class of each pixel.

5 Evaluation

We show the results of our models in the Table 1. Every model is trained and tested on Kaggle on a GPU P100, on the dataset described in the previous section. The metric employed is the IoU Metric.

For a quantitative analysis, we examine Table 1. We observe that the highest Intersection over Union (IoU) score in the validation set is achieved by the model utilizing only the RGB image. The reasons we believe justify these results are as follows. Firstly, we employed a pretrained encoder model trained on the ImageNet dataset, perfectly adapted for incorporating only RGB information. Following the same argument, when adding Lidar information as a new channel (LIDAR&RGB), we obtain worse results than when incorporating it as a new value added to all RGB channels (LIDAR+RGB). In the latter case, we still leverage more utility from the pretrained weights of the ImageNet.
The least effective model type is the one using only LiDAR information, which is plausible since LiDAR information is less complete for the task of object detection compared to RGB information. However, even with this limitation, we still obtain meaningful predictions, particularly producing sharp boundaries for nearby objects to the LiDAR sensor as can be observed in Figure 2.
The LiDAR+RGB model overfits and produces the highest IoU metric result on the

training set. We are confident that the dataset utilized in the experiment was insufficient to train such larger models. Expanding the size of the models with different ImageNet backbones does not lead to higher scores; on the contrary, we generally obtain lower values due to the more parameters to fit which do not find enough data. We believe that another extension of this work should undoubtedly consider a larger dataset.

In addition to the quantitative analysis, we also performed qualitative analysis on the models.
TWe begin by considering the way predictions are inferred by each model, as illustrated in the example in Figure 2. We observe differences in the prediction patterns of each model. For example, the LiDAR model excels at predicting objects close to the sensor, producing sharp and accurate predictions. On the other hand, the RGB model tends to overpredict the vehicles class, likely because it is the most prevalent class in the dataset. This could also be the reason why it achieves the highest IoU metric, even though regularization is applied through the loss. Moreover its predictions are the one with more noise across all the approaches, even though they capture harder and more ambigous examples. The RGB+LiDAR segmentation mask presents a fairly accurate prediction by identifying most objects in the scene, albeit with some noise. Similarly, the RGB&LiDAR segmentation mask also incorporates good predictions by identifying more objects than only cars, as in the RGB prediction, but it integrates even more noise. It is also interesting to notice that all the models employing the LIDAR information have more difficulty to predict the car hidden behind the poles, with an exception for the RGB&LiDAR, which seems to be able to use more the RGB information compared to the one of the LiDAR. The approach RGB+LiDAR seems to use more the LiDAR information by producing sharper predictions than its counterpart, but ignoring smaller or more hidden objects. A future work could investigate in more details how the two approaches differ in the way they predict.
It is worth noting that most models predict parts of bikers as being a person, which is accurate in practice but contradicts the truth label. Hence, the model seems to learn meaningful predictions from the limited training data, even though the labels can be debated.
While quantitative analysis gives us an overview on the model performance, we are not able to extract other important information such as robustness of the models. In our scenario, robustness is an important element to monitor, since false predictions might be the cause of catastrophic consequences. An example of out qualitative analysis can be seen in Fig. 3. Fig. 1a shows the original image we applied our analysis on, while Fig. 3b represents the transformed LiDAR data. To understand model performances when presented with perturbances, we occluded part of the image using a black box. We applied this on the original image Fig. (Fig. 3c) and on the concatenated LiDAR data (Fig. 3d). Comparing Fig. 3g with Fig. 3h we can observe how the two models react to the perturbance. The model using only RGB data is not able to efficiently deal with it and creates artifacts. The model using the augmented data in as we can see from Fig. 3h, it is still able to segment the car, albeit with higher degree of uncertainty (see Fig. 3f). It is worth noting that we did not train the model on the new data. In other words, the model is trained on normal data and then the augmented data has been fed at end of the training.

