Optimizing Learned Object Detection on Point Clouds from 3D Lidars Through Range and Sparsity Information

Jacob Lambert[1], Eijiro Takeuchi[1], Kazuya Takeda[1]

***Abstract*—Supervised learning methods for object detection on 3D lidar point cloud have recently emerged. Image-based algorithms, naively adapted to this new sensing modality, were shown to be somewhat effective. However, the data structure of point cloud differs significantly from images, and approaches taking this into consideration must be developed. In this research, we demonstrate how point sparsity, which depends on range from the lidar, complicates the training of lidar-based detection model. We use an expected point sparsity metric to filter the KITTI dataset and improve the overall quality of the training labels. We perform in-depth ablation studies which show how much an optimized training dataset can improve 3D lidar object detection models. Finally, we show that including range information directly in the network input, as a indicator for expected point sparsity, also improves detection capabilities.**

I. INTRODUCTION

A crucial task for the development of mobile autonomous systems is object detection. The majority of autonomous platforms are equipped with some exteroceptive sensors, traditionally cameras, which are capable of relaying information about the surrounding environment. With these perception capabilities, it then becomes possible, albeit difficult, to detect objects of interest in the surroundings. For example, automated cars may be particularly interested in locating other vehicles, or pedestrians and cyclists so as to avoid any risky situations. Regardless of the object of interest, camera-based object detection method have been thoroughly researched by the computer vision community. In recent years, methods relying on supervised learning and deep convolutional neural networks (DCNNs) have unequivocally dominated object detection benchmarks.

However, alternatives to cameras have seen growing popularity. Depth cameras and radars have seen some use, but most notably, 3D lidars have become a key component in many autonomous mobile platforms. Lidars offer many direct advantages over cameras, such as having their own power-source; they emit infra-red lasers which perform equally well in low light conditions, and are less susceptible to intemperate weather. The data provided by the lidar, a large array of sparsely distributed 3D points representing the environment, known as a *point cloud*, is complementary to a camera's dense 2D color information, suggesting sensor fusion. While using multiple sensing modalities is preferable in practice, it introduces reliability issues, and so knowing the capabilities of individual sensors remains crucial.

[1]All authors are with Takeda Laboratory, in the Department of Intelligent Systems, Graduate School of Informatics, Nagoya University, 1 Furo-cho, Chikusa-ku, Nagoya-shi, Aichi-ken, Japan.

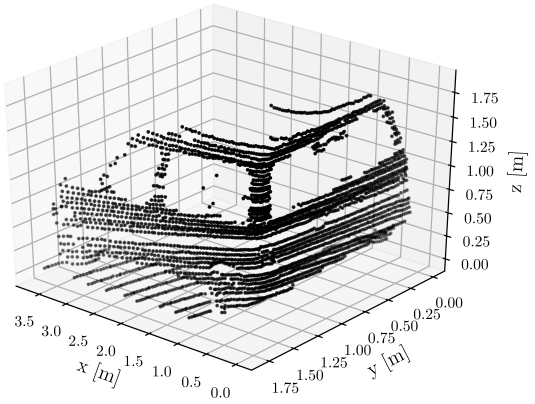

Fig. 1: Segmented point cloud inside a *car* class label in the KITTI dataset, which contains 3063 points, at a distance of roughly 7.5 meters from the sensor.

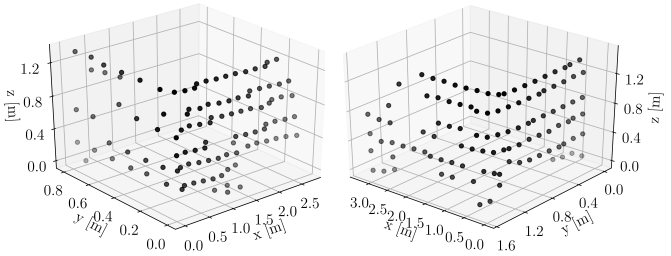

Fig. 2: Two segmented point clouds, each containing 107 points and at a distance of roughly 35 meters from the sensor. One corresponds to a *car* class bounding box, the other is a building corner. The image on the right is the car.

