Technical Report

Anonymous WACV Applications Track submission

Paper ID 2736

1. Detect Object Presence

1.1. Local Peak Search Algorithm

We use Algorithm 1 to find local peaks. In this algorithm, the input includes s_num, which represents the number of slices we aim to create, and $P.d$ which represents the value of the data point in coordinate d. The output, however, is the list of the range where the detected peaks are located. In Algorithm 1, we first get the statistics of the number of points that fall into the s_num slices, and the edge of each slice (lines 2 to 4). Then, we iterate through the slices with the given stride and find the local peaks within each stride range lines 6 to 8.

Algorithm 1: Peak search

```
Input  : $s\_num, P.d, stride$
Output: $l\_range$
1 $l\_range = \emptyset$
2 max, min = get_max_min_value(P.d)
3 value_range = max - min
4 l_count, l_bin_edge = get_histogram(value_range,
    s_num)
5 for $i \leftarrow stride$ to $|l\_count| - 1 - stride$ do
6     $idx = get\_idx\_of\_max\_value(l\_count[i - stride, i + stride]);$
7     $range = [l\_bin\_edge[idx], l\_bin\_edge[idx + stride]];$
8     $l\_range.add(range);$
9 return $l\_range$
```

1.2. Time complexity analysis of Density Peaks base Solution

For the proposed method, we analyze the efficiency of each step: For slicing, the complexity is $\mathcal{O}(|P|)$. For peak searching, the complexity is $\mathcal{O}(m+n)$, where m and n are the numbers of slices in the x and y axes respectively. To detect objects by checking crossing slices in the x and y axes, the complexity is $\mathcal{O}(m \times n)$. Thus, the overall complexity is $\mathcal{O}(|P|)$ + $\mathcal{O}(m+n)$ + $\mathcal{O}(m \times n)$. Due to m and n are very small compared to $|P|$, the complexity is dominated by $|P|$ and is $\mathcal{O}(|P|)$.

1.2.1 Data-driven parameter study

In order to implement this method effectively, a detailed parameter study is essential. The key parameters include m, n (the number of slices in x and y axes respectively), as well as $stide_x$ and the $stride_y$ for the peak searching. Due to variations in point clouds collected by different LiDAR devices, which cause variations in resolutions and viewable ranges, these parameters must be derived from the training data.

Within the point cloud, various types of objects are present, each associated with a bounding box. To determine m and n, we leverage the following information from the dataset: For each object category, we get the **average number of unique points** along the x and y axes. Subsequently, we derive the minimum and maximum values across all object categories for both dimensions, denoted as uc^o_{minX}, uc^o_{maxX}, uc^o_{minY} and uc^o_{maxY}.

The objective of our slicing process is two-fold: (1) Accurately detecting the appearance of an object, requiring slices to be neither too small nor too large. (2) Ensuring complete object detection and avoiding partial object retention, necessitating an appropriate selection of $stirde$.

We dynamically adjust m, n, $stride_x$ and $stride_y$ based on the point cloud P. Let uc^P_X and uc^P_Y denote the unique number of points in the x dimension and the y dimension of p. Then, we have $m = uc^P_X / uc^o_{minX}$, $n = uc^P_Y / uc^o_{minY}$, $stide_x = uc^o_{maxX} / uc^o_{minX}$, and $stride_y = uc^o_{maxY} / uc^o_{minY}$. Intuitively, the slicing process should encompass the smallest object, while the stride ensures that even larger objects are effectively covered.

1.3. Prepare Training Data for Naïve Bayes

In Section 3.3.1, to create the Naïve Bayes model, we need the statistics of the region containing objects. We use Algorithm 2 to verify whether a region r contains objects. From the labeled training data, we can obtain the bounding box of each labeled object. Each bounding box comprises 8 corner points, denoted as $P_{corner} = p_1, p_2, ..., p_8$. In Algorithm 2, the inputs contain the set of labeled bounding boxes, denoted as $\mathcal{P}_{obj}$, the slice range in the X-axis which is denoted as $range\_x$, and the slice range in the Y-axis which is denoted as $range\_y$. In this algorithm, we iterate each bounding box (denoted as P_C) and check whether any part of the object overlaps with the bounding box (lines 3 to 6). If any overlap is detected, then we recognize the region r contains the object (lines 8 to 9).

```
Algorithm 2: check object existence
   Input : $\mathcal{P}_{obj}$, $range\_x$, $range\_y$
1  $isObj = False$;
2  for $P_C \in \mathcal{P}_{obj}$ do
3      $max\_x, min\_x = get\_max\_min\_value(P_C.x)$;
4      $max\_y, min\_y = get\_max\_min\_value(P_C.y)$;
5      $check\_x = (range\_x[0] \leq max\_x \ \& \ range\_x[1] \geq min\_x)$;
6      $check\_y = (range\_y[0] \leq max\_y \ \& \ range\_y[1] \geq min\_y)$;
7      if $check\_x \ \& \ check\_y$ then
8          $isObj = True$;
9          break;
10 return $isObj$;
```

