RESEARCH ON RADAR DATA CLUSTERING

CUI QUAN,WANG CHUNYU

1. Introduction

The rapid development of autonomous driving technology necessitates robust data preprocessing techniques to ensure the reliability and efficiency of autonomous systems. Among the crucial stages in data preprocessing is the application of clustering methods to group similar data points together. Therefore, This research

2. Internal Index

Since the goal of clustering is to make objects in the same cluster similar and objects in different clusters different, most internal validations are based on the following two criteria:

- **Compactness measure:** How closely objects are in the same cluster. Compactness can be measured in different ways, such as using the variance of the points within each cluster, or calculating the average pairwise distance between them.
- **Separation measure:** The degree to which a cluster is distinct or separated from other clusters. Examples of separation measures include pairwise distances between cluster centers or pairwise minimum distances between objects in different clusters.

Commonly used internal effective index includes the following:

2.1. SSE (Sum of Squared Errors).

$$\text{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Where:

- n is the number of observations.
- y_i represents the actual values of the observed data points.
- $\hat{y}_i$ represents the predicted values of the model for the corresponding data points.

2.2. **Silhouette Coefficient (SC).** The Silhouette Coefficient (SC) is a measure of how well-separated clusters are in a clustering algorithm, which quantifies the similarity of each data point to its own cluster compared to other clusters.

$$SC(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Where:

- $a(i)$ is the average distance from the i-th data point to other data points in the same cluster.
- $b(i)$ is the average distance from the i-th data point to data points in the nearest cluster (i.e., the cluster that is not the one to which i belongs).

The overall Silhouette Coefficient for the clustering is the average of $SC(i)$ for all data points. The SC ranges from -1 to 1, where a high SC value indicates that the object is well-matched to its own cluster and poorly matched to neighboring clusters, while a low or negative SC value indicates that the object may be assigned to the wrong cluster.

2.3. **Calinski-Harabasz Index (CHI).** The Calinski–Harabasz index (CHI), also known as the Variance Ratio Criterion (VRC), is defined as the ratio of the between-cluster separation (BCSS) to the within-cluster dispersion (WCSS), normalized by their number of degrees of freedom.

$$CHI = \frac{BCSS}{WCSS} \times \frac{k-1}{n-k}$$

Where:

BCSS (Between-Cluster Sum of Squares): is the weighted sum of squared Euclidean distances between each cluster centroid (mean) and the overall data centroid (mean):

$$BCSS = \sum_{i=1}^{k} n_i ||\mathbf{c}_i - \mathbf{c}||^2$$

Where:

- n_i is the number of points in cluster C_i.
- $\mathbf{c}_i$ is the centroid of C_i.
- $\mathbf{c}$ is the overall centroid of the data.

BCSS measures how well the clusters are separated from each other (the higher the better).

WCSS (Within-Cluster Sum of Squares): is the sum of squared Euclidean distances between the data points and their respective cluster centroids:

$$WCSS = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \mathbf{c}_i||^2$$

WCSS measures the compactness or cohesiveness of the clusters (the smaller the better). Minimizing the WCSS is the objective of centroid-based clustering algorithms such as k-means.

2.4. **Dunn Index (DI).** The Dunn index aims to identify dense and well-separated clusters. It is defined as the ratio between the minimal inter-cluster distance to maximal intra-cluster distance.

$$DVI = \frac{\min_{i \neq j} \{d(c_i, c_j)\}}{\max_k \{d_{\max}(c_k)\}}$$

Where:

- c_i, c_j, c_k represent different clusters.
- $d(c_i, c_j)$ denotes the shortest distance between different clusters.
- $d_{\max}(c_k)$ represents the maximum distance among points within the same cluster.

2.5. **V-measure.** V-measure is a metric used to evaluate clustering results, combining the concepts of homogeneity and completeness, and then harmonically averaging them to provide a comprehensive assessment of clustering effectiveness.

Homogeneity measures the extent to which each cluster contains only members of a single class. A clustering result is fully homogeneous if each cluster only contains members of a single class, in which case its homogeneity score is 1.

Completeness measures the extent to which all members of a given class are assigned to the same cluster. A clustering result is fully complete if all members of a given class are assigned to the same cluster, in which case its completeness score is 1.

The formula for V-measure is given by:

$$V = \frac{2 \times h \times c}{h + c}$$

where h represents homogeneity and c represents completeness. Homogeneity h and completeness c are calculated as follows:

$$h = 1 - \frac{H(C|K)}{H(C)}$$
$$c = 1 - \frac{H(K|C)}{H(K)}$$

In these equations, $H(C|K)$ and $H(K|C)$ are the conditional entropies of given clustering K and given true classes C respectively. $H(C)$ and $H(K)$ are the entropies of classes C and clustering K respectively.

2.6. **Davies-Bouldin Index (DBI).** The DBI computation involves the average distances within clusters and the distances between cluster centroids, utilizing the ratio of the sum of the average distances to the distance between centroids. A lower DBI indicates better clustering effectiveness.

$$DBI = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right)$$

Where:

- k is the number of clusters.
- c_i, c_j represent different clusters.
- σ_i, σ_j are the average distances from the samples in a cluster to the cluster's centroid.
- $d(c_i, c_j)$ is the distance between the centroids of clusters c_i and c_j.

3. External Index

The external evaluation index is evaluated based on the dataset of known classification labels, so that the original label data can be compared with the clustering output. The ideal clustering result of the external index is that data with different class labels are aggregated into different clusters, and data with the same class labels are aggregated into the same cluster.

External evaluation criteria are usually measured using indicators following:

3.1. **Purity (P).** One of the simplest and most intuitive ways to evaluate clustering results is to calculate its purity. In simple terms, clustering purity is the number of samples that are clustered correctly divided by the total number of samples.

$$Purity(\Omega, C) = \frac{1}{N}\sum_{k}\max_{j}|\omega_k \cap c_j|$$

Where:

- N is the total number of samples.
- $\Omega = \{\omega_1, \omega_2, \cdots, \omega_k\}$ is cluster division.
- $C = \{c_1, c_2, \cdots, c_j\}$ is true class division.

3.2. **Normalized Mutual Information (NMI).**

$$NMI(\Omega; C) = \frac{I(\Omega; C)}{(H(\Omega) + H(C))/2}$$

Where:

- I is Mutual Information (MI).
- H is entropy.