Assessing the object detection capabilities of 3D lidars has been the subject of many recent research. Traditional approaches relied on complex heuristic-based clustering and segmentation methods, but recently, supervised learning methods analogous to the ones used on images have been applied to lidar data. It is a natural assumption that the success of 2D convolutional networks for object detection would translate to different types of data, including 3D range information. Modern lidars also report increasingly reliable single channel intensity information, which deepens the connection to images. On the other hand, lidar data is fundamentally different to camera data. The most glaring difference is the increased dimensionality of the data, making it unnatural to directly apply 2D CNNs directly. To quickly apply existing methods for 2D object detection on 3D lidar data, several authors have opted to project 3D data information to 2D images.

Typically, the data is either observed from the top, the so-called *bird's eye view* (BEV)[1], or the lidar data is arranged in a 2D matrix, with intensity corresponding to range or depth, often called *front-view image* or *depth image*[2]. By including information like height or intensity as additional 2D channels, these methods have seen relative success, even for predicting 3D bounding boxes. At the same time, convolutional neural networks naturally extend to 3D, and recent methods operating directly on undistorted 3D point cloud have further improved lidar-based detection accuracy.

Nevertheless, state-of-the-art methods have not been fully adapted to this new data type. Recent work improved the discretization of point clouds by implementing feature encoding networks[3], and utilizing sparsity to improve computation time has also been researched[4]. While implemented on 3D lidar data, these methods were originally developed for volumetric data[5, 6] which differs significantly from 3D lidar data. Otherwise, the impact of range on object appearance has been largely ignored. One of the main challenges in image-based object detection is that objects can be found at different scales, but 3D point cloud data is fixed in metric scale. Still, this sensing modality does not circumvent appearance variation for the same object class; instead, object appearance varies significantly with range, due to the increasing sparsity of the data. This is highlighted Figure 1 and 2, where in the bottom figure it is already difficult to distinguish a car from a wall at a range of 35 meters and with more than 100 lidar points. Yet, in the KITTI dataset, there are thousands of objects at greater range and with far fewer points.

In this work, data-driven methods were developed to improve the reliability of training on 3D lidar point cloud. Specifically, a statistical analysis of the point cloud labels in the KITTI data set shows the varying sparsity of data. Through comparative studies, the hypothesis that some labels of poor quality should be omitted from training was tested by comparing detection accuracy given different training dataset. The contributions of this paper are experiments that show:

- a simple point sparsity threshold improves detection accuracy,
- a range-based threshold that takes into account sensor specifications further improves accuracy, and
- including range information directly as an additional network input is also beneficial.

While the suggested methods were tested on the KITTI dataset and the SECOND object detection model[4], we stress that any 3D lidar-based detection method which rely on voxel representations could apply this research. Furthermore, the analysis performed in this work could also be applied to other datasets.

The rest of the paper is structured as follows: recent research on 3D lidar-based object detection is first discussed in Section II. In Section III, current issues with training 3D lidar-based approaches on the KITTI dataset are shown, and solutions are proposed. Then, the results of thorough ablation studies are shown in Section IV, by first comparing the impact of our modifications with the SECOND baseline implementation, and then by comparing with the state-of-the-art on the KITTI 3D benchmarking suite.

II. RELATED WORK

A. 2D Representations

The extensive literature on the object detection for 2D images motivated the early attempts at object detection on point cloud data to adopt 2D representations equivalent to images. This allowed existing method and software to be applied to point clouds with minimal modification and kept computation tractable.

The VeloFCN network[2] proposed to use single-channel front-view image as input to a shallow, single-stage convolutional neural network which produced 3D vehicle proposals. LMNet[7] expanded on this idea, first by adding several other channels to the input, such as lidar intensity and range to the sensor. Another use of depth image was shown for semantic classification of lidar points[8]. Through depth image-based object labels, alongside pointwise flow estimation[9] for lidar point cloud, they achieved accurate pointwise segmentation of lidar data.

