Object Detection in Lidar Scan PCD Files using DBSCAN and Random Forest

Hailin Hu

College of Engineering and Applied Science, University of Colorado Boulder

CSCA 5622 Supervised Learning Final Project

31 May 2024

1. Abstract

This study investigates object identification in Lidar scan PCD files using DBSCAN (Density-Based Spatial Clustering of Applications with Noise)[1] and Random Forest algorithms to enhance autonomous vehicle navigation systems. Utilizing the ApolloScape 3D Lidar Object Detection and Tracking dataset[2], the research applies rigorous data cleaning, feature extraction, and exploratory data analysis (EDA) to optimize the classification models. Methods include dynamic and static feature analysis and advanced model evaluation techniques such as *MPE* based on *IoU*. Results demonstrate improved object identification accuracy, contributing significant insights into 3D object detection methodologies for autonomous driving applications.

Keywords: Object Identification, Lidar Scan, PCD Files, DBSCAN, Random Forest, Autonomous Vehicles, Feature Engineering, Machine Learning.

2. Introduction

In the field of automatic vehicle, objects detection and identification are crucial topics. Many approaches are available based on recent technology. Figure 1 shows a rendering of PCD (Point Cloud Data) files. This type of file is composed of points' location (namely describing coordinates of X, Y and Z), popular in object identification, which is the main data source of this project to identify and classify objects.

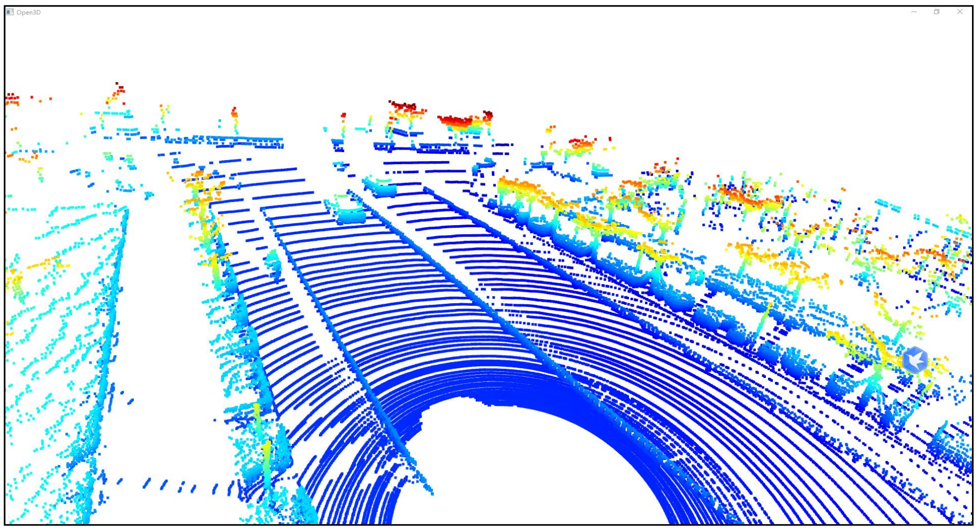

Figure 1: single frame of PCD data rendering, sourced from ApolloScape

The main type of this project is supervised learning, involving algorithms of DBSCAN (Density-Based Spatial Clustering of Applications with Noise), an unsupervised clustering algorithm used to group points in the PCD files into distinct objects based on their density; Random Forest, a supervised classification algorithm used to train a model on the extracted features of the objects identified by DBSCAN and classify these objects in the test data.

The task of this project is object identification and classification in Lidar scan PCD files. By integrating algorithms above, the project aims to accurately identify and classify objects from Lidar scan data, providing comparison and contributing insights into 3D object detection and classification methodologies for autonomous driving applications.

3. Data Source Description

The dataset used for this project is sourced from the ApolloScape 3D Lidar Object Detection and Tracking dataset, available on GitHub (see Reference). This dataset includes LiDAR scanned point clouds with high-quality annotations collected in Beijing, China, under various lighting conditions and traffic densities. The data captures complex traffic flows involving vehicles, cyclists, and pedestrians.