The formulas for calculating I and H are defined as follows:

$$I(\Omega; C) = \sum_{k}\sum_{j} P(\omega_k \cap c_j)\log\frac{P(\omega_k \cap c_j)}{P(\omega_k)P(c_j)}$$
$$H(\Omega) = -\sum_{k} P(\omega_k)\log P(\omega_k)$$

Where:

- $P(\omega_k)$, $P(c_j)$, and $P(\omega_k \cap c_j)$ are the probability that the sample belongs to cluster ω, belongs to category c, and belongs to both, respectively.

The minimum value of MI is 0, and when the cluster and the class are independent, Ω does not bring any useful information for C. If the resulting Ω is more closely related to C, then $I(\Omega; C)$ the higher the value. If C is reproduced in its entirety by the Ω, the MI is maximum, meanwhile $I(\Omega; C) = H(\Omega) = H(C)$. Since the entropy is introduced and the entropy will increase with the increase of the number of clusters when MI increases, NMI will punish the clustering results with a large number of clusters, and the expectation that the smaller the number of clusters, the better, can be formalized when other conditions are the same.

3.3. **Rand Index (RI) and Adjusted Rand Index (ARI).** The Rand Index is a metric used to measure the similarity of two datasets; The Adjusted Rand Index is an improved version of the Rand Index, which can remove the influence of random labels on the Rand Index evaluation results.

For the pair confusion matrix in the clustering problem, we can define it as follows:

- TP: the number of cases in the same cluster of two similar sample points;
- FP: the number of cases in which two non-homogeneous sample points are in the same cluster;
- TN: the number of cases in two clusters of two non-homogeneous sample points;
- FN: the number of cases in two clusters of two similar sample points;

The formulas are defined as follows:

$$RI = \frac{TP + TN}{TP + FP + TN + FN}$$

$$\begin{aligned} ARI &= \frac{RI - E[RI]}{MAX(RI) - E[RI]} \\ &= \frac{2 \cdot (TP \cdot TN - FN \cdot FP)}{(TP + FN)(FN + TN) + (TP + FP)(FP + TN)} \end{aligned}$$

Where:

- $E[RI]$ is the expectation of the Rand Index,
- $MAX(RI)$ is the maximum of the Rand Index.

3.4. **Accuracy (AC).** Clustering accuracy (AC) is used to compare the obtained labels with the true labels provided by the data. The formula for calculating accuracy is defined as follows:

$$AC = \frac{\sum_{i=1}^{n} \delta(s_i, \text{map}(r_i))}{n}$$

Where:

- r_i is the clustered label,
- s_i is the real label,
- n is the total number of data,
- $\delta(x, y)$ indicates the indication function, as follows:

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

4. Algorithms and Evaluation

4.1. **K-mean.** K-means is an iterative algorithm for clustering data into K distinct groups. It minimizes the sum of squared distances between data points and the centroid of their assigned cluster, aiming for similarity within clusters and dissimilarity between them. The goal is to create homogeneous clusters with minimal variation within each group.

Steps:

(1) **Initialization:** Randomly select K data points as initial centroids. Choose the number of clusters, K. Randomly initialize the centroids: $\mu_1, \mu_2, ..., \mu_K$.
(2) **Assignment:** Assign data points to clusters: For each data point x_i, calculate the distance to each centroid, typically using Euclidean distance:

$$d(x_i, \mu_j) = \sqrt{\sum_{k=1}^{n}(x_{i,k} - \mu_{j,k})^2}$$

Assign the data point to the cluster with the nearest centroid.

(3) **Update:** For each cluster, C_j, calculate the new centroid:

$$\mu_j = \frac{1}{|C_j|}\sum_{i \in C_j} x_i,$$

where $|C_j|$ is the number of data points in cluster C_j.

(4) **Repeat steps 2 and 3 until convergence:** The objective of the K-means algorithm is to minimize the sum of squared distances between data points and their respective cluster centroids, given by the objective function:

$$J = \sum_{j=1}^{K}\sum_{i \in C_j} ||x_i - \mu_j||^2,$$

where J is the objective function, C_j is the j-th cluster, and μ_j is the centroid of the j-th cluster.

Objective Function: K-means aims to minimize the sum of squared distances between data points and their respective centroids, known as SSE (Sum of Squared Errors).

Choosing K Value: Typically, the elbow method is employed, observing the trend of SSE with varying K values and identifying the "elbow" point as the suitable number of clusters.

Suggested evaluation metrics:

- **Silhouette Score:** Suitable for a general evaluation, as it comprehensively considers both intra-cluster cohesion and inter-cluster separation. It is a good choice when without prior knowledge about the data.
- **Calinski-Harabasz Index (CHI) and Davies-Bouldin Index (DBI):** Useful when we focusing on within-cluster variance and between-cluster differences. Particularly effective when the dataset exhibits clear clustering structures.

4.1.1. *A Radar Signal Sorting Algorithm Based on Data Field Cluster.* [8]

Abstract: All the radar data samples are calculated by normalization, and the potential values of the samples are calculated according to the data field theory. The initial clustering center and the number of clustering centers could be determined without pre-defining the number of clustering.

Methods:

(1) Data field

We build potential function of data field as:

$$F(y) = \sum_{i=1}^{n} f_y(x_i) = \sum_{i=1}^{n} e^{-\frac{d^2(x_i,y)}{2\sigma^2}}$$

The different centers surrounded by potential lines are potential centers. The number of potential centers is the number of clustering centers. The sample closest to potential centers is selected as the initial clustering center.

(2) For the pulse description word (PDW) received by radar, the pulse frequency (RF), pulse width (PW), to the angle of arrival (DOA) is used for sorting. After normalization, We obtain pdw=(rf,pw,doa).

$$rf = \frac{RF(i) - \min RF}{\max RF - \min RF}, pw(i) = \frac{PW(i) - \min PW}{\max PW - \min PW}, doa(i) = \frac{DOA(i) - \min DOA}{\max DOA - \min DOA}$$

The potential value of a radar point m in the data field is:$F_M = \rho\sum_{i=1}^{n} e^{-\frac{d^2(M,pdw_i)}{2\delta^2}}$, the potential value of each point is calculated by $F_{max} \geq F(i,j)$.

And the nearest sample point of the maximum point in the data field is used as the initial clustering center, and the number of maximum values is used as the initial clustering number.

Calculate the distance from each data point to these clustering centers and divide the data samples into the classes represented by the nearest center pdw^i.