The alternative approach, BEV images have seen widespread popularity, likely in part due to the new KITTI benchmark. A drawback of this top-view projection is that lidar points which vary primarily in height occlude each other in BEV images. MV3D[1], as well as multi-channel BEV images, each channel correspond to a different range of heights, so as to minimize these occlusions. Alongside these height maps, camera images, front-view images, density maps and intensity maps are used to produce object proposals which are then fused and refined in a deep fusion network, which performs multi-class detection and 3D bounding box regression. Complex-YOLO[10] borrows heavily from this work, but uses exclusively a 3-channel BEV representation that encodes height, density and intensity as input to the YOLOv2[11] network. Other notable BEV research use slightly different inputs or networks[12, 10, 13, 14], while some use map-based masks to pre-process the BEV data and increase accuracy and computational cost [15, 16].

B. 3D Representations

Using volumetric data directly, without the need for projection in 2D space, has been the topic of many recent works. A 3D representation, through 3D occupancy grid called voxel grids, was first applied for object detection in RGB-D data[17], which is structurally similar to 3D lidar data. Shortly thereafter, a similar approach was used on point clouds created by lidars [18]. In their following work, they used a 3-layer CNN for car, pedestrian and cyclist detection[19].

Following the aforementioned VeloFCN, a 3D dimensional version of the CNN was proposed[2]. This network used a single channel voxel grid, storing only binary occupancy, as representation of the 3D point cloud. While this network produced decent results, the first truly convincing results for point cloud-only 3D bounding box estimation was shown in VoxelNet[3]. Instead of hand-crafting input features or simply

using occupancy, VoxelNet used recent research in voxel feature encoding (VFE) to learn CNN input features[5]. The VFE uses a fully connected neural network that converts the variable number of points in each voxel, to a feature vector of fixed size. The region proposal network[20] receiving inputs from the VFE performed multi-class object detection with state-of-the-art accuracy. This work was then improved both in terms of accuracy and computational efficient by SECOND[4] by using focal loss and exploiting the natural sparsity of lidar data to significantly improve computation time. SECOND is very notable as the first 3D method to be released open-source by the authors; other works had not been shown to be entirely reproducible.

Finally, PointRCNN[21], who recently achieved state-of-the-art on the KITTI benchmark using only lidar data, uses a novel feature encoding method: an autoencoder produces pointwise feature vectors which are then used for foreground (object) point segmentation. Each foreground points then produce 3D bounding box proposals which are refined through a second network. This latest work represents significant divergence from the earlier, image-based computer vision literature.

III. Methodology

A. KITTI Dataset Analysis

The KITTI dataset was created in 2012 as a computer vision benchmarking tool[22]. In this section, some analytical details of the dataset are provided as basis for this work. Despite being somewhat dated, it remains one of the best sources of labelled 3D lidar data. The lidar used in the dataset is a Velodyne HDL-64, which has a 360° field of view in the horizontal and angular resolution of $\theta_{hor} = 0.08°$. Its 64 vertically-spaced lasers are angularly spaced by approximately $\theta_{ver} = 0.4°$, for a total vertical field of view of 26.9°.

The dataset provides 7481 training examples with full 3D labels, as well as 7518 test-only examples, used for official benchmarking, with 45,570 training labels of various classes, though detection is benchmarked on 3 classes: cars, pedestrians and cyclists. These classes are further divided into 3 categories, easy, moderate or hard, depending on the bounding box height, occlusions and whether the object is only partly in the image plane, known as truncation. It is important here to note that the 3D bounding box as well as occlusion and truncation level are defined through images, and that only objects in the image plane are labelled.

