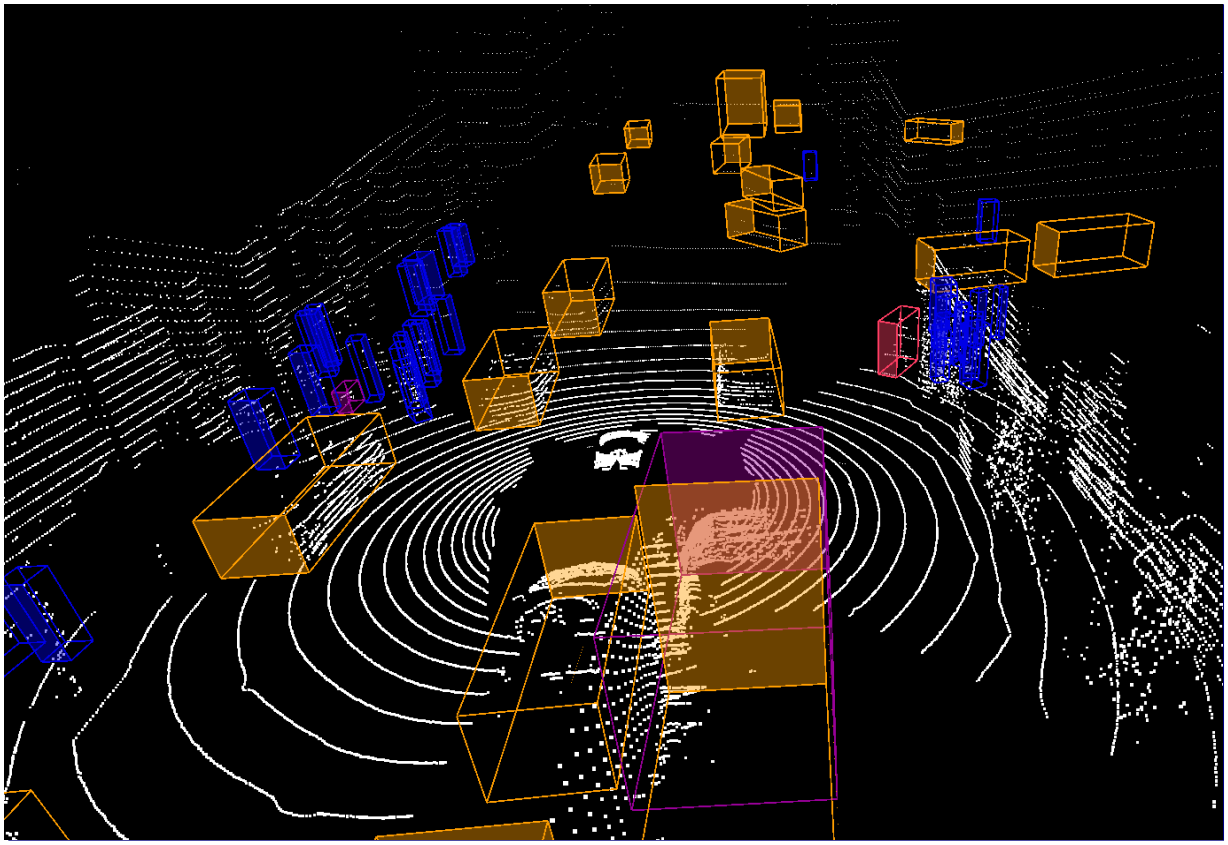
CHALMERS



Methodology of Driving Scenario Extraction Based on Vehicle Trajectory

Master's Thesis in Mobility Engineering

XINHAO WANG

HUAXIANG WU

Department of Architecture and Civil Engineering
Division of Geology and Geotechnics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
Master's thesis ACEX30

MASTER'S THESIS ACEX30

Methodology of Driving Scenario Extraction Based on Vehicle Trajectory

XINHAO WANG

HUAXIANG WU

Department of Architecture and Civil Engineering
Division of Geology and Geotechnics
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2023

Methodology of Driving Scenario Extraction Based on Vehicle Trajectory

XINHAO WANG
HUAXIANG WU

Examensarbete ACEX30
Institutionen för arkitektur och samhällsbyggnadsteknik
Chalmers tekniska högskola, 2023

Department of Architecture and Civil Engineering
Division of Geology and Geotechnics
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Cover:
Visualization of object detection from NuScenes dataset.

Chalmers Reproservice / Department of Architecture and Civil Engineering
Göteborg, Sweden 2023

Methodology of Driving Scenario Extraction Based on Vehicle Trajectory
Master's Thesis in Mobility Engineering
XINHAO WANG
HUAXIANG WU
Department of Architecture and Civil Engineering
Division of Geology and Geotechnics
Chalmers University of Technology

ABSTRACT

In the development of autonomous drive functions, in-house testing and simulation are indispensable, which are very dependent on large amounts of data. Unfortunately, labeling the data and picking up useful scenarios are difficult since there is so much redundant information that is useless. This is why the thesis is born.

Specifically, the thesis aims to mine as many interesting driving scenarios as possible with the help of rule-based methods and clustering methods and to find a better way to do scenario extraction from data. Based on the open-source data set NuScenes, the rule-based method is first used but with less than satisfactory results due to too little information. The total number of objects cut in, fast approaching, and object cut-out scenarios is around 81, which is relatively small compared to the entire data set, and the classification precision rate is about 50%. Then, clustering models including HDBSCAN, K-means, GMM, Hierarchical clustering, and DBSCAN are tried. On the basis of these models, the work done includes but is not limited to cross-comparing the impact of distance matrices obtained from DTW and LCSS, different partitioning settings, and whether to scale the input data. Certainly, hyperparameter adjustment is also done by the Optuna algorithm. The results from the rule-based method are used to generate metrics and objectives for clustering models. Consequently, the best F_2-score is from the K-means model, which is 0.1688. For comparison, the same method and idea are used on another data set named NGSIM. It turns out that the rule-based results are very good, with the precision rate reaching up to 90% because of the rich information provided like lane ID. Correspondingly, the clustering results improved a lot, reaching 0.4653 for the DBSCAN model, even though it is still far from the ideal result.

In summary, the external lane and object information is quite crucial for rule-based method classifying scenarios. Besides, the rule-based method is more efficient than clustering in scenario classification and clustering models may not be applicable to driving scenario extraction based on relative trajectory classification.

Keywords: NuScenes, NGSIM, rule-based, clustering, scenario extraction.

Acknowledgements

The thesis is carried out between China Euro Vehicle Technology (CEVT) and Chalmers University of Technology under the support of supervisors Yasaman Alavi, Mikel Broström, Per Nordqvist, and Hui Zhong from CEVT. The examiner is Jiaming Wu from Chalmers University of Technology.

First of all, we would like to thank our supervisors from CEVT, who gave us a lot of help and instructions. Yasaman was very responsible for the whole thesis and played a decisive role in the structural direction and task milestone setting. Mikel brings his expertise and provides a lot of technical guidance as a machine learning expert. Per brought many useful ideas and Hui even helped us contact external expert Feng who also supported us in technical issues.

Many thanks to our examiner Jiaming, who helped us a lot in terms of thesis procedure, documentation, etc. Furthermore, he helped us analyze the problem, reorganize the logic, and guided us to explore new avenues towards the final target when we encountered difficulties and slow progress.

We sincerely show gratitude to the AD/ADAS team in CEVT for providing such a friendly environment. Especially our manager Alejandro Morate, who involved us in the team's daily routine, and also gave us many chess guidance, which entertained us a lot.

And both of us would like to thank our teammate, this work could not be done without you.