For each class, the mean value of the data samples in the newly generated class is recalculated, the obtained mean value is taken as the new clustering center, and the new clustering center is compared with the center obtained last time. If there is no change or minimal

change, the algorithm converges and obtains the result. On the contrary, if the new center changes from the previous center comparison, all the data samples should be re-divided according to the new clustering center. Until the convergence condition of the algorithm is satisfied. In this paper, the convergence condition adopts the criterion function of the sum of square of error:

$$E = \sum_{j=1}^{k} \sum_{i=1}^{n} [(rf^i - {m_{rf}}^j)^2 + (pw^i - {m_{pw}}^j)^2 + (doa^i - {m_{doa}}^j)^2]$$

Results:

In this paper, compared with the traditional clustering algorithm such as K-means clustering algorithm. The number of iterations is 4, while the average number of iterations of k-means clustering algorithm is 13, which greatly reduces the amount of computation, and the correct sorting probability is also increased from 88.91% to 98.59%.

4.1.2. *Radar Signal Sorting Algorithm of a new k-means clustering.* [9]

Abstract: This paper employ K-means as baseline cluttering algorithm and modifies traditional K-means from two aspects:

(1) Ensure the number of cluster K by data field;
(2) Then grey relational degree is used to determine the similarity between data samples.

Method:

(1) Data field
We build potential function of data field as:

$$F(y) = \sum_{i=1}^{n} f_y(x_i) = \sum_{i=1}^{n} e^{-\frac{d^2(x_i,y)}{2\sigma^2}}$$

The different centers surrounded by potential lines are potential centers. The number of potential centers is the number of clustering centers. The sample closest to potential centers is selected as the initial clustering center.

(2) Grey correlation analysis
The initial cluster center is $X_0(j)$, j-th sample is $X_i(j)$, and compare the absolute value difference of the j-th dimension as $\Delta_i(j)$. Then the correlation coefficient is:

$$\zeta_i(j) = \frac{\min_i \min_j \Delta_i(j) + \rho \max_i \max_j \Delta_i(j)}{\Delta_i(j) + \rho \max_i \max_j \Delta_i(j)}$$

Min and Max represent the minimum difference and maximum difference of two levels respectively. Then the $X_0(j)$ and $X_i(j)$ correlation coefficient is the set of all dimensions, and the correlation degree is:

$$\gamma(X_0, X_i) = \frac{1}{N} \sum_{j=1}^{N} \zeta_i(j)$$

If the correlation degree is larger, it means that $X_0(j)$ and $X_i(j)$ is more similar.

Table 1. Comparison of K-means and the proposed algorithm

Methods	iterations	accuracy
K-means	13.1	0.8891
Modified K-means	4	0.9682

From Table 1, the modified k-means algorithm achieves higher accuracy at the much less iterations.

4.1.3. *Radar Signal Sorting Algorithm of K-Means Clustering based on Data Field.* [4]

Abstract: This paper presents a K-means clustering radar signal sorting algorithm based on data field by introducing data field theory, which can determine the number of cluster centers. The proposed method can eliminate noise points and automatically obtain initial cluster center and cluster number, which is suitable for non-cooperative radar without prior information.

Methods:

(1) Data field
We have introduced the data field in [8][9].
(2) Data normalization

$$X_i^{'}(j) = \frac{X_i(j) - \min X_i}{\max X_i - \min X_i}$$

(3) Noise point elimination based on data field: the noise point is far away from the normal data point, and the noise point is sparse. If the potential of this point is very low, it can be considered as a noise point. Algorithm steps: calculate the distance between data points and use the normalized data set; calculate the potential value of data points; find the smallest data point to remove as noise point.
(4) Initial clustering center based on data field: selecting local maximum point of potential function as clustering center;
(5) Clustering using standard k-means algorithm.

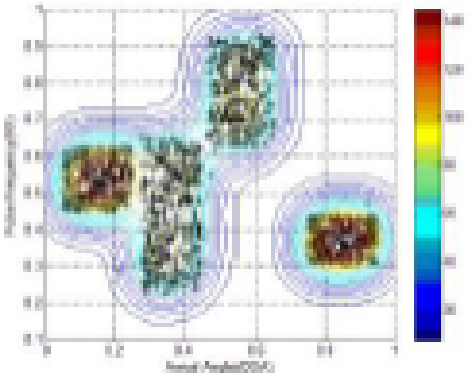

FIGURE 1. the equipotential diagram

Result:

Figure 1 is the cluster center graph. Compared with the traditional K-means algorithm, this algorithm has faster convergence speed and higher accuracy.

4.1.4. *Radar Signal Sorting Based on Ant Colony and K-Means Clustering.* [5]

Abstract:

In this paper, clustering analysis technology is introduced into radar signal selection, ant colony algorithm and K-means are combined to make up for each other's shortcomings, and a new radar signal selection method is proposed.

Methods:

Principle of ant colony algorithm: the main idea is to randomly place the data to be clustered on a two-dimensional plane initially, and then generate some virtual ants on this plane. The behavior of these ants is similar to that of ants in reality. The difference is that they do not observe whether the objects they are carrying are the same as those around them, but judge whether they are similar. In this way, similar data can be grouped together. The average similarity between data and its neighbors is defined as:

$$f(O_i) = \max\left\{0, \frac{1}{S^2}\sum_{C}\sum_{k' \in h(r)'}[1 - \frac{d(O_i, O_j)}{\alpha_i}]\right\}, \alpha_i = \frac{1}{n-1}\sum_{j=1}^{n} d(O_i, O_j)$$

The implementation steps are as follows:

(1) the radar signal parameters DOA, RF and PW are extracted for data standardization;
(2) through ant colony clustering algorithm, K-means clustering algorithm is provided with a correct number of clusters and an appropriate initial centroid;
(3) The initial clustering number and centroid provided by ant clustering algorithm are used as input, and K-means algorithm is used for clustering analysis;
(4) Output clustering results.

Results:

The characteristic of ant colony and K-means clustering algorithm is that it does not need to preset the initial clustering center, that is, it can complete the clustering and sorting of radar signals without the future classification information.

4.2. **DBSCAN.** DBSCAN is a density-based clustering algorithm that groups together data points that are closely packed, while marking points that lie alone in low-density regions as outliers. Unlike K-means, DBSCAN doesn't require specifying the number of clusters beforehand and is robust to noise and outliers.