Unfortunately, the quality of image labels does not guarantee the quality of the point cloud label. To show this, the 3D bounding box parameters are used to segment the points contained in each bounding box. Figure 3 shows the number of bounding boxes with a specific number of points, from 0 to 100 points. While some bounding boxes are dense with data, many can be seen to be sparse; more than 500 boxes contain no points whatsoever. Given this, it can be asked whether it is reasonable to expect a 3D point cloud-only algorithm to detect these objects. More specifically, this work investigates whether these labels should be omitted from training, and to what extent.

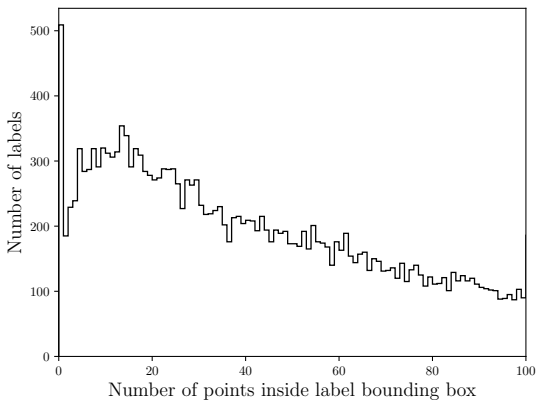

Fig. 3: Histogram showing the number of labels containing a specific number of lidar points.

B. Label Filtering

The previous section discussed some glaring issues with the quality of the training data in the KITTI dataset. To understand the impact of these less than adequate labels when training object detection algorithms, a filter was applied to the dataset prior to training. Note only the training set was filtered, not the validation set, and that a 50:50 split was followed as in previous work[4].

The first method used is simply to discard any bounding box which has less points than some hard threshold. To avoid confusing signals during training, the voxels that contain discard labels do not contribute any loss.

One argument against this approach is that points at higher distance should be allowed to have a small number points, as sparsity naturally increases with distance from a 3D lidar sensor. However, as shown in Figure 5, labels with low number of points occur at essentially all distances. Another approach is therefore to apply a threshold based on the distance from the sensor. A distance-based threshold based on the specific sensor used and the statistics of the *car* class in the KITTI dataset was derived: the Velodyne HDL64 lidar used in the KITTI dataset has a horizontally angular resolution of $\theta_{hor} = 0.08°$ and vertical angular resolution of $\theta_{ver} = 0.4°$ and is at a height of $h_l = 1.73$ m from the ground. The mean length, width and height of a car in the KITTI dataset is $l = 4.00$ m, $w = 1.65$ m and $h = 1.59$ m respectively. Given that car length is much greater than width or height, a vehicle with the same heading as the ego-vehicle will create the smallest number of lidar points, so this represents a worst case scenario. Assuming the ground is locally flat, previous work shows how to estimate number of lidar points on a 3D bounding box a distance r from the sensor[23]:

$$N_{ver}(r) = \frac{1}{\theta_{ver}} \left(\arctan\left(\frac{h_l}{r}\right) - \arctan\left(\frac{h_l - h}{r}\right) \right), \quad (1)$$

$$N_{hor}(r) = \frac{2}{\theta_{hor}} \arctan\left(\frac{w}{2r}\right) - 1, \quad (2)$$

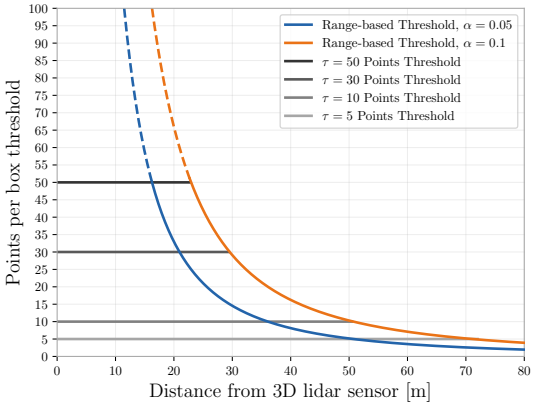

Fig. 4: Range-based thresholding function, for $\alpha = 0.1$ (orange) and $\alpha = 0.05$ (blue) for 4 different maximum thresholds, τ. For example, the horizontal line at 50 points means any label with at least 50 points will be included in the training dataset, regardless of the function at close range, shown by a dotted line.