- **data structure[3]**

1) **train.zip**: Training data in PCD (Point Cloud Data) and bin file formats, captured at 2 Hz.
2) **detection/tracking_train_label.zip**: Labelled data for object detection and tracking, with each file containing 1-minute sequences at 2 fps. Labels include `frame ID`, `object ID`, `object type`, `position (x, y, z)`, `dimensions (length, width, height)`, and `heading`.
    a. **Object Types**: 1 - small vehicles, 2 - big vehicles, 3 - pedestrians, 4 - motorcyclists and bicyclists, 5 - traffic cones, 6 - others.
    b. **Units**: Position and bounding box dimensions are in meters; heading is in radians.
3) **pose.zip**: Lidar pose data including `frame index`, `lidar time`, `position (x, y, z)`, and `quaternion (x, y, z, w)`. Positions are in absolute coordinates.

Because the test dataset provided by the origin source does not explicitly show the true label of test data, for this project, the `detection_train_pcd_1`, `detection_train_pcd_2`, and `detection_train_pcd_3` files were used, along with the corresponding pose and label data, to create the training and testing datasets. The whole frames consist of 53 sectors with multiple sequential frames, each of which is composed of approximately 100,000 points in space.

4. Data Preparation and Feature Engineering

- **Overview**

Data preparation for this task is a crucial step in preparing the dataset for model training and prediction. The main goals are to reduce noise, eliminate irrelevant data points, and extract meaningful features for model training and testing in objects detection task.

- **Steps and Justification**

1) Files loading and transformation of coordinate system

In this step, I load the frames of PCD, label and pose files of the same frame sequentiallywith open3d library[4]. According to the documentation of ApolloScape, pose files use global coordinate system and PCD and label files utilize relative coordinate system. I transform the files into global coordinate system for further feature engineering.

2) Points cloud object extraction

For training data, the origin dataset has been labeled and thus I can directly extract objects via information in label files, including `object_id`, `object_type`, `position_x`, `position_y`, `position_z`, `object_length`, `object_width`, `object_height`, and `heading`. Based on the location, size and direction of certain objects, which compose of bounding box of certain object, I extract labeled objects into different point cloud sets.

For testing data, however, since I do not know object information before I identify and map the object in sequential frames, finding an efficient and accurate way for object detection is crucial. For this task, DBSCAN algorithm is proper to identify objects of spacial point clusters.

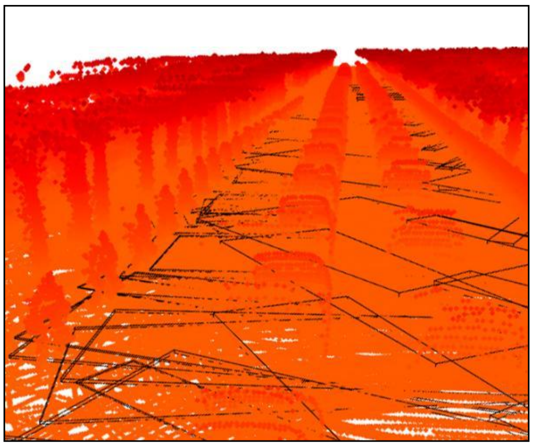

Figure 2: Detected objects using DBSCAN algorithm without ground noise filtering, whose bounding boxes are closed to ground with low height.

However, if inspecting the rendering image of PCD provided in Figure 2, it is easy to notice that noise from irrelevant objects, especially ground, makes detected objects much less accurate, most of which have low height and close to ground.

This indicates specific approach is required to avoid noise from ground scanned points. In this problem, I simply filter points below certain Z coordinate.

After this, I then implement DBSCAN algorithm with proper parameters of `eps` and `min_samples` to identify objects in the form of points cloud.

3) **Feature vector extraction**

To distinguish different type of objects, I introduce feature vectors. It is easy to learn that there are two types of physical features: time-based and non time-based features.

Non time-based features, namely static features, including `length`, `width`, `height`, `volume`, `surface_area`, `xy_area` (projection of XY plane), `density` (number of points per unit volume, divided by squared distance), and `pca_z_cosine` (cosine of angle of PCA axis and Z axis), which capture the physical dimensions and spatial characteristics of the objects without identify sequence of frames.

Time-based features, however, are dynamic, which change correspondingly when objects move in space and time-axis, including `principal_camera_cosine` (cosine of angle of PCA axis and camera moving direction), `velocity_camera_cosine` (cosine of angle of object moving direction and camera moving direction), `velocity_magnitude`, `acceleration_magnitude`, `angle_change`, and `angle_speed`. These features capture the motion and orientation of the objects relative to the camera.