Contents

List of Acronyms

AD	Autonomous Driving
CEVT	China Euro Vehicle Technology
CMA	Compact Modular Architecture
DBSCAN	Density-Based Spatial Clustering of Application with Noise
DTW	Dynamic Time Warping
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
LCSS	Longest Common Sub-Sequence
Lidar	Light Detection and Ranging
MDL	Minimum Description Length
MHMM	mixture of hidden Markov models
NGSIM	Next Generation Simulation
ODD	Operational Design Domain
OEM	Original Equipment Manufacturer
OPTICS	Ordering Points To Identify the Clustering Structure
PCA	Principal Component Analysis
RC-GAN	Recurrent Conditional GAN
SAE	Society of Automotive Engineers
TRACLUS	Trajectory Clustering
t-SNE	t-distributed stochastic neighbor embedding
TTC	Time to Collision

List of Figures

List of Tables

1 Introduction

1.1 Background

Automated driving has gained great achievement over the last few years, an L2 automated driving system, defined by the Society of Automotive Engineers (SAE) [1], is set in many new vehicles in the market nowadays. While L2 is only partial driving automation, higher-level automated driving is also being investigated and developed to make vehicles more intelligent and safer.

Before people trust autonomous driving vehicles, they should be tested to prove that they are safe enough to be on the road without human drivers' actions, because autonomous driving could also bring new risks. Research has presented that it may need hundreds of millions of miles, sometimes even hundreds of billions of miles to prove autonomous driving vehicles' reliability since it's rare to see fatalities and injuries [2]. Thus, X-in-the-loop is introduced for simulation autonomous driving testing, as it is more affordable and simpler [3]. Software-in-the-loop is one of the simplest and repeatable ways to do testing, that's also why it's attractive for vehicle companies.

However, what could be the input of this testing? One of the solutions could be scenario-based testing [7][8]. Specific driving scenarios that a driver may encounter need to be specified and tested [9], e.g., cut-in, lane change, etc. These critical scenarios need to be extracted from real-world data to build more test cases that are close enough to reality. Then, if testing takes scenarios as input, how driving scenarios can be extracted from real-world data becomes crucial.

In this developing process of scenario extraction, real driving data is a fundamental part, because of the need to analyze practical driving behaviors and have sufficient support to test autonomous driving strategies. There are some open-source driving data like NuScenes [4], KITTI [5], and also driving data collected by the companies. 3D object detection has already been developed by multiple researchers based on those data in different ways [6].

Besides, there are two main ways to classify the scenarios by analyzing the extracted trajectories, rule-based algorithms, and machine learning methods.

1) Rule-based

For the former way, Zhao et al. (2017) [11] proposed an open naturalistic driving scenario library called TrafficNet, which was extracted and preprocessed from raw sensor data using rule-based algorithms. As mentioned in the literature, the dataset for the six most widely used scenarios could be used for vehicle evaluation and self-driving functions test. The original dataset used is informative since it contains information such as the type, velocity, position of the object in front of the vehicle, and even lane information. The author set some thresholds and status flags to classify 6 critical driving scenarios, including free flow, car-following, frontal cut-in, lane change, pedestrian crossing, and cyclist, based on two well-prepared tables called Event table, which stores the main keys of all the events, and Sequence table, which logs all the data for each scenario in time sequence.

In the ConScenD dataset [12], the authors considered both cut-in and lead vehicle brakes as interesting scenarios. The parameters being used include challenging vehicle

brake duration, challenging vehicle cut-in trigger distance, challenging vehicle brake trigger distance, etc. It also takes swerving vehicles into account as a scenario, which would be crucial and useful for highway driving operational domain. The paper tried different velocity threshold settings and compared the number of results for different thresholds.

2) Machine learning

Demetriou et al. (2020) [13] used an architecture based on combined Recurrent Autoencoder with GANs (Generative Adversarial Networks), and a Recurrent Conditional GANs (RC-GAN) architecture to generate vehicle driving trajectories. The trajectories' dimensionality was reduced with Principal Component Analysis (PCA) and t-SNE [14], then clustering methods were applied to classify the trajectories into three classes (cut-in, left pass-by, and right pass-by).

Martinsson et al. (2018) [15] implemented a mixture of hidden Markov models (MHMM) to deal with the problem of unequal length among data vectors. The authors used a Python package *pomegranate* and stacked a general mixture model on top of HMM, which in turn are stacked onto multivariable Gaussian distributions.

Hoseini et al. (2020) [16] first computed pairwise dissimilarities between the variable-length time series and extracted the temporal relations, and then embedded them into a vector space. Next extracted transitive relations between trajectories and embedded into a vector space. Finally, the authors applied the clustering method.

To sum up, most authors discovered scenario extraction based on their own datasets collected by stationary sensors or generated datasets, it's rare to see a validation of the method's reliability on data collected by driving, which is a more practical way for the vehicle companies to collect data. Additionally, a comparison between rule-based methods and machine learning methods can also be made to find which would be the better way to classify scenarios. Therefore, this thesis aims to find a way that is suitable for scenario extraction based on trajectories and will try to compare the efficiency of rule-based methods and machine learning methods.

In the first chapter, the background will be described, and some literature review has also been done. Then, the datasets have been chosen and the way of preprocessing will be explained, trajectories will be made by the end of this chapter. With extracted trajectories, rule-based method and machine learning method will be implemented respectively. Finally, the results will be analyzed and compared between two kinds of algorithm.

1.2 Literature review

Since rule-based methods have been reviewed in the last part, it won't be further explained in this section. The literature review will cover the theoretical knowledge of clustering and scenario definition.

1.2.1 Clustering and trajectory processing

The last part is about some latest research on the scenario-related topic, this part will give a review of which clustering method is available to be used and what corresponding reprocessing session needs to be done before clustering. It is considered to be a more technical and practical part. Thus, a separate sub-chapter is listed here.

1) Clustering

Clustering is an unsupervised learning algorithm that doesn't need labels to be ground truth. It can group data into multiple clusters based on their features and similarities, it's a useful tool for data mining and unknown features exploration [17].

Because of the scenario that is being faced with, the number of clusters cannot be defined. In other words, there must be many unknown driving scenarios that cannot be come up with in advance. Therefore, the density-based clustering method becomes more interesting, for which Density-Based Spatial Clustering of Application with Noise (DBSCAN) [18] is most representative. Given a set of data points, each point will be first treated as a core point to find other data points that are within the range of $[0, \varepsilon]$. If the core point has found more than min_{sample} data points, it can be treated as a true core point p_1 and form a new cluster c_1 including all neighbors. Its neighbors are able to do the same thing and include more points in this cluster. If a data point cannot find enough neighbors within the range of $[0, \varepsilon]$, and also cannot be detected as a neighbor point by others, this point becomes a noise point. The advantage of DBSCAN is that this algorithm generates clusters that are not limited by irregular shapes. It would also automatically adjust the number of clusters corresponding to the parameter ε.

While ε is set mostly based on the user's experience, it's hard to determine optimally [19]. To fix this problem, Ankerst et al. (1999) [20] proposed Ordering Points To Identify the Clustering Structure (OPTICS), which creates a segmented ordering that represents the density-based clustering structure of the dataset. This contains similar information to the result but doesn't make the only cluster distribution directly.

Another interesting way to solve the ε problem is Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [21]. This algorithm could perform DBSCAN over varying ε values and find the best integration of all clustering results which would give the best stability. It just needs a little hyperparameter tuning with minimum cluster size and minimum sample, without modifying ε. Healy (2018) [22] introduced HDBSCAN as a hierarchical and density-based clustering method, apart from k-means as a flat and centroid-based method or DBSCAN as a flat and density-based method. In conclusion, HDBSCAN takes a reasonable time with little parameter tuning, and it's trustworthy to give meaningful clusters.