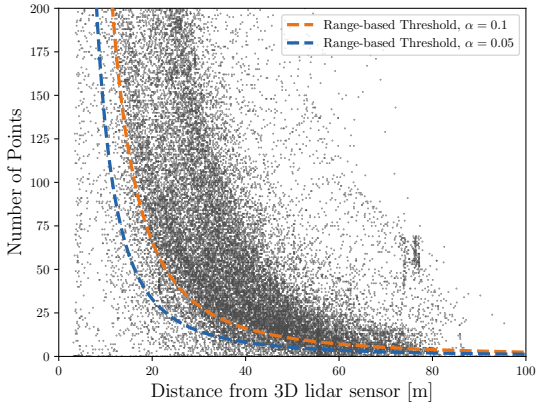

Fig. 5: Bounding box distance from sensor versus the number of lidar points inside the bounding box, for boxes with less than 100 points. The dashed line is the range-based thresholding function from Section III-B with $\alpha = 0.1$ represented by the orange line and $\alpha = 0.05$ by the blue line; under this criteria, labels below or to the left would be filtered.

where $N_{ver}(r)$ is the number of points in a vertical slice, and $N_{hor}(r)$ is the number of points in a horizontal slice. Therefore, the total number of points is given by:

$$N_{pts}(r) = N_{ver}(r) \cdot N_{hor}(r). \tag{3}$$

In practice, the area of a car is much smaller than that of a rectangular prism. Furthermore, the fact that cars may be occluded, further reducing the number of hits, has to be accounted for. A weight factor of α was therefore added to the function, based on the distribution of points in Figure 5.

Furthermore, the number of points get very large near the sensor, but in that region, the assumption that the entire vehicle is in the sensor field of view tends to be false. A max threshold for this range-based function for the near region, τ, is used. The resulting function is:

$$N_{pts}(r, \alpha, \tau) = \min\left(\alpha \cdot N_{ver}(r) \cdot N_{hor}(r), \tau\right) \tag{4}$$

which essentially applies a hard threshold near the sensor, and then gradually reduces the number of points. We show this function for different max threshold in Figure 4. Note that when filtering bounding boxes, the function was rounded down to the nearest integer.

In Table I, the number of training labels given the different filtering parameters is shown for some key classes, to give perspective on how much data is sacrificed. In Section IV-A, the impact of filtering on detection accuracy is shown, specifically for the *car* class. This work does not perform multi-class detection, but this range-based filtering approach could be extended by following same analysis for other classes.

	Val+Train	Train			
Min. Points N_{pts}	All	All	Cars	Pedest.	Cyclists
0	40570	19700	14357	2207	734
1	40061	19361	14031	2186	722
2	39876	19231	13891	2180	712
3	39647	19083	13718	2168	699
4	39408	18930	13536	2157	693
5	39089	18744	13331	2139	685
10	37588	17893	12470	2029	657
15	35963	16997	11539	1910	627
20	34482	16262	10866	1815	601
30	31777	14906	9613	1647	546
50	27701	12931	8095	1362	459
100	21041	9712	5723	877	270
$\alpha = 0.1, \tau = 10$	36957	17464	12591	2045	651
$\alpha = 0.1, \tau = 30$	34929	16342	11830	1830	606
$\alpha = 0.05, \tau = 5$	39356	19294	13635	2185	715
$\alpha = 0.05, \tau = 10$	39068	19126	13529	2151	707
$\alpha = 0.05, \tau = 30$	38763	18917	13430	2084	699

TABLE I: Number of labels in the dataset after filtering, for different filtering criteria. The first group is a hard minimum threshold, each label must contain a minimum of N_{pts}. The second group is range-based according to Eq.4, up to a threshold τ.