For extracted objects of testing data, however, works of feature vector involve not only object classification itself, but mapping object in adjacent frames to calculate time-related dynamic features. Since I do not know the exact mapping relationship in testing data, I only calculate static feature vectors of testing dataset in this feature extraction procedure and further calculate and map objects in further steps.

- **Analysis and Summary**

Challenges: As articulated above, in the procedure of data preparation and feature extraction, I encounter challenges of

1) Noise excluding problem including ground reflection points and other noise in origin data in testing data processing.
2) Object mapping in the testing phase to calculate motion vectors pose significant challenges due to the lack of known relationships between frames.

Strategy: As I implement in the specific steps above, I address the problem by

1) Introducing Z-axis threshold to filter noise points from ground reflection, and implementing DBSCAN cluster algorithm to cluster points objects from testing dataset.
2) Processing testing data into static feature vectors, temporarily ignoring sequential relationship related to time sequence.

Summary: As the Figure 3 shows, I have processed the origin data of PCD with points of X, Y, and Z coordinates into static training and testing feature vectors, and dynamic training vectors for further operation.

5. Exploratory Data Analysis (EDA)

- **Overview**

The physical features I extract above may involve multilinear regression or collinearity, since I simultaneously calculate some features may correlated with each other, such as `velocity_magnitude`, and `acceleration_magnitude`, or `length`, `width`, `height`, and `volume`. Therefore, introduction of Exploratory Data Analysis (EDA) is essential.

The EDA aims to investigate the relationships between static and dynamic features, and between these features and the target labels (object types). This helps in understanding the underlying patterns in the data and in preparing it for the Random Forest model.

- **Analysis steps of EDA**

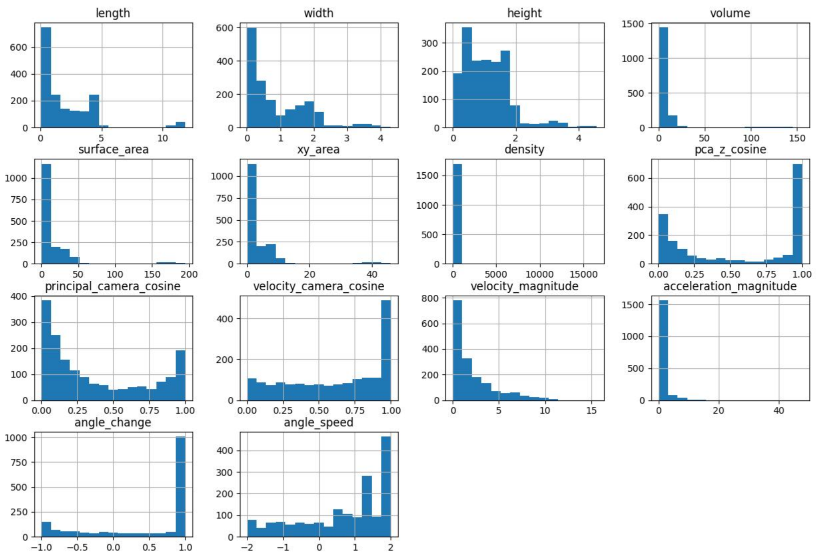

Figure 3: Visualization of histograms of different feature

1) Histograms

Figure 3 displays histograms for various features of objects, showing data distribution trends and characteristics essential for data preprocessing and modeling decisions. Analysis is as follows:

a. **`length, width, height, volume, surface_area, xy_area`**: These features predominantly exhibit right-skewed distributions, indicating that most objects have smaller dimensions with occasional larger outliers. This skewness suggests a concentration of smaller objects in the dataset.

b. **`density`**: This feature shows most of data has the same density, but there are some outliers with extremely large values.

c. **`pca_z_cosine`**: A concentration of values near 1.0 suggests high alignment of many objects along the PCA Z-axis, indicating little variation in this orientation.

d. **`principal_camera_cosine, velocity_camera_cosine`**: The data clusters at higher values indicate that objects generally maintain a consistent orientation relative to the camera's perspective

e. **`velocity_magnitude, acceleration_magnitude`**: These features are extremely right-skewed, showing that most objects move at low speeds and accelerations, with very few exhibiting higher dynamics.

f. **`angle_change`**: The bi-modal distribution around -1 and 1 highlights significant directional changes in some objects, potentially indicating rotational movements.

g. **`angle_speed`**: This shows data primarily centered around zero with a small number of instances having higher absolute values, up to 2. This distribution suggests that most objects have minimal rotational movement, with a few exceptions exhibiting faster spins or turns.