2) Similarity and distance measurements

Clustering is the way to realize grouping or classification, however, the set of trajectories cannot be the input to fit the model. The trajectories are high-dimensional time series data and are not possible to be directly processed by clustering algorithms because of low efficiency. In principle, the approach to clustering trajectories is to find similarities in the motion of objects, and then group trajectories with similar motion patterns into the same class. One of the most important parts is similarity measurement,

which means trajectories need to be compared with each other and generate similarity scores to fit the clustering model [19]. Based on the property of trajectories and the aim of classification, there could be multiple ways to do processing and similarity measurement, about which this part will try to give some ideas.

Euclidean distance is the most widely used implementation of distance measurement. It would take two points corresponding to the same time and calculate their Euclidean distance. Let trajectory L_i and L_j be the example, which is p-dimensional with a length of n. Their Euclidean distance could be derived from the formula below:

$$D_E(L_i, L_j) = \frac{1}{n}\sum_{k=1}^{n}\sqrt{\sum_{m=1}^{p}(a_k^m - b_k^m)^2} \tag{1.1}$$

Although this algorithm is very efficient with a complexity of $O(n)$, it needs many prerequisites to be fulfilled. In practice, Euclidean distance could be constrained by trajectories' asynchrony or different lengths.

Dynamic time warping (DTW) estimate [23] is a way to find the optimal alignment between two trajectories no matter they have the same length or not. This algorithm could locally scale the time dimension and find similarities between trajectories by calculating the smallest Euclidean distance. Since it could overcome problems with different trajectory lengths, it's more realistic in actual trajectory comparisons.

While the algorithms above focus on the whole trajectory's similarity, the special feature could happen only at a few parts of the trajectories. If the dissimilarity of the remaining parts is significant, it would cause recognition errors for global matching algorithms.

The Longest Common Sub-sequence (LCSS) algorithm could deal with the problem if the trajectories are not fully similar [24]. LCSS could extract the longest common subsequence between two trajectories and doesn't need all points aligned. Besides, two thresholds σ and ϵ for both lateral and longitudinal coordinate differences should be set. If the difference is less than the thresholds, this pair of points are considered similar and the value of LCSS is increased by 1 and this pair will be marked as 1 too. If all points are not similar, the LCSS value will be zero; if not, the longest common subsequence can be found by finding the consecutive "1" marks. The LCSS can be converted to a distance estimate by the formula below [25]:

$$D_{LCSS}(L_i, L_j) = 1 - \frac{LCSS_{\sigma,\epsilon}(L_i, L_j)}{\min(n,m)} \tag{1.2}$$

Where $\min(n, m)$ represents the minimum length between L_i and L_j, and $LCSS_{\sigma,\epsilon}(L_i, L_j)$ is the length of the longest common subsequence. The longer the LCSS, the lower the D_{LCSS}, and the high similarity score between the two trajectories.

DTW and LCSS have the same complexity $O(m * n)$, but DTW is more sensitive to noise and considers the similarity of the full trajectory, LCSS depends on the threshold setting and considers part of the trajectory.

Additionally, Lee et al. (2007) [26] developed an algorithm to only compare one single subsequence in trajectories. The algorithm called TRACLUS (Trajectory Clustering)

partitions trajectories into line segments to classify. It uses minimum description length (MDL) as the principle to partition trajectories and implement a very DBSCAN-like clustering algorithm to group the segments while avoiding only the segments from one or very few trajectories in one cluster. Regarding similarity measurement, this algorithm takes longitudinal, lateral distance, and angle differences into account.

Because the algorithm above partitions trajectories before clustering, it may be limited by the time windows of the segments. T-OPTICS proposed by Nanni et al. (2006) [27] could partly solve this problem. It would do clustering to every possibly reasonable time window of trajectories and compare the clustering results. Finally, the optimal result shall be got, which leads to the highest similarity within one cluster and the lowest similarity among clusters. But this algorithm has a higher complexity and may not be suitable for trajectories that have interesting segments at very different times.

1.2.2 Scenario definition

Had the methods of classification, it's also important to have a general understanding of scenarios before the extraction. The official description of scenarios is a parameter-free space-time description of the traffic state [28]. More specifically, scenarios can be split into three sub-classes, functional scenario, logic scenario, and concrete scenario.

The authors Erwin de Gelder et al. (2020) [29] proposed that scenarios should cover a high variety of situations that Automated Vehicles could encounter. Hence, scenario categories are generated where a scenario category could be viewed as a concentration of a quantitative scenario. The authors introduced some necessary concepts to define the term scenario including Ego vehicle, Activity, Event, Actor, Static environment, Dynamic environment, and Conditions. The definition of scenario taken from [30] is a quantitative description of the ego vehicle, its activities, and/or goals, its dynamic environment (consisting of traffic environment and conditions), and its static environment. From the perspective of the ego vehicle, a scenario contains all relevant events. The difference between this and others is the definition from [30] explicitly claims that a scenario should be regarded as a quantitative description. However, the authors Erwin de Gelder et al. (2020) [29] suppose a scenario contains both quantitative and qualitative descriptions. The qualitative description is defined by the author as a scenario category, which contains concrete and comprehensive descriptions as well as tags that characterize the category. For instance, if a scenario happens during nighttime, it will include a property marked as 'tags={Nighttime}'. The authors highly recommend using tags and trees of tags considering the required different levels of varieties existing in scenarios.

According to Erwin de Gelder et al. (2020) [29], the given cut-in in front of the ego vehicle is generally described as the object vehicle driving in the same direction as the ego vehicle in an adjacent lane. Later on, the object vehicle changes the original lane so that it becomes the lead vehicle of the ego vehicle. It is not important to regard the reason for a lane change. The provided definition of scenario "Ego performing lane change with the vehicle behind" is the object vehicle is driving in the same direction as the ego vehicle in an adjacent lane. Afterward, the ego vehicle makes a lane change toward the lane where the object vehicle is driving. The presented definition of the cut-out in front of the ego vehicle is the object vehicle is driving in the same direction as

the ego vehicle acting as the lead vehicle from the perspective of the ego vehicle and then the object vehicle is not the lead vehicle any longer. Last but not least, the definition of an Ego vehicle approaching a slower lead vehicle is the object vehicle is driving slower in front of the ego vehicle because of some reason. Consequently, the object vehicle enters the ego vehicle's field of view. As a result, the ego vehicle could brake or change the lane to avoid a possible collision. All of the above descriptions are general descriptions, but the author also provides formal descriptions including static environment, ego vehicle, dynamic environment, and tags.

2 Data

In this chapter, two datasets, i.e., NuScenes data and NGSIM data are introduced in detail. The introduction is conducted in terms of several different angles including the origins, features, structures, and contained information of the data, the data processing, and the method of extracting trajectories for respective dataset. Afterwards, a comparison between these two data is made, which is intuitive for readers to comprehend the differences, strengths, and weaknesses of these two data. Lastly, the categories of chosen target scenarios and the definitions of them in the thesis are clarified explicitly for these two datasets individually.

2.1 NuScenes data

Firstly, NuScenes data was under investigation, which is a multimodal dataset [4]. It collected driving sensor data including Lidar (Light detection and ranging), radar, and cameras from Boston and Singapore. The testing ego vehicle was mounted with 5 radars, 6 cameras around the vehicle, and 1 Lidar on the top of the vehicle. The data was generated based on the global coordinate system, the city map, which gives each of the objects its absolute coordinates in a certain range of the city.

Besides, the dataset has an object list that labels most of the objects having been detected by the sensors, e.g., pedestrians, cars, and trucks, and also their corresponding location coordinates on the local map. Each object would have a unique token, and for each frame that the object shows up, there's another token for the frame (called "sample" in NuScenes) as well. Therefore, the absolute position of objects in the local map can be tracked and extracted from the NuScenes dataset. Some examples of NuScenes' annotations are shown below.