C. Range Input

As previously stated, euclidean distance to sensor, or *range*, is an important factor determining the appearance of an object as perceived by a 3D lidar. Since point positions are typically encoded with respect to the voxel grid cell origin during the discretization process, range information is lost. To test the hypothesis that this information is invaluable for detection, the impact of including range information as an additional channel is investigated. This additional channels simply holds the euclidean distance between the sensor and the 3D point, normalized between $\{0, 1\}$ by the maximum possible distance to the sensor, with the range used for detection being $[0, 70.4] \times [-40, 40] \times [-3, 1]$.

This approach is compared with other choices of input,

namely (i) using only occupancy, (ii) using occupancy and intensity (baseline), (iii) using occupancy and range and (iv) using occupancy, intensity and range. The results of these experiments are reported in Section IV-B.

IV. RESULTS

In the following experiments, the default SECOND[4] training parameters are used, along with their 50:50 training and validation split, only training on the car class.

A. Label Filtering

This section reports the results of filtering labels based on the number of points they contain. The baseline model, as shown in Table II has no minimum point requirement, which is then increased incrementally up to 100 points. The results of these experiments show that, up to a minimum number of points of 30, the trained model performs slightly better than the baseline model. The best model was obtained when only labels with a minimum of 4 points are considered, which filters about 6% of the cars in the dataset. When the threshold is set at 50 or 100 points, the accuracy for the moderate and hard categories goes down significantly, while the easy category sees only a very small decrease in accuracy. This is likely because hard thresholding has the consequence of preferably filtering labels farther from the sensor; a high threshold may leave only few examples at high range to train the model on, while only slightly reducing the number of close-range examples.

Motivated by the shortcomings of a hard threshold, a thresholding function based on expected sparsity of the lidar sensor was developed. It was tested for two values of $\alpha = \{0.05, 0.1\}$ and a few values of $\tau = \{5, 10, 30\}$. While the hard threshold showed disproportional change in accuracy across label difficulties, this approach takes range into account and therefore is much more consistent. This approach considerably outperforms the baseline model, especially in the moderate class, with the best model using $\alpha = 0.05$ and $\tau = 30$, which is not very aggressive in terms of how many labels are filtered, but still has a significant impact on model output. Table I shows that using $\alpha = 0.05$ and $\tau = 30$ filters a number of labels equivalent to a hard threshold between 4 or 5 points. A value of $\alpha = 0.1$ is comparatively more aggressive filtering and performs only sightly better than hard thresholding.

B. Range Input

Detection accuracy for different inputs to the voxel feature encoder are shown in Table III. A significant improvement from including range information was shown when compared to only using occupancy. However, adding range alongside intensity information produced only a marginal improvement to detection accuracy.

C. Optimized Network

We test an optimized network model on the KITTI test set and compare it with the state-of-the-art as reported on the official KITTI 3D benchmark[1]. The aforementioned 50:50 training and validation split was used for consistency with previous experiments. The network input is occupancy, intensity and range, and a range-based filtering approach was applied to the training labels, with $\alpha = 0.05$ and $\tau = 30$.

The internal comparison on the validation set in Table IV shows the compounded accuracy improvement of this approach. While significantly better than the baseline, including range information was shown to produce only marginal accuracy improvements on the moderate and hard class. On

	Cars		
Min. Points N_{pts}	Easy	Moderate	Hard
0 (baseline)	88.30	77.98	76.40
1	88.45	78.36	76.61
2	88.39	78.20	76.41
3	88.86	78.44	76.83
4	**88.96**	**78.86**	**77.43**
5	88.31	78.27	76.75
10	88.51	78.31	76.50
15	88.38	78.33	76.64
20	87.92	78.05	76.66
30	88.04	77.97	76.47
50	88.18	70.33	69.44
100	87.03	61.63	53.11
$\alpha = 0.1, \tau = 10$	89.43	82.20	77.67
$\alpha = 0.1, \tau = 30$	88.99	81.90	77.47
$\alpha = 0.05, \tau = 5$	89.94	85.59	78.01
$\alpha = 0.05, \tau = 10$	89.66	85.47	77.91
$\alpha = 0.05, \tau = 30$	**89.94**	**85.80**	**78.34**

TABLE II: AP for the car class on the validation set after filtering labels with less than N_{pts}. Hard thresholding is shown first for different values of N_{pts}, followed by range-based filtering based on $N_{pts}\,(r, \alpha, \tau)$.