(1) **Parameter Setup:** Choose Eps (radius) and $MinPts$ (minimum number of points to define a core point).
(2) **Core Point Identification:** For each data point, count the number of data points within the radius Eps. If a data point has at least $MinPts$ data points within its Eps-radius neighborhood, it is considered a core point.
(3) **Cluster Expansion:** Form a cluster around each core point by connecting it to all other core points within Eps. If a non-core point is within the Eps-radius of a core point, it becomes part of the same cluster.
(4) **Noise Identification:** Points that are not core points or part of any cluster are considered noise or outliers.

Parameters

- Eps: The radius within which to search for neighboring points.
- $MinPts$: The minimum number of points required to form a dense region (core point).

Key Concepts:

- **Core Point:** A data point with at least $MinPts$ data points within its Eps-radius neighborhood.
- **Border Point:** A data point within the Eps-radius neighborhood of a core point but with fewer than $MinPts$ neighboring points.
- **Noise (Outlier):** A data point that is neither a core point nor part of any cluster.

Advantages:

- **Robust to Irregularly Shaped Clusters:** DBSCAN can find clusters of arbitrary shapes and is not sensitive to the initial configuration.
- **Automatic Number of Clusters:** DBSCAN does not require specifying the number of clusters in advance.
- **Noise Handling:** DBSCAN can identify and handle noise, classifying outliers as noise points.

Disadvantages:

- **Sensitivity to Parameters:** The performance of DBSCAN can be sensitive to the choice of Eps and $MinPts$ parameters.
- **Difficulty with Varying Density:** DBSCAN may struggle with clusters of varying densities.

Evaluation Metrics:

- DBSCAN does not have a specific objective function like K-means. Evaluation is often performed using external indices such as Rand Index, Adjusted Rand Index, and other cluster validity measures.

4.2.1. *FLIC: Fast Lidar Image Clustering.* [6]

Abstract:
Due to the potentially large amount of Lidar points per scan in ADAS, tailored algorithms are required to identify objects (e.g. pedestrians or vehicles) high precision in a very short time. In this work, the authors proposed an algorithmic approach for real time instance segmentation of Lidar sensor data.

Method:
The author proposed a method for processing raw data from Lidar sensors, specifically focusing on ground extraction and object segmentation. Here's a summary of the key points proposed by the author:

(1) Data Processing Method: The proposed method simplifies the processing of Lidar sensor data by operating directly on the raw two-dimensional representation. This bypasses the computational complexity of processing data in three dimensions, making the method more efficient.

(2) Ground Extraction: To accurately extract the ground plane from Lidar data, the method leverages detailed sensor information to determine the angle of intersection between the laser beam and the surface. This allows for the exclusion of range image values below a certain height, preventing misinterpretation of ground features as objects. The exclusion is determined by the equation:

$$\beta = \arctan\left(\frac{d1 - d2 \cdot \cos\alpha}{d2 \cdot \sin\alpha}\right)$$

(3) Object Segmentation: Object instance segmentation is achieved through clustering within the image space of the range image. By removing ground-associated Lidar measurements and calculating Euclidean distances between adjacent measurements, the method effectively separates Lidar points into clusters representing objects and background points, which helps determine which points belong to the same object based on a predefined threshold. With the calculated threshold, Lidar points are clustered into separate groups representing objects and background. It also performed well with a threshold of 0.8 meters, enabling the clustering of three-dimensional objects up to 114.59 meters vertically and 509.3 meters horizontally.

(4) Map Connections: To address under- and over-segmentation due to missing measurements, the method introduces Map Connections (MC). These connections extend clustering by connecting points in the vertical and horizontal directions, improving segmentation quality without compromising real-time capability. While there isn't a specific equation here, the method involves connecting Lidar points based on their spatial relationships to ensure robust segmentation.Examples of such challenging instances are shown in Figure 1.

Result

The experimental results highlight the effectiveness of the proposed algorithm in achieving real-time instance segmentation of Lidar data with superior performance in terms of both runtime efficiency and segmentation quality compared to existing methods. The incorporation of MCs proved to be a valuable enhancement, contributing to more accurate and robust segmentation results, particularly in challenging scenarios.

(1) When compared to existing methods, the proposed approach exhibited faster execution speeds, with an average frequency of 165 Hz, surpassing the previously fastest algorithm. As can be seen in Figure 8, when adding MCs to the proposed methods, the execution suffers from a slightly longer runtime, while still running at a frequency of 26 to 105 Hz depending on the number of additional MCs. This is still between 2.6 to 10 times faster than the recording frequency of the used sensor.
(2) The segmentation results were evaluated using the SemanticKITTI dataset as ground truth. Despite challenges such as ground plane extraction and varying scene contexts, the proposed method consistently outperformed existing algorithms in terms of segmentation quality. By

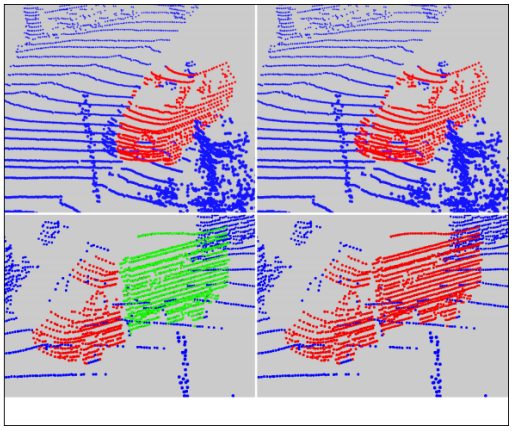

FIGURE 2. Left: Results using only the direct connectivity between neighbouring Lidar points. Right: A single additional MC between every second Lidar measurement. The proposed MCs enable a more accurate segmentation of the car (top) and reduce the over-segmentation of partially occluded objects, as the truck in the bottom images shows.

leveraging the Intersection over Union (IoU) metric, the algorithm demonstrated higher accuracy in separating objects and matching ground truth instances.