2) Pair Plot

Figure 4 shows a comprehensive pair plot (scatter matrix) with histograms along the diagonal, illustrating the relationships among various features in a dataset. The aim of this type of visualization is recognition of collinearity, because scatter plots that show a clear linear trend (either positive or negative) indicate strong collinearity between those features.

a. **`length, width, height, volume, surface_area, xy_area`**: It seems correlated or collinear in features that measure different dimensions of the same objects (like length, width, height, and volume), because the volume or other feature like surface area is calculated by the length, width and height. But it is normal for length, width and height themselves appears to be correlated or collinear, because they have very close value distribution for most objects. Therefore, further analysis and implementation of this kind of feature may be required.

b. **`density`**: This feature seems to be not related to other features, showing a line paralleled with X or Y axis.

c. **`angle_speed, angle_change`**: These two features show correlated with each other, which means

d. **`Others`**: They do not explicitly show collinearity because the points are randomly distributed in these plots.

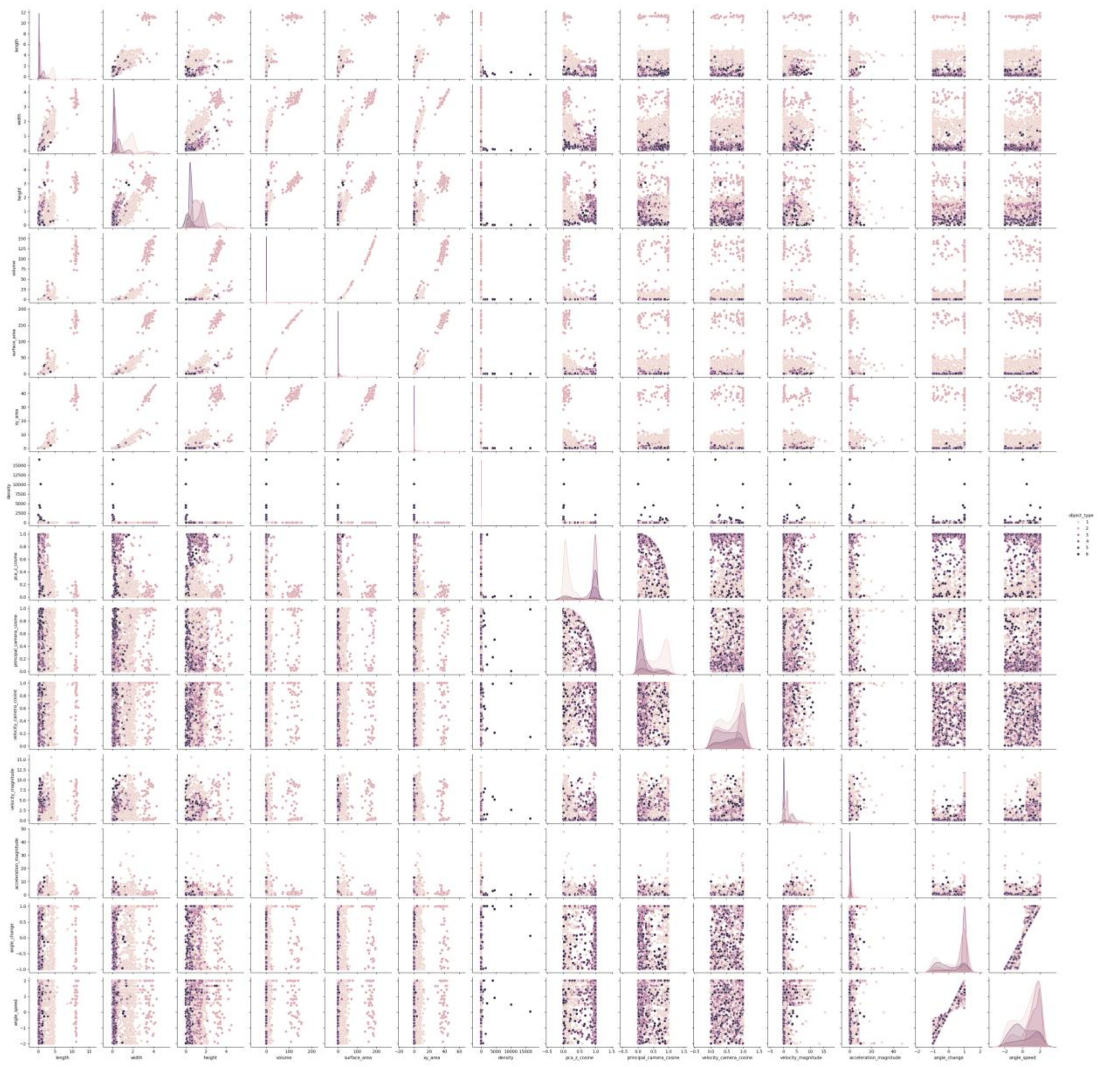

Figure 4: Visualization of pair plot of features

3) **Correlation matrix**

Figure 5 provide correlation matrix with the correlation coefficients between various features of a dataset, which is crucial in understanding relationships, predicting potential multicollinearity, and guiding feature selection for modeling. Analysis is as follows:

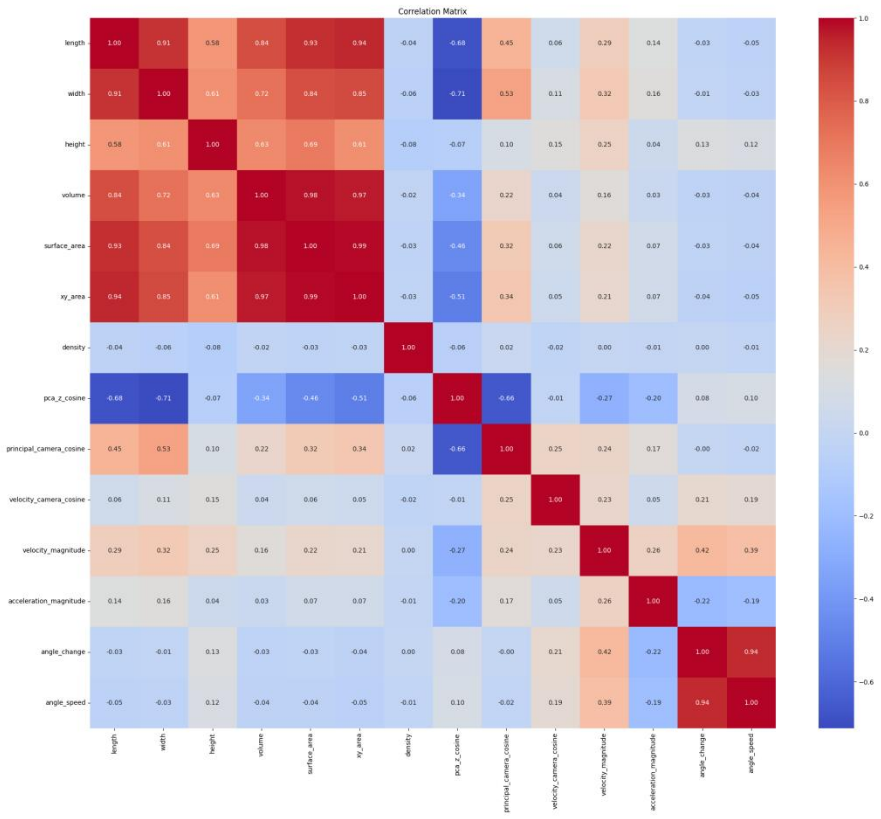

Figure 5: Visualization of Correlation matrix of features

a. **`length, width, height, volume, surface_area, xy_area`**: These features show very high correlations with each other (most coefficients > 0.8). This suggests these features are measuring similar aspects of the objects and may contribute redundantly when used together in predictive modeling.
b. **`density`**: This feature displays very low correlations with all other features, indicating it does not vary systematically with changes in size or shape parameters of the objects.
c. **`angle_speed, angle_change`**: These two features show correlated with each other, which means
d. **`pca_z_cosine`**: This feature shows moderate to strong negative correlations with `length`, `width`, and `height` (-0.68, -0.71, -0.37 respectively), indicating that as these dimensions increase, the alignment with the PCA Z-axis decreases.
e. **`principal_camera_cosine, velocity_camera_cosine`**: They have a strong correlation (0.66), suggesting redundant information about the orientation of objects relative to the camera.

f. **`velocity_magnitude, acceleration_magnitude`**: They show some degree of correlation (0.39), indicating that faster objects tend to accelerate more.
g. **`angle_speed, angle_change`**: They have a high correlation (0.94), which is expected as these both describe aspects of angular movement.