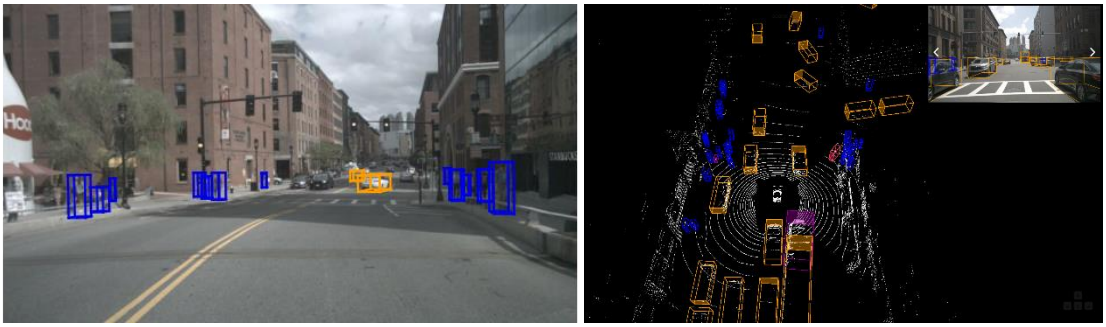


Figure 2.1 NuScenes dataset's visualization.

Specifically, the NuScenes dataset has both the ego vehicle's coordinates and all objects' coordinates with corresponding timestamps, from which the trajectory of the vehicles' relative movements may be generated.

Coding details will be mentioned from here on, it is important to clarify that the language being used is Python. Thus, many libraries in Python will be implemented.

2.1.1 Trajectory extraction

Trajectory, in this case, to analyze driving scenarios, is the sequence of the vehicle's position, which contains space and time information of the vehicle. It can be split into two sub-categories, absolute trajectory, and relative trajectory. The former indicates the trajectory that describes temporal-spatial relationships between vehicles and the fixed space, i.e., the local map based on GPS (Global Positioning System). The relative trajectory is the trajectory that represents the relative temporal-spatial relationships between object vehicles and the ego vehicle, which means the ego's position will always be $(0,0)$ in a relative coordinate system.

Each piece of recorded data is called a scene in NuScenes, which contains 40 samples of data. The coordinates of each object in each sample can be found by using the NuScenes devkit tool.

First, interesting categories need to be defined. Considering scenarios that are going to be investigated will be mainly correlated with vehicles, the following classes were listed: car, motorcycle, bus bendy, bus rigid, and truck. The class-choosing function can be done with the help of NuScenes' annotation system. NGSIM only contains vehicles, so this part of filtering could be omitted.

Second, all objects' information, no matter which scene it was located, was looped to find the coordinates $c_n = (x_n, y_n)$, and generate trajectories $t_i = (c_1, c_2, \dots, c_n), n \in [10,41]$. Because of synchronization between the Lidar and cameras, each scene could have 40 or 41 samples. Besides, even though the scene has 41 samples, most of the objects don't have a trajectory of length 41 since an object cannot be always in the ego vehicle's field of view. Thus, if the trajectory is shorter than 41 pairs of coordinates, it needs to be padded with zeros $z_n = (NaN, NaN)$ at the end of the trajectory sequence to stack all trajectories in one array using $numpy$ library. Finally, each trajectory would be $t_i = (c_1, c_2, \dots, c_n, z_1, \dots, z_{41-n})$. The absolute trajectory in the global coordinate system was extracted for both the ego vehicle and objects and stored in pairs.

Additionally, the category corresponding to each trajectory sequence was also extracted, as well as its scene token and instance token, in case visualization and manual accuracy validation are needed.

2.1.2 Data filtering

Two criteria are being considered, trajectory length, and relative distance between the ego vehicle and the objects.

1) Since the timestep between two recorded pairs of coordinates is 0.5 seconds and the trajectory is considered meaningful only if the included information is longer than 5 seconds, the length of each trajectory should be longer than 10 elements.

2) Besides, the relative distance needs to be shorter than 100 meters to be treated as meaningful. The schematic diagram is shown below.

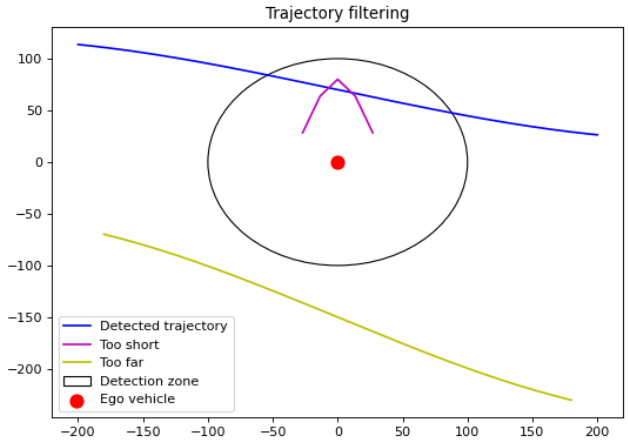


Figure 2.2 Trajectory filtering logic visualization.

In the figure above, the purple curve only has 5 pairs of coordinates, and no points from the yellow curve are in the detection zone, therefore, they will be regarded as meaningless and be excluded.

2.2 NGSIM dataset

As the last part showed, this part will introduce the NGSIM dataset and present how it was processed.

2.2.1 NGSIM introduction

Next Generation Simulation (NGSIM) open data [31] is provided by the US Federal Highway Administration (FHWA). Having partnered with ITH DataHub, the detailed vehicle trajectory data has been generated from the collected videos from NGSIM. The NGSIM program collected traffic flow data from four locations, including two highway and two arterial segments. The original version of chosen data consists of 25 columns and 11.8 million rows of trajectory data. The reconstructed version [32] of highway data collected at the US I-80 [33] has been used in this thesis, where noises are reduced, and the trajectories are more continuous.

Compared to NuScenes, the trajectory data has Lane_ID, following Vehicle_ID, preceding Vehicle_ID, velocity, acceleration, and space headway information, which are significant information for classifying scenarios by providing the vehicles' relative location.

2.2.2 Data preprocessing

After receiving the data specifically collected at the US I-80, the redundant information was removed for less needed memory and only relevant information containing 13

entities is retained, including Vehicle_ID, Frame_ID, Local_X, Local_Y, Lane_ID, velocity, etc.

Making use of lane ID information, the instances of vehicles making lane changes could be extracted. Specifically, the lane ID numbers are iterated over and the vehicles that change lane ID but have constant vehicle ID and total frames are extracted. Plus, the vehicle ID and frame ID at the time points when vehicles make lane changes are stored for later reference.

Later on, partitioning is conducted to concentrate on interesting segments of vehicle trajectories. In detail, what is done is to extend frames forward for fifty and backward for thirty based on the frame where the vehicle makes a lane change, which constitutes the window of 8 seconds near the lane-change maneuvers. Then, within the whole dataset, the rows of vehicle trajectory data of which the frame ID numbers are within the interval could be extracted. As a result, the same number as the length of frames where vehicles make lane changes of data frames are extracted. Then, for each data frame, the frame ID numbers of both the vehicle making a lane change and other object vehicles are synchronized by truncating so that all the vehicles in a data frame have overlapping intervals.