Method	IoU_μ (No Ground)	IoU_μ (Ground)	P_μ	$P_{0.5}$	$P_{0.75}$	$P_{0.95}$
Bogoslavskyi et al.	73.93	73.93	59.31	83.75	63.52	13.18
Ours	76.20	72.31	63.73	84.30	67.51	22.03
Ours (*1 MC*)	77.97	73.65	66.68	85.60	70.21	27.19
Ours (*6 MC*)	81.14	75.48	71.92	88.25	74.99	36.05
Ours (*14 MC*)	**84.25**	**76.39**	74.68	**89.75**	**77.61**	40.63
DBSCAN	76.21	72.77	**76.50**	81.54	76.45	**69.25**

FIGURE 3. Comparison of the segmentation quality using the Intersection over Union and the precision average for the algorithms of [3],[1] and variations of the proposed method

4.2.2. *Automatic Vehicle Detection With Roadside LiDAR Data Under Rainy and Snowy Conditions.* [14]

Abstract:

A revised DBSCAN method named 3D-SDBSCAN is raised up to distinguish vehicle points and snowflakes in the LiDAR data in this paper. Adaptive parameters were applied in the revised DBSCAN method to detect vehicles with different distances from the roadside LiDAR sensor.

Method:

The DBSCAN in 2D space did not consider the point distribution in different heights, so the number of snowflakes were accumulated in 2D space, which caused in sonwy weather, the clustering results generates wrong vehicle counts information.

However, those snowflakes were in different heights (z-axis), with most snowflakes at the height of the roadside LiDAR sensor, as illustrated in Fig. 3. Unlike snowflakes distribution, vehicle points were in a limited narrow height range from the ground surface because the height of the vehicle was fixed, and at a higher density than snowflakes. Therefore, the point distribution in z-axis (height) could be used to improve the accuracy of vehicles clustering.

The DBSCAN algorithm requires 2 parameters :

- **epsilon**, which specifies how close points should be to each other to be considered a part of a cluster

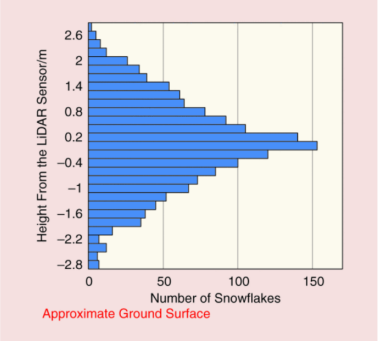

FIGURE 4. Height distribution of snowflakes

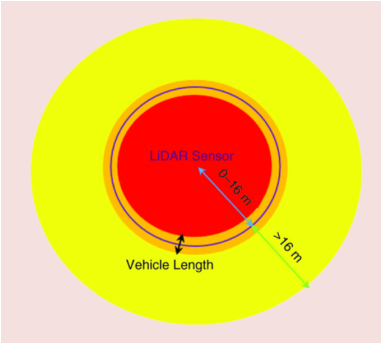

FIGURE 5. DBSCAN area division

- **minimum number of points (M)**, which specifies how many neighbors a point should have to be included into a cluster.

The selected epsilon is 1.75 m in this paper. The practice shows that when the weather is snowy, the M-10 used in the previous research [13] could not identify the vehicle correctly since the number of snowflakes can be easily higher than 10 within an area of 1.75 * 1.75 + 1.75m3. Then the M is increased to 70 until the vehicle can be identified.

Meanwhile, the author find that using same M in different distance cannot identify exactly.Therefore, in this study, the whole space was divided into three parts (0-16 m and more than 16 m and the an overlapping ring to avoid that if the vehicle was crossing the border of 16 m, points were cut into two parts.), as shown in fig .4.

Different M values were assigned to the different subareas for applying the DBSCAN algorithm.The whole 3D space was divided into subareas based on the distribution of snowflakes. Different DBSCAN parameters were used in different subareas. This revised DBSCAN algorithm is named as section-based DBSCAN in 3D space (3D-SDBSCAN).

Result

the performance of the 3D-SDBSCAN was quantified by two metrics: true positive rate (TPR) and false detection rate (FDR). TPR is defined by

$$TPR{=}TDV/AV,$$

Where AV is the actual number of vehicles, TDV is the number of truly detected vehicles obtained from data processing algorithm.

FDR is defined by

$$FDR{=}SV/(SV{+}TDV),$$

Where SV is the number of false vehicles created by snowflakes or raindrops.

Result

	Normal Weather		Rainy Weather		Snowy Weather	
Actual number of vehicles extracted from LiDAR videos (AV)*	35405		23642		5413	
Vehicle clustering method	DBSCAN	3D-SDBSCAN	DBSCAN	3D-SDBSCAN	DBSCAN	3D-SDBSCAN
Number of truly detected vehicles (TDV)	35 361	35 354	23 399	23 407	3125	5347
Number of false vehicles created by snowflakes/raindrops (SV)	NA	NA	16	3	48 450	35
TPR	99.87%	99.86%	98.97%	99.01%	57.73%	98.78%
FDR	NA	NA	0.07%	0.01%	93.94%	0.65%
Average detection range (m)	38.7	38.1	36.3	37.5	19.5	37.4

*The ground truth vehicles were also manually labeled.

FIGURE 6. Comparison of 3D-SDBSCAN and DBSCAN.

The evaluation results are shown in fig.5. The results showed that during normal or rainy weather, the difference between DBSCAN and 3D-SDBSCAN was small. When it was snowing, the 3D-SDBSCAN can provide better accuracy than the original DBSCAN.

4.2.3. *A Multi-Stage Clustering Framework for Automotive Radar Data.* [11]

Abstract:
In this article, the DBSCAN algorithm is evaluated using different distance metrics based on an $x/y/vr/t$ grid. Epsilon values for the individual parameter dimensions are optimized with the assumption that a single fixed set should suffice. Equally, real world road users' extents are expected to remain in similar ranges of values. To enable static epsilon values, beam dilation is counteracted using a range-dependent value for $Nmin$. Splitting the clustering process into two separate DBSCAN steps, should relax the requirements for the clustering algorithm to the point where additional information can be extracted from the first clustering results. The extra information is then used to merge clusters that would have required extensive parameterizations in single step clustering.

Method:
the proposed framework consists of three components that are discussed individually in this section:

- **Detection filtering**
- **Point clustering**
- **Cluster merging**

In the step of clustering, the radar points remaining after the filtering process are associated by utilizing a slightly modified version of a DBSCAN algorithm. The main differences to conventional DBSCAN is an adaptive number of minimum points Nmin(r) required to form a cluster core point, where r is the range, i.e., distance between detection and sensor. This adjustment is based on the fact that remote objects have a smaller maximum number of possible reflections (from a constant false alarm rate (CFAR) detector) due to range-independent angular resolution. Since the physical extents of road users do not change, this article utilizes only the minimum point property to account for remote objects:

$$N_{\min}(r) = N_{\min,50\mathrm{m}} \cdot \left(1 + \alpha_r \cdot \left(\frac{\mathrm{clip}(r, 25\mathrm{m}, 125\mathrm{m})}{50\mathrm{m}} - 1\right)\right).$$

The equation above has two tuning parameters: $N_{\min,50\mathrm{m}}$ and α_r which represent a minimum point baseline at 50 m and the slope of the reciprocal relation. To avoid very low or high numbers for $N_{\min}(r)$ the range r is clipped to values between 25 m and 125 m.