6 Conclusion

The aim of this work was to study whether the addition of LiDAR information can effectively improve the performance of classical Neural Networks for the image segmentation task in the field of autonomous driving. While our approach has achieved promising results, multiple are the challenges which are remained to be faced. Firstly, the results we obtained are clearly below the expected results in similar benchmarks. We already provided a brief explanation for this reason, i.e. the lack of a more comprehensive dataset for this specific task. One direction which must be taken is to create a more comprehensive dataset to better train the model on. Furthermore, while we provided a use-case, the model we trained must be considered as a basis to further build upon. We opted for a classical architecture which is vastly used, but novel techniques and architectures must be considered for a more thorough analysis. Furthermore, while we explored some techniques for data fusion and we mentioned some others, there still exist a plethora of approaches. We still believe that other techniques can be applied to achieve better results. For example, in combination with an ad-hoc dataset, we believe the attention mechanism can be used to further increase performances, at the clear cost of more power hungry solutions with an higher computational cost and time. Similarly, other techniques must be explored for the representation of the LiDAR data. We suspect that, while convenient, the representation of this data as RGB might not be powerful enough. Neural Network architectures, such as PointNet++([13]) could be used in combination with another representation, such as 3D point cloud, to achieve higher scores.
In conclusion, although it is still at its initial stage, the achieved results suggest that this approach can lead to a proper fusion of these two fundamentally different types of data. We are confident in saying that this approach could also be applied in many other tasks, such as object detection and avoidance, to increase the performances of autonomous cars all around the world.

References

[1] Hassan Alhaija, Siva Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision (IJCV)*, 2018.

[2] Aseem Behl, Kashyap Chitta, Aditya Prakash, Eshed Ohn-Bar, and Andreas Geiger. Label efficient visual abstractions for autonomous driving. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, October 2020.

[3] Arda Taha Candan and Habil Kalkan. U-net-based RGB and LiDAR image fusion for road segmentation. *Signal, Image and Video Processing*, 17(6):2837–2843, 09 2023.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[5] Christopher Goodin, Daniel Carruth, Matthew Doude, and Christopher Hudson. Predicting the influence of rain on lidar in adas. *Electronics*, 8(1):89, 2019.

[6] Maximilian Jaritz, Raoul de Charette, Émilie Wirbel, Xavier Perrotton, and Fawzi Nashashibi. Sparse and dense data with cnns: Depth completion and semantic segmentation. *CoRR*, abs/1808.00769, 2018.

[7] Bum jin Park Ji yoon Kim. A Study of LiDAR's Detection Performance Degradation in Fog and Rain Climate. *The Journal of The Korea Institute of Intelligent Transport Systems*, 21:101–115, 2022.

[8] Jisoo Kim, Bum-jin Park, Chang-gyun Roh, and Youngmin Kim. Performance of mobile lidar in real road driving conditions. *Sensors*, 21(22):7461, 2021.

[9] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[10] In-Su Lee and Jae-One Lee. Performance evaluation of terrestrial laser scanner over calibration baseline. *Journal of the Korean Society of Surveying, Geodesy, Photogrammetry and Cartography*, 28(3):329–336, 2010.

[11] H. Liu, Y. Yao, and Z. et al. Sun. Road segmentation with image-LiDAR data fusion in deep neural network. *Multimed Tools Appl*, 79:35503–35518, 2020.

[12] Karl Montalban, Christophe Reymann, Dinesh Atchuthan, Paul-Edouard Dupouy, Nicolas Riviere, and Simon Lacroix. A quantitative analysis of point clouds from automotive lidars exposed to artificial rain and fog. *Atmosphere*, 12(6), 2021.

[13] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017.

[14] Abhijit Guha Roy, Nassir Navab, and Christian Wachinger. Recalibrating fully convolutional networks with spatial and channel 'squeeze & excitation' blocks. *CoRR*, abs/1808.08127, 2018.

[15] Debanjan Saha and Shuvodeep De. Practical self-driving cars: Survey of the state-of-the-art. *Preprints*, February 2022.

[16] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[17] Tesla, Inc. Transitioning to tesla vision, 2023. Accessed: January 16, 2026.

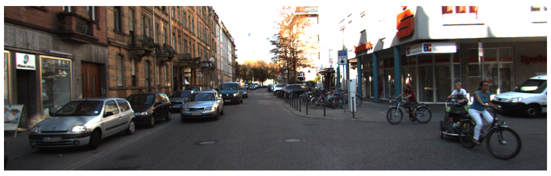
[18] Teslarati. Toyota adopts tesla's camera-only approach to self-driving development, 2023. Accessed: January 16, 2026.