- **Conclusions of EDA**

1) **High Collinearity Among Dimensions**: Features like `length`, `width`, `height`, `volume`, `surface_area`, and `xy_area` show high correlation (>0.9 in most cases). These indicate redundancy as they likely measure related aspects of object size and shape.
2) **Negligible Variance in Some Features**: The histograms indicate some features like `density` have little variance across different observations, appearing as vertical lines in scatter plots which suggests no significant variability or correlation with other features.
3) **Dynamic Features Correlation**: Features describing movement dynamics, such as `angle_change` and `angle_speed`, show high correlations (0.94), which suggests redundancy.

- **Further implementation**

1) **Exclude Redundant Features**: Based on the correlation matrix, features such as `angle_speed` can be considered redundant if `angle_change` is included. Similar decisions can be made for other highly correlated pairs.
2) **Exclude Low Variance Features**: If features like `density` do not vary much across the dataset as suggested by their histograms and scatter plots, they might not be useful for models that rely on variability to distinguish between outcomes.

This comprehensive analysis highlights the importance of understanding feature relationships and distributions before proceeding with model building. By applying dimension reduction, excluding redundant or low variance features, and normalizing data, you can significantly improve the model's performance and interpretability.

6. Model Building and Analysis

- **Overview**

The model section addresses multiple advanced techniques including feature engineering, hyperparameter tuning, multiple models for tracking and cost matrix and object mapping.

- **Model building steps**

1) **Feature engineering and hyperparameter tuning**

As the EDA above shows, after I extract feature vectors, several features appear to be collinear or correlated. Based on EDA conclusion, I exclude redundant features such as `angle_change` and low variance features like `density` which does not vary much across the dataset as suggested by their histograms and scatter plots. Besides, I exclude some highly collinear size measurement features like `surface_area` and `xy_area`.

2) **Multiple models for tracking**

In the context of object tracking, since I have extracted static and dynamic features respectively, I build two models to classify in different occasions. The first model, a static model, is typically trained to identify objects based solely on static features extracted from each frame. In initial identification process, static model is trained to predict label of each objects clustered in each frame. A Random Forest model is chosen due to its superior interpretability and lower risk of overfitting in high-dimensional space compared to models like SVM.

The choice of Random Forest is further justified as it can handle various types of data and automatically perform feature selection, which is crucial in scenarios where not all measured attributes contribute equally to object identification. This makes Random Forest particularly suited for initial object labeling where the goal is to categorize objects without prior knowledge of their temporal characteristics.

3) **Cost matrix and object mapping**

After initial identification of static features, the next problem is objects mapping between different frame to calculate time-related features. I introduce a mechanism to mapping based on static features and then verify and adjust based on dynamic features.

This mechanism involves cost matrix of cosine similarities, physical distance and label-mismatch penalty to search for the lowest cost of mapping relationships based on static features. Once object mapping relationship come out, I calculate dynamic feature vectors.

The construction of a cost matrix and the application of the Hungarian algorithm are central to the dynamic aspect of object tracking, where the challenge lies in mapping objects across frames based on both static and dynamic features.

The cost matrix is an essential tool for object tracking as it quantifies the "cost" of matching each object in the current frame to objects in the subsequent frame. Each element of the matrix represents a potential match between objects across two frames, with the cost computed based on several factors:

a. **Spatial distance:** The physical distance between objects in different frames, which reflects their likelihood of being the same object based on movement. Considering real situation, the speed of objects has maximum limitation, such as 140 km/h in highway and 80 km/h in downtown. This can help us filter those which is too far to be treated as the same objects.

b. **Feature similarity:** Metrics such as cosine similarity for feature vectors, it is calculated as follows:

$$\text{cosine\_similarity} = \frac{\vec{A} \bullet \vec{B}}{\left\|\vec{A}\right\|\left\|\vec{B}\right\|} \tag{1}$$

The value of cosine similarity is within the scope of [-1,1], for value close to 1, the vectors are close to each other, and vice versa.

c. **Label matching**: The static label mismatching is considered as a cost since the static mismatching indicates low probability of the same object in adjacent frames.