This paper discusses different neighborhood criteria and their corresponding distance metrics used in point clustering.

1. The first neighborhood criterion combines the differences $\Delta\cdot$ of all four variables:

$$\Delta x < \in_x \wedge \Delta\, y < \in_y \wedge \Delta\, v_r < \in_{v_r} \wedge \Delta\, t < \in_t.$$

2. The second method aims to achieve full rotational invariance by utilizing the Euclidean distance of the spatial components:

$$\sqrt{\Delta\, x^2 + \Delta\, y^2} < \in_{xy} \wedge \Delta\, v_r < \in_{v_r} \wedge \Delta\, t < \in_t.$$

3. The third criterion combines both spatial components and the radial velocity in a single Euclidean distance:

$$\sqrt{\triangle x^2 + \triangle y^2 + \frac{1}{\in_{v_r}^{2\prime\prime}} \cdot \triangle v_r^2} < \in_{xyv_r} \wedge \triangle t < \in_t.$$

In the step of cluster merging(second clustering step), only those points are considered that have been assigned a cluster label in the previous step. The key idea for this step is that a cluster contains more information than its individual detections on their own. Therefore, cluster merging will be based on different DBSCAN parameters than the first clustering step. Further, it can be seen that background noise is well suppressed by the filtering and clustering algorithms.

Result

For this article the V-measure [10] is chosen as a suitable score function as it combines several of the named requirements. The V-measure is a combined score based on two intuitive clustering criteria, homogeneity and completeness. Homogeneity is maximal when a predicted cluster only contains points from a single ground truth cluster. Contrary to that, completeness aims to assign all points from a single ground truth cluster into a single cluster prediction. Both are based on the conditional entropy of predicted clusters K given ground truth clusters C and vice versa:

$$\text{Homogeneity} = 1 - \frac{H(C|K)}{H(C)}, \tag{3.1}$$

$$\text{Completeness} = 1 - \frac{H(K|C)}{H(K)}, \tag{3.2}$$

$$\text{with } H(A|B) = -\sum_{a\in A}\sum_{b\in B} \frac{n_{a,b}}{n} \cdot \log \frac{n_{a,b}}{n_b},$$
$$\text{and } H(A) = -\sum_{a\in A} \frac{n_a}{n} \cdot \log \frac{n_a}{n}.$$

$H(A)$ and $H(A|B)$ represent the entropy and the conditional entropy, respectively. A and B can be substituted for C and K in the required order to match Eqs. 3.1 and 3.2 with n being the total number of samples, n_a and n_{ab} being the number of samples belonging to a, or a and b at the same time. The V-measure V_1 is the harmonic mean of homogeneity and completeness:

$$V_1 = 2 \cdot \frac{\text{Homogeneity} \cdot \text{Comp1eteness}}{\text{Homogeneity} + \text{Comp1eteness}}. \tag{3.3}$$

In Fig. 6 the resulting best parameter setting is compared to the baseline parameterization, the ground truth, and the raw data on a short measurement sequence for visual comparison. It can be seen that how better clustering settings help to retain the contours,

In this paper, the assignment of radar detections to clusters is an early step in a traditional classification processing chain. Subsequently, a DBSCAN algorithm with specialized distance metric is used to obtain a first clustering of the scene. An adaptive number of minimum neighbors for cluster formation in DBSCAN provides good results regardless of distance. When combining all individual components, the overall test results are improved even further and clearly outperform all baseline experiments.

4.2.4. *Grid-based DBSCAN for clustering extended objects in radar data.* [7]

Abstract:
This paper presents an algorithm based on DBSCAN modified to deal with the non-equidistant sampling density and clutter of radar data while maintaining all its prior advantages. In addition, it is robust against clutter and a non-equidistant sampling density. With its optional input parameter, this algorithm is flexible and can be adjusted not only for the sensor but also for the desired cluster size or separation resolution of two close objects.

Method:

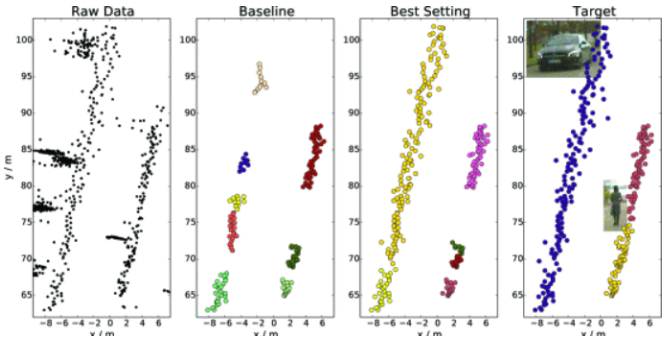

FIGURE 7. Different processing stages of a 20 s radar measurement sequence with two road user trajectories, a car (left) and a pedestrian (right). Beside raw data and target (ground truth) view, the results from the baseline clustering and the best estimated cluster setting are depicted. Each cluster instances has its own color. Even in the ground truth, the pedestrian instance is labeled as two clusters, because it was not present in the radar data for almost 1 s

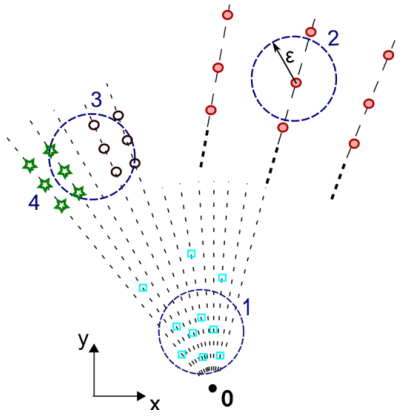

FIGURE 8. Effects of original DBSCAN algorithm on two close objects (green, black), one object far away (red) and clutter (cyan)

Applying DBSCAN on a dataset with a non-equidistant sampling density induces some disadvantages which are shown in Fig. 7:

(1) **Presence of clutter**: There is a high sampling density in Cartesian coordinates close to the sensor, cause even widely distributed clutter is clustered to objects easily.
(2) **Limited Range**: Objects which are far away from the sensor aren't clustered because the same observations (k) in different range.
(3) **Separation resolution**: The ability to separate two objects in the azimuth direction varies significantly with the range.
(4) **Object orientation**:the algorithm prefers small and long objects in the azimuth direction and discriminates objects in the range direction.