When it comes to the part of scenario classification, in each data frame, the data row where the vehicle started making a lane change could be located and the data rows that are two frames ahead of the mentioned row and behind the row could be easily pinpointed. Once the base vehicle is set as the lane-change vehicle, since the data set has the information of the following vehicle ID, the following vehicle ID numbers of the base vehicle before and after making the lane change could be determined. In detail, if the following vehicle ID number is 0, it means there is no following vehicle. On the contrary, other numbers represent the real vehicle ID of object vehicles. With the information achieved, it is easy to decide whether there is a following vehicle behind before the base vehicle changes the lane, which is a potential vehicle that the base vehicle cuts out and likewise, the vehicle behind the base vehicle might be the vehicle it cuts in. To add more limit, the normally used parameter TTC (Time to Collision) is introduced, which is the result of space headway divided by following vehicle velocity and is set as 3s as the threshold. On top of these, the direction of lane change is also determined based on the lane ID number. According to the documentation, the smaller the lane ID number, the further left track is the vehicle on. Consequently, four categories including ‘left cut in’, ‘right cut in’, ‘left cut out’, and ‘right cut out’ are generated.

Additionally, the relative trajectories between object vehicles and base vehicles, which would be fed through the cluster model, are also generated. Literally, the relative trajectories are generated by subtracting the X and Y coordinates of base vehicles respectively with corresponding frame ID from the X and Y coordinates of object vehicles, which would lead to $t_i = (c_1, c_2, \dots, c_n),\ c_n = (x_n, y_n)$, where x_n and y_n are individual relative X and Y coordinates. In total, around 86000 relative trajectories are produced.

2.3 Dataset comparison and analysis

Table 2.1 Dataset comparison

Dataset	Nuscenes	NGSIM (I80)
Author	Motional	Federal Highway Administration
Format	.tar	.zip
Sampling frequency	2	10
Annotation	Annotated using human experts	Annotated using human experts
Location	Singapore and Boston	Los Angeles, Emeryville, Atlanta, California, Georgia
Duration	1000 scenes * 20 sec	15-minute video
Sensor devices	Cameras, Lidar, radars, GPS, inertial measurement unit	Synchronized digital video cameras
Road types	City road	Freeway
Lane ID	None	Leftmost (1) – Rightmost (7)
Following vehicle ID	None	Yes
Vehicle velocity	None	Yes
Vehicle acceleration	None	Yes
Space headway	None	Yes
Relative trajectories generated	34056	86074

As shown above, there are quite a few significant discriminations between Nuscenes and NGSIM data sets. With respect to frequency, there are around 40 samples in one scene within 20 seconds for NuScenes data. This results in 0.5 seconds time gap between every two samples. While NGSIM data just have 0.1 second time gap between every two successive frames, which could produce more accurate and denser information.

Apart from this, the sensors used for Nuscenes data are multiple including lidar, radar, camera, and so on, which are mounted on the ego vehicle to detect ambient objects. However, for the chosen NGSIM dataset, there is just one static synchronized digital video camera monitoring the freeway, creating several 15-minute continuous videos. In the case of NGSIM data, there is no determinate ego or object vehicle.

Furthermore, the NuScenes data collected when the ego vehicle runs around cities without many limits, which is relatively messy while NGSIM data reflect more regular vehicle maneuvers only happening on straight-line freeway.

Most importantly, in terms of scenario extraction, only X and Y coordinates in NuScenes data could be taken advantage of yet for NGSIM data, not only the X and Y coordinates could be made use of, but also the Lane ID information and Following vehicle ID can be used, which is convenient for determining lane-change and the existence of following vehicles when cutting in or cutting out. Considering this difference, it is inherently easier to extract scenarios from NGSIM data than from NuScenes data because there is much more information to rely on in NGSIM data than in NuScenes data.

2.4 Scenario clarification

After considering the recognizability and regularity of the object trajectory, it is decided to include only motor vehicles in the driving scenario classification task, and the specific categories have been mentioned in Chapter 2.2. This is because motor vehicles need to abide by traffic regulations, and the generated trajectories are relatively regular in time and space. For example, most vehicles on highways will keep driving above 80 km/h most of the time without changing lanes. However, the behavior of pedestrians will be much more irregular. For example, they may suddenly receive a call while walking and need to stop to answer the phone. This leads to the difficulty of classifying and extracting scenarios for non-motorized vehicle object trajectories, so they will not be considered in this thesis.

The scenarios being considered are split into 2 major categories and 3 subcategories. They are listed below. The schematic diagram of each scenario is also enclosed.

Table 2.2 Scenario clarification from high-level.

Category		Description
Cut-in/out	Objects cut-in	A vehicle changes from another lane to the ego's lane, within a certain range from the ego at the end of the cut-in.
	Objects cut-out	An object changes from the ego's lane to another lane, within a certain range from the ego at the start of the cut-out.
Approaching	Fast approaching	If it's none of the scenarios above, the distance between an object and the ego is decreasing continuously, it is considered as approaching. While the vehicle is approaching the object in front, their velocity difference is large, and have a relatively high risk of crashing.

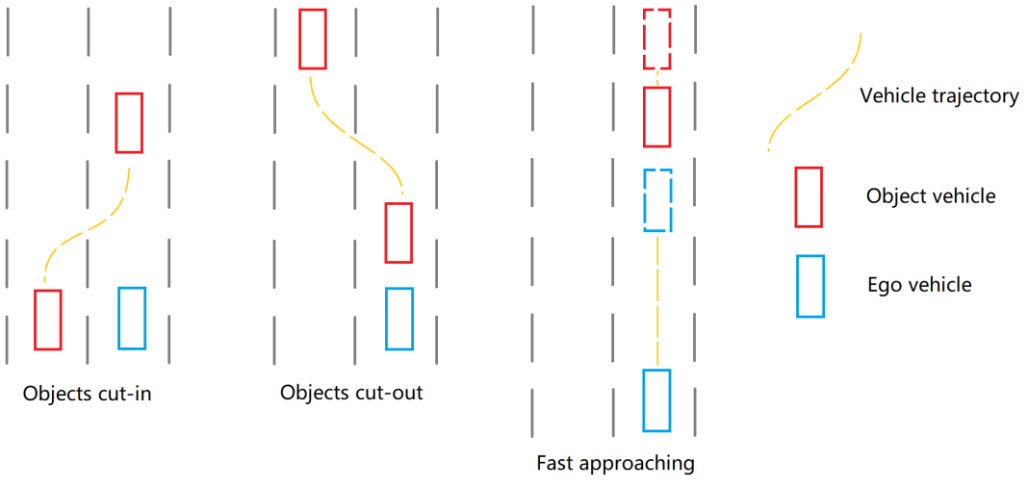


Figure 2.3 Scenarios schematic diagram.

2.5 Summary

This chapter explained the dataset being used, and the scenarios that are going to be identified. Meanwhile, the dataset was filtered roughly at this stage to exclude useless trajectories and objects. After this chapter, the trajectory set is well-prepared.

3 Rule-based classification algorithm

In this chapter, a rule-based method will be developed and implemented, the logic for cut-in/cut-out and fast approaching will be different. At the end of this chapter, the result will be presented, and different thresholds will be tested to find a most appropriate combination of thresholds. The final result of rule-based result will be further compared with machine learning results in the next part.

3.1 Cut-in and cut-out

As a matter of fact, for NuScenes data, only relative coordinates could be utilized, which is not trivial at all. Due to this, the rules set would be inevitably subjective and have nothing to refer to. Now that the target scenario is cut-in, those series of relative x coordinates that decrease continuously are regarded as the coordinates at the moments of making a lane change towards the adjacent ego vehicle. Based on this, further rules could be conducted.

TTC is a key parameter, which could limit the scope of involved object vehicles and could be modified according to classification results, in which case it is set as 5 eventually. Besides, the rules ensuring that the ego and object vehicle is on the same line should be set like the lateral distance after finishing cutting in should be within a certain range, which is 0 to 1.6. And the differential of the slope of ego and object vehicle should be within a scope, of which the absolute value should be smaller than 0.5 in the case. Also, the displacements of both ego and object vehicles should be larger than a threshold value in the process, which is 3 meters, to make sure both of them are moving.