Another significant algorithm in object mapping is the Hungarian algorithm[5], also known as the Kuhn-Munkres algorithm, which is employed to find the minimum cost matching between objects across frames. It effectively assigns objects in one frame to objects in another in a manner that minimizes the overall cost of these assignments. This method ensures that the sum of the costs of the assigned pairs is the lowest possible, thus optimizing the tracking process.

7. Results and Analysis

● Overview

The results and analysis section aims to assess the performance of 3D object detection models using advanced metrics and comprehensive visualizations. When evaluating the predicted objects and true objects, mapping is also an challenge to address. To map the object in predicted set and true set, I introduce techniques of *IoU* threshold and closest objects with lowest Euclidean distance.

● Evaluation Metrics

1) ***IoU***

$$IoU(o_i, h_j) = \frac{O_i \cap H_j}{O_i \cup H_j} \tag{2}$$

Where O_i and H_j are the corresponding 3D bounding boxes for o_i and h_j. I set *IoU* threshold for each type as 0.5. If $IoU(o_i ,h_j)$ is less than 0.5, I consider the tracker has missed the object. Mean IoU is calculated as:

$$\overline{IoU} = \frac{1}{N}\sum_{i=1}^{N} IoU_i \tag{3}$$

2) **Mean Position Error (*MPE*)**

$$MPE = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2+(z_i-\hat{z}_i)^2} \tag{4}$$

Where x_i,y_i,z_i are true center coordinates for object *i*, and $\hat{x}_i, \hat{y}_i, \hat{z}_i$ are predicted center coordinates object *i* if object i are matched as the same object, and *N* is the total number of predictions.

3) **Precision**

$$precision = \frac{TP}{TP+FP} \tag{5}$$

Where *TP* (True Positives) is the count of correct positive predictions, and *FP* (False Positives) is the count of negative instances incorrectly classified as positive.

4) **Recall**

$$recall = \frac{TP}{TP+FN} \tag{6}$$

Where *TP* (True Positives) is the count of correct positive predictions, and *FN* (False Negatives) is the count of positive instances incorrectly classified as negative.

- **Visualizations**

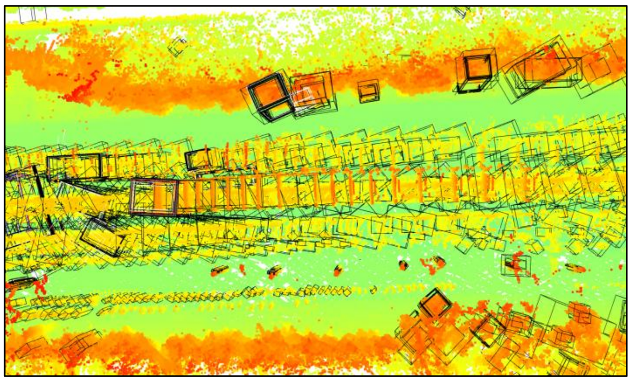

Figure 6: Object detection result rendering, whose bounding boxes are not 6 (Others)

- **Summary with basic results**

The evaluation metrics of this model is as follows:

```
{'Mean Position Error': 0.7120022922804883,
 'Mean IoU': 0.10327457323063242,
 'Precision': 0.3346662565364503,
 'Recall': 0.3486062159564242}
```

, which means:

1) **For Mean Position Error (0.712)**

This metric indicates a moderate level of error in the predicted positions of objects relative to their actual positions. Although there's a slight improvement compared to the previous data, the value suggests that there are still inaccuracies in how the model positions the bounding boxes around detected objects.

2) **For Mean Intersection over Union (0.103)**

This value is quite low and remains essentially unchanged from the previous report. It highlights a significant challenge in achieving precise overlap between the predicted and actual bounding boxes. The low IoU indicates that the algorithm struggles to accurately outline the objects in the environment, potentially due to the complexity or density of the scene.