The modified grid-based DBSCAN algorithm is based on neighborhood relations and can use the r- and θ-cell information as a grid.The grid based DBSCAN uses only the ratio c to calculate the spatial sampling density between the radial and angular distance for each point:

$$c = \frac{r_{i,j}}{2\Delta r} \left(\sin(\theta_{i,j+1} - \theta_{i,j}) + \sin(\theta_{i,j} - \theta_{i,j-1})\right)$$

with:

i, j	index of grid in r/θ-direction
$r_{i,j}$	radial distance
Δr	radial resolution (constant)
$\theta_{i,j}$	azimuth angle

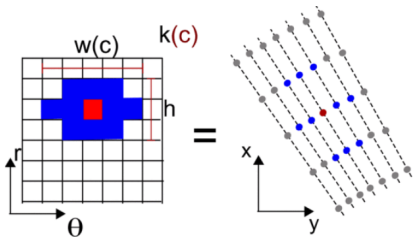

FIGURE 9. Calculation of the density criterion using grid-based DBSCAN (left), with the search area specified by the parameters h and w and the adaptive threshold k. The equal representation in Cartesian coordinates is on the right side

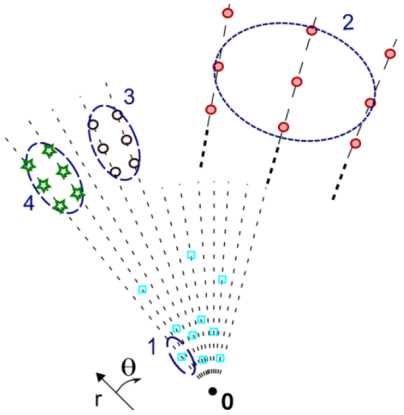

FIGURE 10. Effects of the grid-based DBSCAN algorithm on two close objects (green, black), one object far away (red) and noise (cyan)

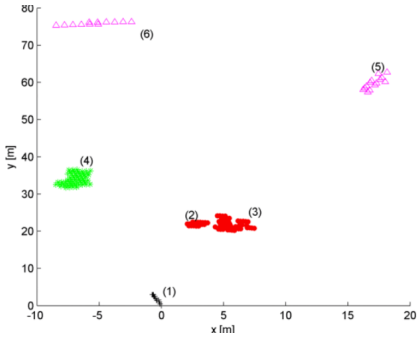

FIGURE 11. Clustering results with original DBSCAN ($\varepsilon = 1\text{m}$, $k = 6$) showing outliers (pink triangle) and 3 clusters (red, green, black) for clutter (1), a pedestrian (2) next to a car (3), two other vehicles (4–5), and a barrier (6).

The advantages of grid-based DBSCAN is a simple comparison in both directions to determine all observations inside the search area. Compared to the calculation of the Euclidean distance between all observations, it is significantly less time-intensive. Furthermore, the data format can be simple integers, compared to float values in DBSCAN.

It can be seen that the several problems mentioned above (Presence of clutter, Limited Range, .etc) can be solve as shown in fig. 9.

Result The results(fig .10 and fig .11) show that Grid-based DBSCAN is able to cluster the car (5) and the barrier (6) properly. Since the search area contains only one cell in each direction, even objects in radial direction (5) and thin objects can be clustered (6). However, DBSCAN is not able to separate the pedestrian (2) next to the car (3) and the two form a cluster.

4.2.5. *Real-Time clustering and LiDAR-camera fusion on embedded platforms for self-driving cars.* [12]

Abstract:

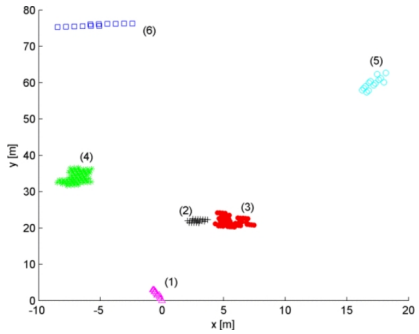

FIGURE 12. Clustering results for grid-based DBSCAN ($f = 2$) showing outliers (pink triangle) and 5 clusters (red, green, black, blue, cyan) for clutter (1), a pedestrian (2) next to a car (3), two other vehicles (4–5), and a barrier (6).

Input: pointcloud $\mathcal{P}$, W, $maxN$, $maxD$, $minCluster$
Output: $clusters$
Data: $nList(p_j) = \emptyset \ \forall p_j \in \mathcal{P}, clusters = \emptyset$, $dir = next$

forall $p_j \in \mathcal{P}$ **do**
  for $i = 0$ *to* $2W$ **do**
    if $i == W$ **then**
      $dir = prev$;
    $p_k = nextNeigh(\mathcal{P}, p_j, i, dir)$;
    $d = euclideanDistance(p_j, p_k)$;
    if $d < maxD$ **then**
      $nList(p_j) = nList(p_j) \cup p_k$;
      if $i < W$ ***and*** $len(nList(p_j)) > \lfloor \frac{maxN}{2} \rfloor$ **then**
        $i = W$;
      if $i \geq W$ ***and*** $len(nList(p_j)) > maxN$ **then**
        break;
forall $p_j \in \mathcal{P}$ **do**
  $\mathcal{P} = \mathcal{P} \setminus p_j$;
  forall $p_i \in nList(p_j)$ **do**
    $\mathcal{P} = \mathcal{P} \setminus p_i$;
    forall $p_k \in nList(p_i)$ ***and*** $\notin nList(p_j)$ **do**
      $nList(p_j).enqueue(p_k)$;
  if $len(nList(p_j)) > minCluster$ **then**
    $clusters = clusters \cup nList(p_j)$;
return $clusters$;

FIGURE 13. Algorithm 1: WBLC

In this paper, a smarter method is presented for real-time clustering, customizing the DBSCAN algorithm for LiDAR point clouds, without losing accuracy on cluster detection.