Aside from those, the rules of guaranteeing the object vehicle actually change the lane should be made like the lateral displacement should exceed a value, that is 1.8 meters here, and the coordinate slope change should also exceed one set value, which is 0.04 here. Additionally, both ego and object should drive straight before the cut-in maneuver by limiting the slope differential as well, which is set to be less than 0.2, and should drive in the same direction by limiting the changing trend of coordinates, either decreasing simultaneously or increasing meanwhile.

Similarly, the rules for cut-out almost follow the same logic but with some slight dissimilarity. For example, for cut out, it is those series of relative x coordinates that increase instead of decreasing continuously should be found first. The ego and object should be aligned in the same line before cutting out, where the lateral distance should not exceed 1.6 meters and the slope discrimination should be below 0.2 and should drive parallel with a slope difference smaller than 0.2 after cutting out, during which process the ego vehicle highly probably drive straight with slope change lower than 0.2 and the slope of the object vehicle coordinates should vary more than 0.02 due to lane change. However, due to the lack of information, no matter whether cut-in or cut-out, neither result is not as good as expected. The basic flow charts demonstrating the logic are as follows:

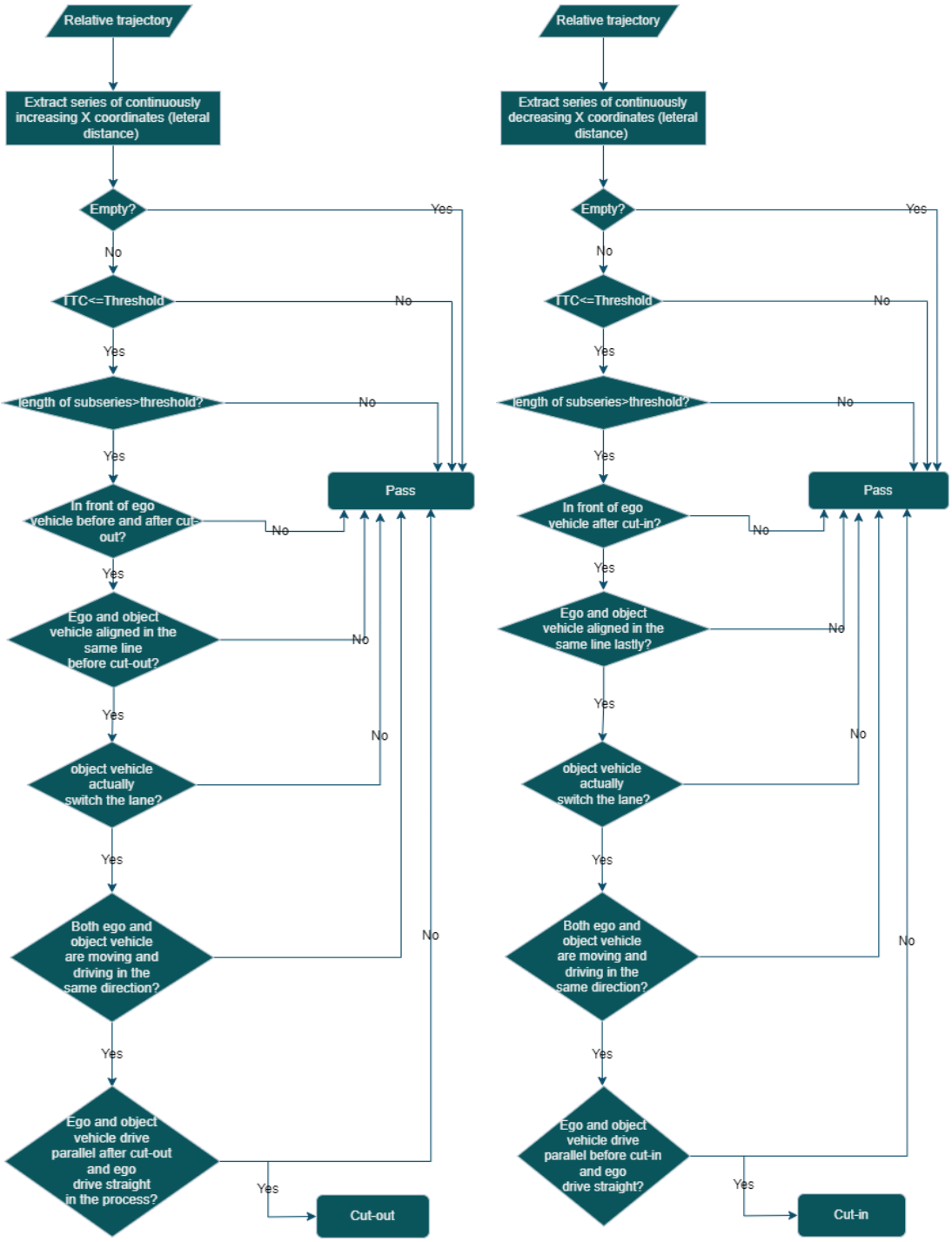


Figure 3.1 NuScenes scenario recognition flow chart.

Different from the rules for NuScenes data, the rules for NGSIM data are more straightforward and logical. To be brief, with lane ID and following vehicle ID, it is not tough to find vehicles that change the lane and the information of following vehicles both before and after lane change with corresponding frame ID, which represents timestep. On the basis of this, one limit that needs to be imposed further is TTC, which is set as 3 seconds in this case. When manually checking some classified samples, it turns out that pretty much all of them are correctly categorized, providing a high precision rate, so compared to the rule-based result from NuScenes data, the one from NGSIM data is much more excellent. The flow chart representing the logic is as follows:

Figure 3.2 NGSIM scenario recognition flow chart.

3.2 Fast approaching

Euclidean distance between the ego vehicle and the object that is under viewing is calculated for each pair of coordinates. If the distance is decreasing in two consecutive timesteps, in other words, three consecutive pairs of coordinates, and the object vehicle is in the same lane as the ego vehicle, it is considered that the ego vehicle is approaching the front object vehicle. This would be the prerequisite for identifying whether it's fast approaching or not.

As long as the trajectory shows an approaching maneuver of the ego vehicle to the front object vehicle, it should be judged by further rules.

It's worth a mention here that according to Ma et al. (2020) [34] and AASHTO. (2011) [35], $\pm 0.35g$ could be a reasonable threshold for longitudinal acceleration and deceleration. This is a conservative acceleration threshold for drivers to keep controlling vehicles even on wet roads. And in the extracted trajectories, the timestep between each data point is 0.5 seconds, which allows the velocity difference to be a threshold metric corresponding to the reference. Besides, to avoid the oncoming vehicle effects from the opposite lane, the absolute velocity difference should also be lower than ego vehicle velocity. To make it easier and include more positive predictions, the velocity difference setting is calculated as below:

$$\left|v_{ego}\right| > |dv| > 0.35g * 0.5 \approx 1.5m/s^2 \tag{3.1}$$

Additionally, the distance between vehicles should also be taken into account to measure the risk of crashing. TTC is considered as another threshold here and has been set to 3 seconds, which refers to Brown et al. (2001) [36] that it would be considered a dangerous situation when the TTC is lower than 3 seconds.