1.4. Naive Bayes Explaination

$$P(y_r^{obj}|r.d_x, r.d_y) = \frac{P(y_r^{obj}) \times P(r.d_x, r.d_y|y_r^{obj})}{P(r.d_x, r.d_y)} \quad (1)$$

To understand the above equation, $P(y_r^{obj}|r.d_x, r.d_y)$ represents the probability that region r is categorized as an object, given the point density of slices along the x and y axes where r locates (which is denoted as $r.d_x$ and $r.d_y$). $P(y_r^{obj})$ represents the probability of region r being recognised as containing objects. This data can be obtained from the training data. $P(r.d_x, r.d_y|y_r^{obj}) = P(r.d_x|y_r^{obj}) \times P(r.d_y|y_r^{obj})$. This probability represents the probability of the appearance of the density in d_x and d_y under the condition that region r contains objects,. $P(r.d_x, r.d_y) = P(r.d_x) \times P(r.d_y)$ represents the probability of density d_x and d_y appearing in region r respectively. Those two probabilities are constants and can be obtained by statistics of the dataset, so we can infer $P(y_r^{obj}|r.d_x, r.d_y) \propto P(y_r^{obj}) \times P(r.d_x, r.d_y|y_r^{obj})$. For the classification task, we use $\hat{y_r} = argmax_y P(y_r^{obj}) \times P(r.d_x, r.d_y|y_r^{obj})$.

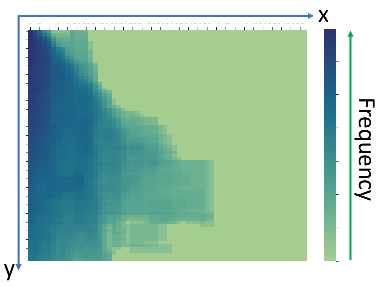


Figure 1. Probability of object appearance at regions

2. Study of Datasets

2.1. Object Distribution

In Figure 1, we present the appearance rate of objects in each region based on the training data from KITTI. Light colors indicate regions where objects are less likely to appear, while dark colors signify a high likelihood of object presence. Notably, more than half of the regions never feature object appearances. Due to the sparsity of the point cloud, when considering the statistics for the entire training set, only 1% of the regions contain objects. Consequently, the training data for our binary classification task exhibits bias, contributing to the poor performance of the MLP-based method.

2.2. Lower Bound of Sample Rate

We leverage two key statistical attributes from the dataset: the average count of points for each category of a single object, denoted as $c_i.points$, and the occurrence frequency of each object category within a given point cloud, denoted as $c_i.freq$. The minimum required number of points to be sampled, denoted as $|P|_{min}$, is computed as the summation of $c_i.points \times c_i.freq$ for each category $c \in c_1, c_2, ...$ Subsequently, we establish the lower bound of the sample rate as $rate_{lower} = |P|_{min}/|P|_{avg}$. Table 1 shows an example of statistics of the KITTI dataset. Using this approach, we estimate that the lower bound sample rate for the KITTI dataset is approximately 10%, while for the nuScenes dataset, it is around 7%.