[19] Feng Wang, Guoqing Zhou, Jiali Xie, Bolin Fu, Haotian You, Jianjun Chen, Xue Shi, and Bowen Zhou. An automatic hierarchical clustering method for the lidar point cloud segmentation of buildings via shape classification and outliers reassignment. *Remote Sensing*, 15(9), 2023.

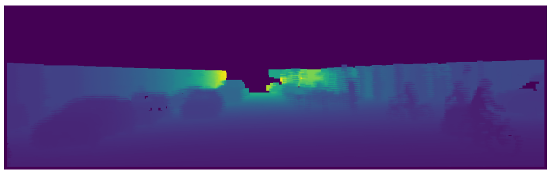
[20] Yiming Zhao, Xiao Zhang, and Xinming Huang. A divide-and-merge point cloud clustering algorithm for lidar panoptic segmentation. *CoRR*, abs/2109.08224, 2021.

[21] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. *CoRR*, abs/1807.10165, 2018.

7 Appendix

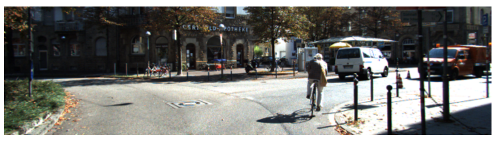


(a) Original Image

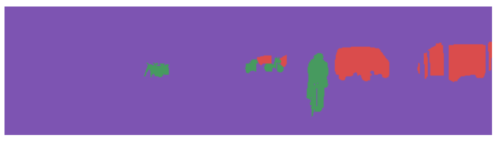


(b) Example of the depth map generated by kernel-based interpolation (6 neighbours)

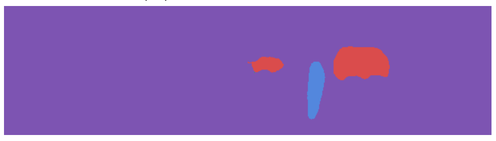
Figure 1: Original image and related depth map



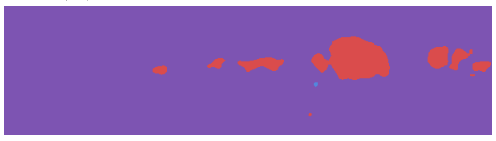
(a) Original Image



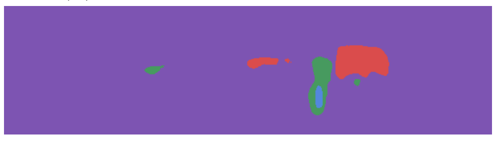
(b) Original Segmentation Mask



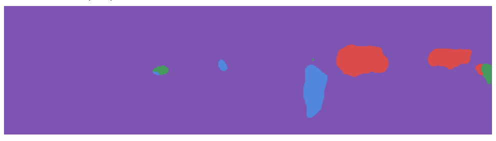
(c) LiDAR Segmentation Mask



(d) RGB Segmentation Mask

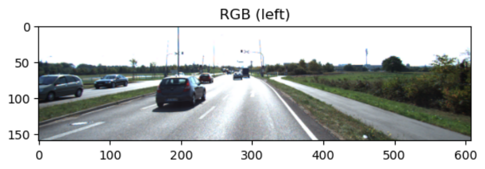


(e) RGB+LiDAR Segmentation Mask

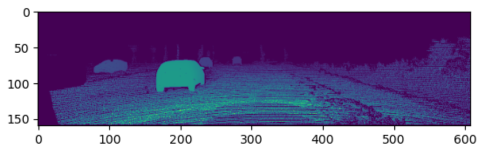


(f) RGB&LiDAR Segmentation Mask

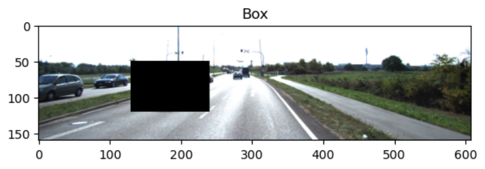
Figure 2: An image from the validation set with different predictions of the various model with the resnet50 backbone. All the predictions are taken from the checkpoint of each model with the smallest validation loss. The color of the segmentation masks are the following: blue for the person class, green for the bikes class, red for the vehicle class and purple for the other class.