3) **For Precision (0.335)**

A slight improvement in precision suggests that the detection system is becoming slightly better at correctly identifying true positives. However, the majority of positive predictions are still false, indicating a high rate of incorrect detection.

4) **For Recall (0.349)**

There's a slight improvement in recall, showing that the model is now capturing a slightly larger proportion of actual objects. Nonetheless, over 65% of true objects are still missed by the model, indicating considerable room for improvement in sensitivity.

- **model performance discussion**

Figure 6 shows a complex urban or cluttered setting with multiple objects closely packed together. The high density and diversity of objects might be contributing to the difficulties in achieving higher accuracy. The use of bounding boxes of different sizes suggests an attempt to detect objects of varying dimensions, but the low Mean IoU indicates that these boxes often do not align well with the actual object outlines.

The evaluation metrics also indicate noise points make performance of model worse. I learn that to get accurate bounding box of each object is crucial in this task. Inaccurate objects will disturb the detection and mapping process of the project.

8. Discussion and Conclusion

- **Discussion**

1) **Learning and Takeaways**:
   a. The performance significantly relies on DBSCAN parameters selection, if `eps` is too small, the cluster will be divided smaller; if `min_samples` is too large, nearly no objects will be detected.
   b. Noise points deteriorate performance of the model, as the Figure 6 shows, many objects are classified as incorrect type.
2) **Challenges Encountered**:
   a. PCA axis is not accurate since the points distribution sometimes concentrated in certain direction, due to Lidar scanning angle, distance to camera and other factors.
   b. Noise filter require more accurate technology, because the result shows that some incorrect objects are detected which actually are not targeted objects.
3) **Improvement Suggestions**:
   a. In the process of feature extraction, it is lack of the pose and shape feature in feature engineering.
   b. Better advanced technology such as Convolutional Neural Networks (CNN), maybe more suitable for this kind of issue, especially classification based on shapes.

- **Conclusion**

The exploration into the use of DBSCAN for clustering in the context of this study has provided valuable insights, particularly in the realms of parameter sensitivity and noise management. It has been determined that the selection of DBSCAN parameters such as `eps` and `min_samples` is crucial; improper values can lead either to overly fragmented clusters or an inability to distinguish any meaningful clusters at all. This sensitivity underscores the need for a careful approach to setting these parameters based on the specific data characteristics and the objectives of the analysis.

Further, the study highlighted the challenges posed by noise in the data. As evidenced in Figure 6, noise points significantly impair the model's performance by leading to misclassifications of object types. This issue is exacerbated by factors such as the inaccuracies in the PCA axis orientation, likely influenced by the variable conditions under which data was collected, such as Lidar scanning angles and the distances to the camera.

To address these issues, several improvements have been suggested. Firstly, enhancing the feature extraction process by incorporating pose and shape features could provide a more robust basis for clustering. Current feature engineering practices appear insufficient for capturing the complex variances within the data, particularly those relevant to object shapes and orientations.

Moreover, the incorporation of more sophisticated technologies like Convolutional Neural Networks (CNN) could offer substantial benefits. CNNs have demonstrated remarkable success in tasks requiring recognition and classification based on shape, making them a promising alternative for tackling the challenges identified in this study.

In summary, while the current approach using DBSCAN offers a foundational method for clustering analysis, the adoption of advanced techniques and a more nuanced handling of feature engineering and noise management could significantly enhance performance. These improvements are not just beneficial but necessary for progressing towards more accurate and reliable object detection and classification in complex datasets.

9. References

[1] M. Ester and H.-P. Kriegel and J Sander and X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, KDD, 1996.

[2] Ma, Y., Zhu, X., Zhang, S., Yang, R., Wang, W., & Manocha, D. (2019). TrafficPredict: Trajectory Prediction for Heterogeneous Traffic-Agents. AAAI (oral).

[3] ApolloScapeAuto. (n.d.). ApolloScape 3D Detection and Tracking Dataset. GitHub. Retrieved from https://github.com/ApolloScapeAuto/dataset-api/tree/master/3d_detection_tracking

[4] Zhou, Q.-Y., Park, J., & Koltun, V. (2018). Open3D: A Modern Library for 3D Data Processing. arXiv:1801.09847.

[5] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2, 83-97.