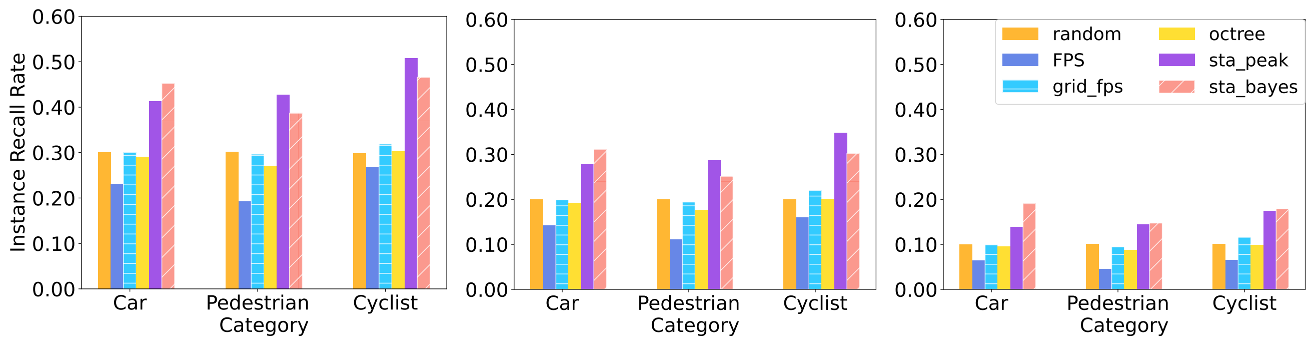
3. More Experiment Results

3.1. Study of Instance Recall Rate

Analysing Categorical mAP with Instance Recall Rate. We introduce a concept called Instance Recall Rate by referring [2], which is the rate of points retained to depict the object. In Figure 2 and 3 at range of sample rates, we demonstrate the Instance recall rate for Part-A2 with KITTI. By reading Figures 2 and 3 with 4 and 5 in conjunction, we can find that a higher instance recall rate would lead to a higher mAP for most cases.

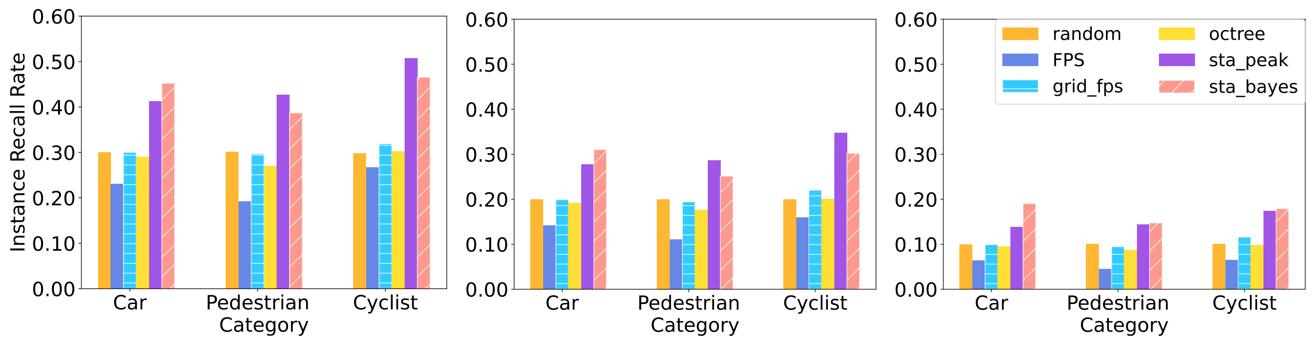
Table 1. Statistics of KITTI

Category	Car	Pedestrian	Cyclist	Truck	Van	Tram	Person_Sitting	Misc
avg points/obj	326	207	141	412	555	584	205	429
avg obj/pc	3.8	0.6	0.2	0.16	0.4	0.07	0.04	0.16



(a) Sample Rate 30% (b) Sample Rate 20% (c) Sample Rate 10%

Figure 2. Categorical Instance Recall rate with Part-A2 on KITTI (high sample rate)



(a) Sample Rate 8% (b) Sample Rate 6% (c) Sample Rate 4%

Figure 3. Categorical Instance Recall rate with Part-A2 on KITTI (low sample rate)

Among all the compared methods, our proposed sta_peak and sta_bayes exhibit notably high instance recall rates compared to all baselines. This observation provides insight into why these methods excel in terms of effectiveness. Additionally, in the range of lower sampling rates from 8% to 4%, sta_bayes retains a more substantial number of object points compared to sta_peak, as indicated in Figures 3.

It is worth noting that, although FPS shows exceptional performance at high sampling rates, the instance recall rate is the lowest. This highlights a valuable lesson: to effectively support downstream object recognition tasks, preserving both instance points and feature-rich points holds equal significance.

3.1.1 Comparison with Trained MLP (Q5)

In this section, we experimented with performing the Object Presence Detection with deep learning methods. We compared our proposed methods with deep learning-based training. We trained two separate models with different strategies: point-level training and region-level training.

Point-level training. We utilized PointNet [1] to extract point features, and we trained the binary classification head with three fully connected layers followed by ReLU. We compared the performance of the point-level trained model with our non-learning density peak-based method, as shown in Table 3. The results revealed that the MLP-based method struggles to outperform the non-learning density peak method. When the sample rate decreased below 10%, the difference became more obvious, which indicated the effectiveness of the non-learning density peak method.