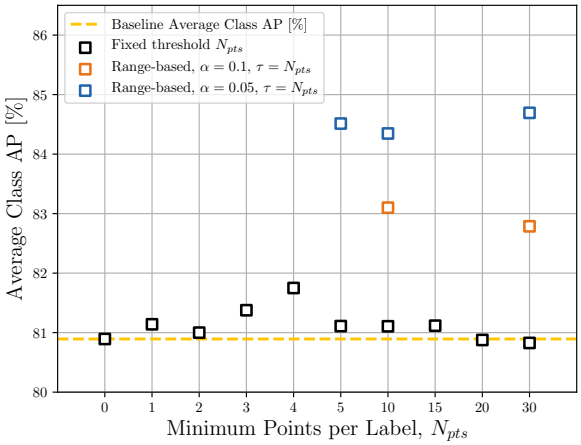

Fig. 6: Average cars class AP on the validation set, for different filtering parameters, with baseline AP shown in red. The black markers have a fixed minimum number of points per label N_{pts}, while the blue points have minimum number of points only near the sensor N_{pts}, as shown in Figure 4.

[1]Published work only, accessed on *February 26, 2019*

	Cars		
Input Type	Easy	Moderate	Hard
Occupancy+Intensity (baseline)	88.30	77.98	76.40
Occupancy	76.25	61.49	55.51
Occupancy+Range	84.35	66.31	59.54
Occupancy+Range+Intensity	**88.68**	**78.08**	**76.46**

TABLE III: AP on the validation set using different input types.

	Cars		
Model	Easy	Moderate	Hard
Baseline	88.30	77.98	76.40
Occupancy+Range+Intensity	88.68	78.08	76.46
Filtering, $N(r, \alpha = 0.05, \tau = 30)$	**89.94**	85.80	78.34
NU-optim	89.92	**85.82**	**78.48**

TABLE IV: Peformance of the NU-optim model which uses range-based label filtering and additional range input.

the KITTI benchmark, our method, labelled *NU-optim* on the official KITTI benchmark shows improvements over SECOND and competes closely with other state of the art methods.

Finally, in terms of computation time, including range information in the input is a trivial modification to the original SECOND algorithm.

V. CONCLUSIONS

In this work, some simple approaches to get the most out of 3D lidar-only detection algorithms based on supervised learning were developed. While it has to be accepted that labeling lidar point clouds directly is difficult, it can also be acknowledged that labeling bounding boxes through images can lead to poor label quality. In this work, it was shown explicitly that poor label quality is detrimental to training object detection models, specifically for the KITTI dataset. Inspecting the training dataset and filtering out labels that do not contain sufficient number of lidar points was shown to be an efficient way to avoid improve the dataset and obtain more reliable detection results. A range-based filtering approach, which takes into account the sensor, was shown to produce even more consistent training sets, which further improved detection accuracy. However, it has to be acknowledged that the KITTI training set is rather small, which accentuates the importance of label quality. It remains to be shown that this filtering is beneficial for datasets 10^1 to 10^2 times larger than KITTI, which is common of image-base applications. Since no such lidar-based open dataset exist at this time, this is left as future work.

Additionally, this work has shown that using range as additional input to the VFE framework work is another optimization that improves detection accuracy. However, the improvement was not as significant as the one obtained through dataset filtering. This work still shows that taking into account range-based appearance changes is a key factor that makes object detection from lidar point clouds challenging. As such, future work will aim to develop network structures and training loss functions that can address this issue more directly.