Admittedly, the ego vehicle can approach the front vehicle while it's steering. But because of the lack of lane ID information, to avoid further algorithm misjudgments, steering will be excluded from fast-approaching classification by reviewing the ego's yaw angle difference. The flow chart showing the logic is as follows:

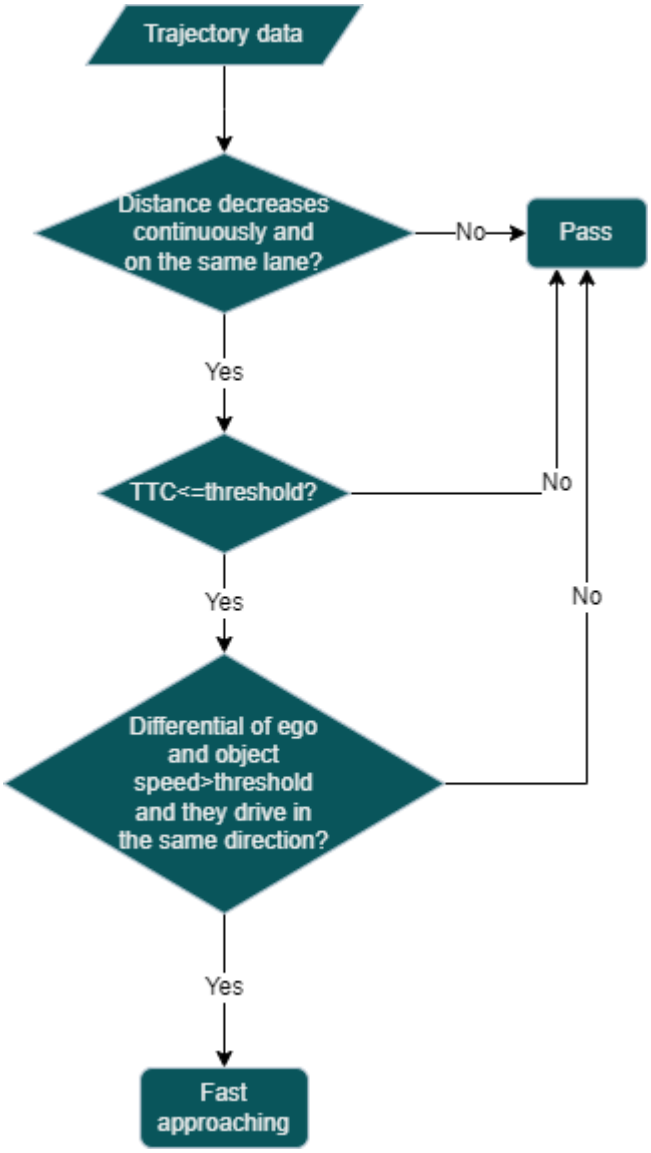


Figure 3.3 NuScenes fast approaching recognition flow chart

3.3 Results for different thresholds

From the description above, TTC is one of the key thresholds that measure the relative distance and danger between two vehicles for both cut-in/out and approaching scenarios. But in the case without ground truth, TTC can only be tuned manually. This may affect

the precision of the classification result. To evaluate this effect, different thresholds were set, and corresponding results were generated.

With regard to NuScenes data, the results were evaluated by reviewing image data from the NuScenes dataset using the tool from NuScenes-devkit [4]. Each object's driving maneuver and the correlation with the ego vehicle were assessed by the authors, and the true annotations were made for the rule-based result depending on the review.

Regarding NGSIM data, the total number of extracted desired scenarios is quite large, and only by watching the video manually second by second could the extracted scenarios be assessed as true or false, which would waste huge amounts of time and is almost impossible to conduct. As the table shows below, what has been done is to choose a small set as a representative of the whole data set and manually check the ground-truth label according to the video, deriving a precision rate based on this and viewing it as the precision of the whole data set. The reason that is accounting for this is that NGSIM data is well-organized and friendly for classification, hence the TTC values mainly influence the number of extracted scenarios rather than affect the precision rate a lot overall.

TTC	left cut in			left cut-out			right cut-in			right cut-out			Overall		
	TP	FP	Precision	TP	FP	Precision	TP	FP	Precision	TP	FP	Precision	TP	FP	Precision
3	31	3	0.91176471	25	3	0.89285714	9	1	0.9	7	1	0.875	72	8	0.9

Figure 3.4 Precision of rule-based method implementation on NGSIM data.

Here the precision score is used, which is defined as the formula below:

$$precision = TP/(TP + FP) \tag{3.2}$$

Where TP is the number of all true positive samples, and FP is the number of all false positive samples. Positive samples are the samples that have been classified as driving scenarios, and true/false indicates their status as correct or not.

The tuning results of threshold TTC for NuScenes and NGSIM are shown in the tables below:

TTC	Objects cut-in			Fast approaching			Objects cut-out			Overall		
	TP	FP	Precision	TP	FP	Precision	`TP	FP	Precision	TP	FP	Precision
2	3	4	0.42857143	14	10	0.58333333	0	0	0	17	14	0.54839
3	4	5	0.44444444	27	26	0.50943396	1	0	1	32	31	0.50794
4	7	7	0.5	35	38	0.47945205	2	1	0.66667	44	46	0.48889
5	10	13	0.43478261	50	71	0.41322314	4	1	0.8	64	85	0.42953

Figure 3.5 TTC tuning for Nuscenes.

TTC	left cut in	left cut-out	right cut-in	right cut-out	Precision
2	413	313	96	96	0.9
3	523	437	126	125	
4	560	497	136	135	
5	571	535	141	138	

Figure 3.6 TTC tuning for NGSIM.

Although a TTC of less than 3 seconds is considered a sign of a dangerous situation [36], in our threshold experiments, we found that if we want to obtain more scenarios, we may still have to make some compromises on the threshold setting. Because extreme cases that occur in real life are rare.

When handling NuScenes data, $TTC = 3$ is selected for picking up fast-approaching scenarios in the rule-based method because it could not only provide comparatively high precision but also could guarantee enough number of extracted intriguing situations whereas, for cut-in and cut-out, $TTC = 5$ is chosen as the threshold instead of $TTC = 3$ in that if TTC is way too low, the desired scenarios would be excluded mistakenly.

To summarize, the standard of choosing TTC as the threshold value is to strike a balance between the number and precision. When it comes to NGSIM data, the situation is much more optimistic than that of NuScenes data due to the mentioned superiority of NGSIM data. It turned out that even though the TTC is as low as 2 seconds, there are still 918 scenarios extracted and as TTC grows, the number of extracted samples increases as it should be. Overall, based on a randomly selected mini-data set, the precision is around 90%.

4 Clustering

In this chapter, a machine-learning method will be implemented in both datasets. The result should be clusters that represent scenarios being needed. The performance of the algorithm will be evaluated and compared with the rule-based method's result above to discover which tool is more suitable for scenario classification.

An ideal clustering result would be able to include most of the rule-based results and probably find more interesting scenarios. Even though it is not applicable for clustering to predict which group a trajectory belongs to, if clustering is proven to perform well, it can be used in other data sets' scenario mining tasks or improve rule-based results. Besides, a good-performing clustering method can be implemented in other datasets as well, without writing a rule-based method and tuning the parameter manually from the bottom. It should have a similar performance with similar data input.

Although Chapter 3 has generated some labels for trajectories, they don't cover all trajectories from the dataset. Hence, an unsupervised learning algorithm would be necessary for this case. Based on the literature review in Chapter 1, because there is no complete ground truth and a fixed number of categories, a density-based Clustering method is used.

4.1 Similarity measurement

The trajectory data being used is time series data with temporal-spatial information. In other words, it is high-dimensional data instead of a set of points that can be clustered only by looking at the points' coordinates. Therefore, to make trajectories able to be clustered, the similarity score or distance of each pair of trajectories must be generated and compared before implementing clustering algorithms [19].

More specifically, the density-based clustering algorithm takes a distance matrix as input, this will be made by similarity measurement algorithms. It's worth a mention that a similarity measure is the converse of a distance function [37], which means the higher the similarity score, the closer the distance. The similarity score is a measure that ranges in [0,1], therefore it's possible to convert the similarity score to distance by subtracting it from 1.