Table 2. Comparison of Region-level Trainings

Rate	**sta_bayes**			Region-level MLP			MLP+FPFH		
	Easy	Med	Hard	Easy	Med	Hard	Easy	Med	Hard
50%	76.98	62.25	58.59	71.08	56.67	53.16	70.86	55.80	51.98
30%	69.52	53.88	50.53	59.93	46.01	42.74	61.16	46.04	42.21
10%	46.99	32.67	30.05	28.91	19.69	17.71	31.05	20.73	18.56
6%	33.69	22.64	20.34	16.54	10.93	9.05	15.64	10.19	8.69
2%	10.49	6.42	5.33	1.70	1.21	0.97	1.73	1.30	1.04

Table 3. Comparison of Point-level Trainings

Rate	**sta_peak**			Point-level MLP		
	Easy	Med	Hard	Easy	Med	Hard
50%	74.26	60.49	56.96	71.45	56.55	53.16
30%	66.65	53.89	49.34	60.97	46.46	42.74
10%	44.33	32.41	28.81	29.74	20.01	17.71
6%	29.50	20.92	17.89	16.22	10.71	9.05
2%	8.23	5.15	4.16	1.89	1.23	0.97

Region-level training. Table 2 demonstrated the comparison of region-level based training results. By following the design of the Naïve Bayes solution, we utilized region density as the feature to train a model for the binary classification of regions (which is Region-level MLP). We also used Fast Point Feature Histograms (FPFH) as a feature extractor for the region, considering MLP was more sensitive to features compared to Naïve Bayes. We trained another model that combined both region density and FPFH feature (which is MLP+FPFH). According to the results, Naïve Bayes showed a great advantage compared to MLP-based methods. Although incorporating features from FPFH results in a slight performance increase, the improvement was not substantial. This highlighted the robustness and efficacy of the Naïve Bayes method for regional-level categorization.

Table 4. Study of Different Strategies

Strategy	Acc (%)	Precision (%)
Naïve Bayes	97.0	60.4
MLP	99.3	4.5
MLP + FPFH	99.4	4.5

Analysis. As shown in Table 4, we analyzed the reason for the terrible performance of MLP-based training by looking into the accuracy and precision of the prediction results. For the region-based method, both region-level MLP, and region-level MLP+FPFH, the accuracy was notably high, while precision revealed a different story with an extra low value (with 4.5%). Only Naïve Bayes achieved a relatively high precision (with 60.4%). The disparity between accuracy and precision could be attributed to the utilization of cross-entropy loss during model training. The scenario of high accuracy coupled with low precision often arose when the training data exhibited an extraordinarily uneven distribution. For detailed insights into the distribution of the training data, please refer to Section 2.1 in this report.

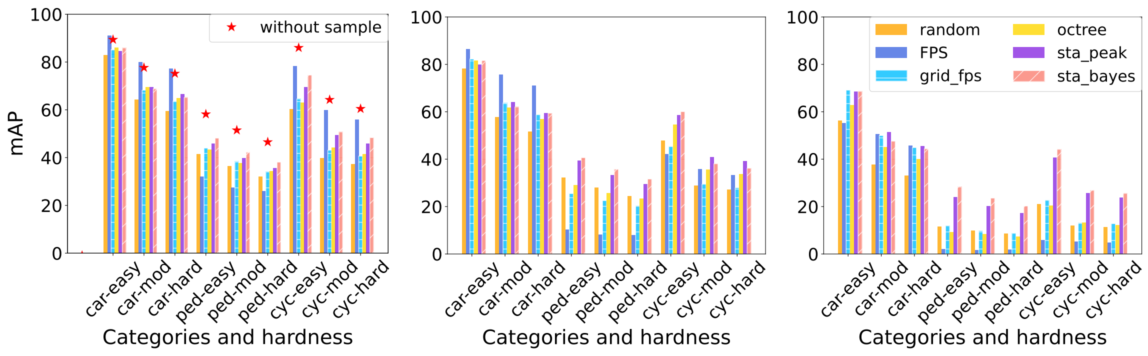
3.2. Results on More Sample Rates

Figure 4, Figure 5, Figure 6, and Figure 7 demonstrate the results for a wider range of sample rates.