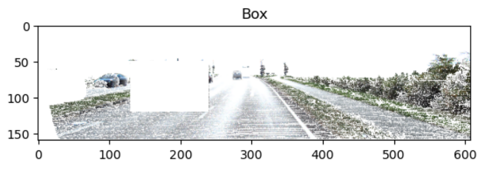
(a) Original Image



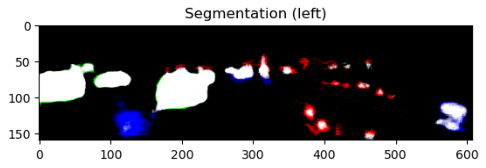
(b) Original LiDAR Data



(c) Original Image with Black Box

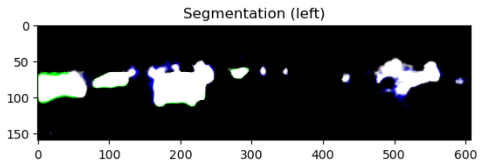


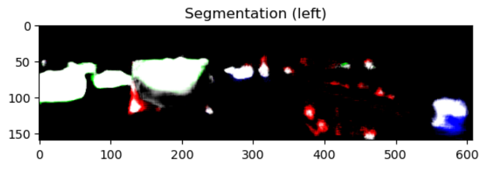
(d) Original + LiDAR with Black Box



(e) Original Segmented image without LiDAR data (RGB)

(f) Original Segmented image with LiDAR data (LiDAR+RGB)





(g) Segmented image without LiDAR data (RGB)

(h) Segmented image with LiDAR data (LiDAR+RGB)

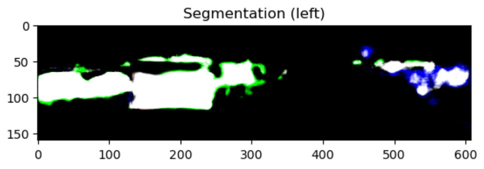


Figure 3: Original image and related depth map

Results						
Model Type	#Parameters	Mean GPU Power (Watt)	Max GPU Memory (%)	Train Time (s)	IoU (Train)	IoU (Valid)
LiDAR, Resnet18	22M	59.98	30.27	917	0.7341	0.4572
LiDAR, Resnet34	32M	58.05	31.61	904	0.5780	0.4358
LiDAR, Resnet50	64M	82.01	47.87	963	0.4949	0.4697
LiDAR, Resnet101	83M	87.74	58.8	991	0.4628	0.4467
RGB, Resnet18	22M	41.12	32.27	930	0.7170	0.5135
RGB, Resnet34	33M	43.26	37.67	911	0.6518	**0.5221**
RGB, Resnet50	64M	83.44	60.47	970	0.6770	0.4749
RGB, Resnet101	83M	87.44	53.93	1007	0.7248	0.4629
LiDAR&RGB, Resnet18	22M	61.34	30.87	897	0.6309	0.4657
LiDAR&RGB, Resnet34	32M	58.91	30.67	906	0.5718	0.4751
LiDAR&RGB, Resnet50	64M	82.12	58.33	936	0.6699	0.4442
LiDAR&RGB, Resnet101	83M	84.61	57.87	1009	0.6932	0.4657
LiDAR+RGB, Resnet18	22M	55.06	31.87	916	0.7526	0.4733
LiDAR+RGB, Resnet34	32M	59.23	32.65	901	**0.8426**	0.4461
LiDAR+RGB, Resnet50	64M	86.66	49.20	961	0.7609	0.4482
LiDAR+RGB, Resnet101	83M	85.12	47.33	1036	0.6447	0.4974

Table 1: In the table, we present the results obtained on the validation set after individually training each model on the same training set. The column 'Model Type' indicates the model by presenting the type of input first and the backbone architectures employed. Specifically, 'LiDAR' indicates the usage of LiDAR information alone, 'RGB' uses only the image, 'LiDAR&RGB' represents the concatenation of LiDAR and RGB images, and 'LiDAR+RGB' denotes the addition of LiDAR information on top of the RGB channels (with a weight of 0.05 for the LiDAR). We display only the maximum Intersection over Union (IoU) score achieved by each model. Additionally, we provide information on the number of parameters (in millions), the mean GPU power used during training (in watt), and the maximum GPU P100 memory usage as a percentage (out of 16GB). We have highlighted the best scores for both IoU in the validation set and IoU in the training set.