		Cars		
Algorithm	Speed [s]	Easy	Moderate	Hard
PointRCNN[21]	0.10	85.94	75.76	68.32
PointPillars[14]	0.016	79.05	74.99	68.30
NU-optim (Ours)	0.04	**83.92**	**74.94**	**66.88**
RoarNet[24]	0.10	84.25	74.29	59.78
SECOND[4]	0.04	83.13	73.66	66.20
IPOD[25]	0.20	82.07	72.57	66.33
AVOD-FPN[13]	0.10	81.94	71.88	66.38
F-PointNet[26]	0.17	81.2	70.39	62.19
VoxelNet (Lidar)[3]	0.23	77.47	65.11	57.73
MV3D (Lidar)[1]	0.24	66.77	52.78	51.31

TABLE V: Average Precision (AP) in % on the KITTI 3D object detection test set, ordered based on *moderate* category accuracy.

ACKNOWLEDGMENT

We would like to acknowledge that this work was supported by the Japan Science and Technology Agency (JST) project on Open Innovation Platform with Enterprises, Research Institute and Academia (OPERA).

REFERENCES

[1] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 6526–6534.

[2] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D lidar using fully convolutional network," in *Proceedings of Robotics: Science and Systems*, June 2016.

[3] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End learning for point cloud based 3D object detection," Nov. 2017.

[4] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, Oct. 2018.

[5] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," Dec. 2016.

[6] B. Graham and L. van der Maaten, "Submanifold sparse convolutional networks," June 2017.

[7] K. Minemura, H. Liau, A. Monrroy, and S. Kato, "LMNet: Real-time multiclass object detection on CPU using 3D lidar," in *Asia-Pacific Conference on Intelligent Robot Systems*, July 2018, pp. 28–34.

[8] L. Liu, Z. Pan, and B. Lei, "Learning a rotation invariant detector with rotatable bounding box," Nov. 2017.

[9] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, "Rigid scene flow for 3D LiDAR scans," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2016, pp. 1765–1770.

[10] W. Ali, S. Abdelkarim, M. Zahran, M. Zidan, and A. El Sallab, "YOLO3D: End-to-end real-time 3D oriented object bounding box detection from LiDAR point cloud," Aug. 2018.

[11] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6517–6525.

[12] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D object detection from point clouds," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 7652–7660.

[13] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3D proposal generation and object detection from view aggregation," in *IEEE/RSJ Conference on Intelligent Robots and Systems*, Oct 2018, pp. 1–8.

[14] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," Dec. 2018.

[15] B. Yang, M. Liang, and R. Urtasun, "HDNET: Exploiting HD maps for 3D object detection," in *2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 2018, pp. 146–155.

[16] M. Ren, A. Pokrovsky, B. Yang, and R. Urtasun, "SBNet: Sparse blocks network for fast inference," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8711–8720.

[17] S. Song and J. Xiao, "Sliding shapes for 3D object detection in depth images," in *European Conference on Computer Vision*. Springer International Publishing, 2014, pp. 634–651.

[18] D. Z. Wang and I. Posner, "Voting for voting in online point cloud object detection," in *Robotics: Science and Systems*, July 2015.

[19] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3Deep: fast object detection in 3d point clouds using efficient convolutional neural networks," in *IEEE International Conference on Robotics and Automation*, May 2017, pp. 1355–1361.

[20] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, pp. 91–99.

[21] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," Dec. 2018.

[22] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[23] P. Narksri, E. Takeuchi, Y. Ninomiya, Y. Morales, and N. Kawaguchi, "A slope-robust cascaded ground segmentation in 3D point cloud for autonomous vehicles," in *IEEE Conference on Intelligent Transportation Systems*, Nov. 2018, pp. 497–504.

[24] K. Shin, Y. P. Kwon, and M. Tomizuka, "RoarNet: A robust 3D object detection based on region approximation refinement," Nov. 2018.

[25] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "IPOD: Intensive point-based object detector for point cloud," Dec. 2018.

[26] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," Nov. 2017.