References

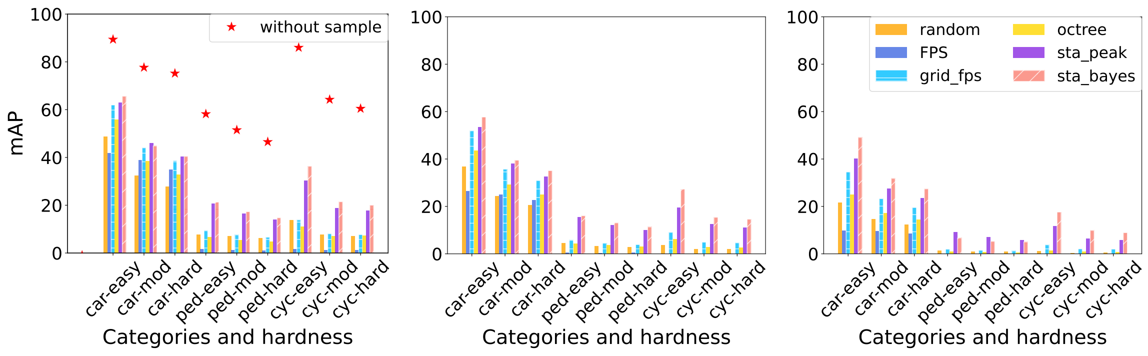
[1] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 3

[2] Yifan Zhang, Qingyong Hu, Guoquan Xu, Yanxin Ma, Jianwei Wan, and Yulan Guo. Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18953–18962, 2022. 2



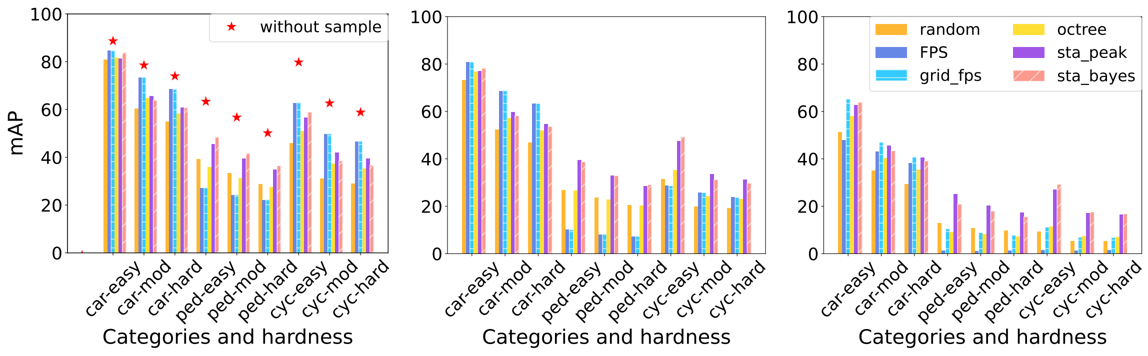
(a) Sample Rate 30% (b) Sample Rate 20% (c) Sample Rate 30%

Figure 4. Categorical mAP with Part-A2 on KITTI (with higher sample rate)



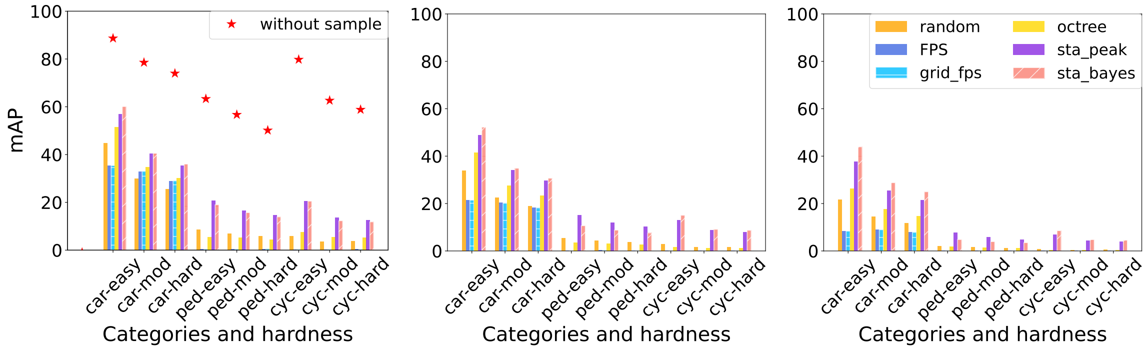
(a) Sample Rate 8% (b) Sample Rate 6% (c) Sample Rate 4%

Figure 5. Categorical mAP with Part-A2 on KITTI (with lower sample rate)



(a) Sample Rate 30% (b) Sample Rate 20% (c) Sample Rate 10%

Figure 6. Categorical mAP with SECOND on KITTI (with higher sample rate)



(a) Sample Rate 8% (b) Sample Rate 6% (c) Sample Rate 4%

Figure 7. Categorical mAP with SECOND on KITTI (with lower sample rate)