Method:

The clustering algorithm is performed only on two dimensions, i.e. latitude and longitude, dropping height. This choice follows by assuming objects are never placed one on top of another.

The search of the neighboring points is then parallelized on a GPU, creating a thread for each point. Moreover, the exploration is bounded to a user-defined threshold, checking at most 2W points for each object in the cloud. This is a reasonable choice for a LiDAR sensor because of the nature of the output data, where cloud points are semi -ordered by horizontal angle. Exploiting this feature, the search for neighbouring points can be limited to adjacent angles.

The proposed Window-Based Lidar Clustering (WBLC) is detailed in Algorithm 1(fig .12), receiving a 2D point cloud as input, and producing the identified clusters as output. The algorithm is divided into two parts: the searching for neighbors and the merging of clusters.

Result

Method	Platform	Dataset	Time (ms)
DBSCAN [17]	i7-6700 @3.4GHz, GTX 1060, Velodyne HDL-32E	N/D	26411.6
kd-tree DBSCAN [17]	i7-6700 @3.4GHz, GTX 1060, Velodyne HDL-32E	N/D	779.142
Range based DBSCAN [17]	i7-6700 @3.4GHz, GTX 1060, Velodyne HDL-32E	N/D	46.14
FRIBS [13] 32-beams	i7 4770K @3.5 GHz, Velodyne Puck VLP-16	2500 clouds (15000 pts)	2.6 ± 0.5
FMP3DPSB [15]	i7-4790K @4GHz, Velodyne HDL-64E S2	3000 clouds (130000 pts)	15
FPCL [14]	i7-4790K @4GHz, Velodyne HDL-64E S2	4000 clouds (130000 pts)	36
FDDMGC [16]	i7-6700 @3.4 GHZ, Velodyne HDL-32E	1 cloud (60000 pts)	11.01
this paper	Tegra Jetson TX2, Velodyne VLP-32C	2963 clouds (60000 pts)	14.60 ± 2.75
this paper	AGX Xavier, Velodyne VLP-32C	2963 clouds (60000 pts)	5.11 ± 0.98
this paper	i7-7700 @3.60GHz, GTX 1060-6GB, Velodyne VLP-32C	2963 clouds (60000 pts)	**2.29 ± 0.43**

FIGURE 14. Enter Caption

Method	Dataset	Precision	Recall
FRIBS [13] 64-beams	Dataset by [16]	0.69	0.71
FMP3DPSB [15]	Dataset by [16]	0.87	0.84
FDDMGC [16]	Dataset by [16]	0.94	0.96
this paper	Our Dataset	0.923	0.928

FIGURE 15. Clustering accuracy performances.

To evalaute the complexity of the algorithm, consider that, in the first part, at most 2 W points are checked for each of the —P— points, leading to a complexity of O(W—P—). In the second part, at most maxN points are checked for each of the —P— points, leading to O(maxN—P—). Since maxN is selected to be smaller than W, the overall complexity is O(W—P—).

The proposed algorithm outperforms all the other methods on general purpose desktops and discrete GPUs, and it is the only one that is able to provide real-time performance even on embedded platforms (TX2 and Xavier) as shown in fig .13.

As shown in fig .14,this algorithm has a significantly higher accuracy than the fastest competing method (FRIBS [1]). Meanwhile, it also has a comparable accuracy to the most precise method (FDDMGC [2]), which however runs five time slower.

REFERENCES

[1] Igor Bogoslavskyi and Cyrill Stachniss. “Fast range image-based segmentation of sparse 3D laser scans for online operation”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 163–169. DOI: 10.1109/IROS.2016.7759050.

[2] Patrick Burger, Benjamin Naujoks, and Hans-Joachim Wuensche. “Fast Dual Decomposition based Mesh-Graph Clustering for Point Clouds”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 1129–1135. DOI: 10.1109/ITSC.2018.8569999.

[3] Martin Ester et al. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Knowledge Discovery and Data Mining*. 1996. URL: https://api.semanticscholar.org/CorpusID:355163.

[4] Liu Y Feng X Hu X. “Radar signal sorting algorithm of k-means cluster in g based on data field”. In: *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*. 2017, pp. 2262–2266.

[5] Bin Chen Guixi Zhao Luqin Luo. “Radar Signal Sorting Based on Ant Colony and K-Means”. In: *Radar Science and Technology*. Vol. 42. 05. 2019, pp. 79–82.

[6] Frederik Hasecke, Lukas Hahn, and Anton Kummert. “Flic: Fast lidar image clustering”. In: *arXiv preprint arXiv:2003.00575* (2020).

[7] Dominik Kellner, Jens Klappstein, and Klaus Dietmayer. “Grid-based DBSCAN for clustering extended objects in radar data”. In: *2012 IEEE Intelligent Vehicles Symposium*. 2012, pp. 365–370. DOI: 10.1109/IVS.2012.6232167.

[8] Jin Hu Ran Jia. “A Radar Signal Sorting Algorithm Based on Data Field Cluster”. In: *Modern Defense Technology*. Vol. 45. 04. 2017, pp. 124–129.

[9] Houpei Xia Ran Zhang. “Radar Signal Sorting Algorithm of a new k-means clustering”. In: *Modern Defense Technology*. Vol. 43. 06. 2015, pp. 136–141.

[10] Andrew Rosenberg and Julia Hirschberg. "V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure". In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Ed. by Jason Eisner. Prague, Czech Republic: Association for Computational Linguistics, June 2007, pp. 410–420. URL: https://aclanthology.org/D07-1043.

[11] Nicolas Scheiner et al. "A Multi-Stage Clustering Framework for Automotive Radar Data". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 2060–2067. DOI: 10.1109/ITSC.2019.8916873.

[12] Micaela Verucchi et al. "Real-Time clustering and LiDAR-camera fusion on embedded platforms for self-driving cars". In: *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*. 2020, pp. 398–405. DOI: 10.1109/IRC.2020.00068.

[13] Jianqing Wu and Hao Xu. "An Automatic Procedure for Vehicle Tracking with a Roadside LiDAR Sensor". In: 2018. URL: https://api.semanticscholar.org/CorpusID:67069728.

[14] Jianqing Wu et al. "Automatic Vehicle Detection With Roadside LiDAR Data Under Rainy and Snowy Conditions". In: *IEEE Intelligent Transportation Systems Magazine* 13.1 (2021), pp. 197–209. DOI: 10.1109/MITS.2019.2926362.