This part will cover two ways of similarity measurement and design a principally more suitable partitioning method for NuScenes trajectory data. Finally, the distance matrix should be made after this sub-chapter, and clustering preparation will be done.

4.1.1 DTW

Instead of comparing the whole trajectory directly, DTW loosens limits on the time axis, the trajectory could be zoomed locally along the time axis by repeating the same recorded point of one trajectory and computing their Euclidean distances to various points in another trajectory. In this way, it could get the minimum distance between two trajectories.

A DTW algorithm takes two time-series $x = (x_0, \dots, x_{n-1})$ and $y = (y_0, \dots, y_{m-1})$ of respective lengths n and m. A Python library *tslearn* [38] is used, where the formula of DTW is demonstrated as:

$$DTW(x, y) = \min_{\pi} \sqrt{\sum_{(i,j)\in\pi} d(x_i, y_i)^2} \tag{4.1}$$

Where $\pi = [\pi_0, \dots, \pi_K]$ is a path from $\pi_0 = (0,0)$ to $\pi_K = (n-1, m-1)$ in the matrix whose elements represent each $d(x_i, y_i)$, the Euclidean distance of x_i and y_i. Each path will represent a distance between two time series, the DTW algorithm is trying to find the minimal distance shown in Figure 4.1 left. In this process, i doesn't have to have the same value as j, hence the time axis will be scaled, and the time series will be aligned as shown in Figure 4.1 right.

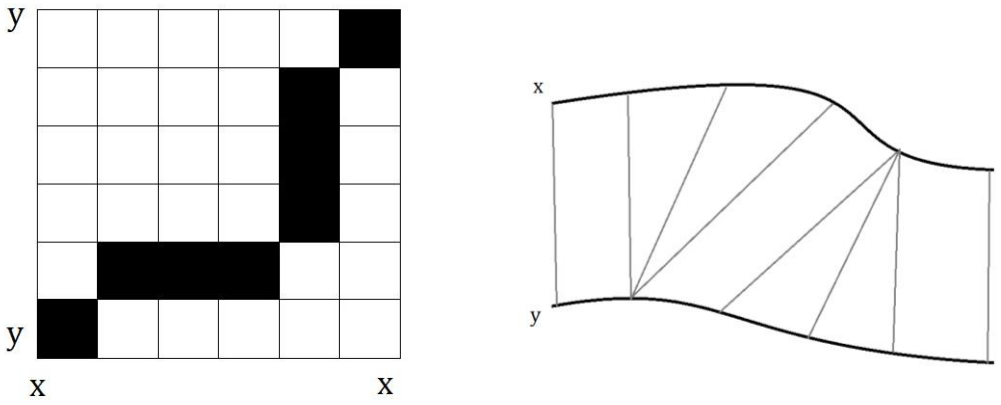


Figure 4.1 DTW illustration. Left: Optimal path. Right: Correspondence between recorded points.

In practice, DTW is used to measure x and y coordinate sequences' distance respectively for each pair of trajectories due to the limit of one-dimensional input of the algorithm. An average distance would be calculated after both x-direction-distance and y-direction-distance are generated. Finally, a distance matrix of shape (z, z) would be generated, where z is the total number of trajectories, and each element of the matrix $d(i, j)$ represents a distance value corresponding to the trajectory L_i and trajectory L_j, $i, j \in [0, z]$.

Since the distance matrix is large and the calculation is very time-consuming, it would be normalized to the range $[0,1]$ and stored in a greyscale .png file as shown below. Each pixel is a distance value between 0 and 1. The darker the color, the more similar the two trajectories are. The solid black color of the diagonal indicates that each trajectory is identical to itself (distance 0, similarity 1).

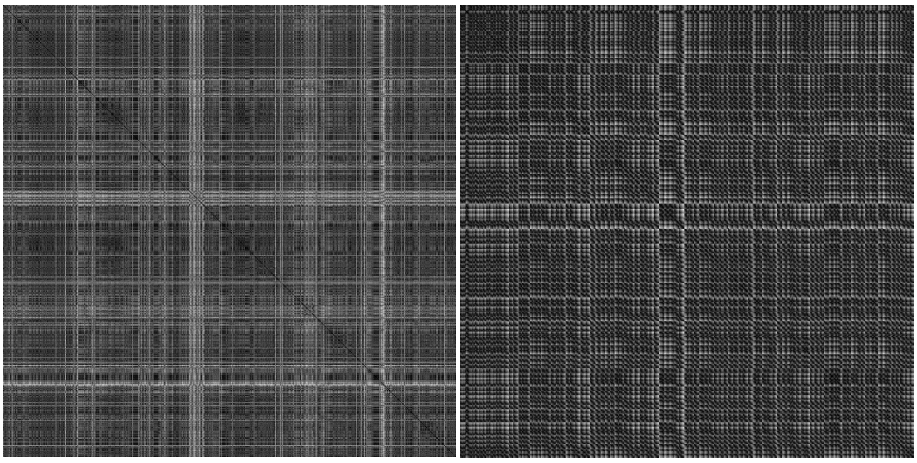


Figure 4.2 Distance matrix generated by DTW. Left: NuScenes. Right: NGSIM.

4.1.2 LCSS

While DTW is an algorithm efficient for comparing the whole trajectory, it is sensitive to noise in the trajectory [19]. NuScenes data was collected by an ego vehicle and sensors mounted on the vehicle, the trajectories happened at many different places in the city, which leads to a high overall dissimilarity of trajectories and many noises. This thesis will try 2 different ways to deal with this problem, Longest Common Subsequence (LCSS) algorithm, and sequence partitioning. This part covers LCSS first. The same method will be implemented on NGSIM data as well to compare the effect.

LCSS is different from calculating the distance between trajectories. On the contrary, LCSS's output represents the similarity of two trajectories, which means the proportion of the longest similar subsequence to the length of the shorter sequence in the two time-series. It's always derived recursively through equation 1.2.

Similarly, the LCSS algorithm from *tslearn* is also implemented, and a distance matrix will be generated as well. But the elements need to be converted from similarity to distance by subtracting from 1.

As mentioned in Chapter 1, thresholds σ and ϵ for lateral and longitudinal distance need to be chosen. This is not an intuitive choice, as there may be a wide range of differences between trajectories. The selection of the threshold needs to exclude significantly different segments while retaining small differences as much as possible. A few tryouts are listed in Figure 4.3 for NuScenes. σ and ϵ are set the same as a value from $[0.5, 1, 2{,}5]$.

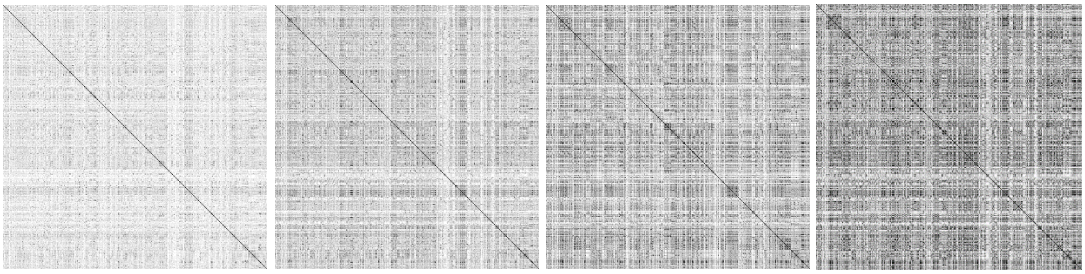


Figure 4.3 NuScenes input distance matrix generated by LCSS, threshold setting from left to right: 0.5, 1, 2, 5.

For NGSIM, σ and ϵ are set the same as a value from $[0.5, 1, 2, 5]$.

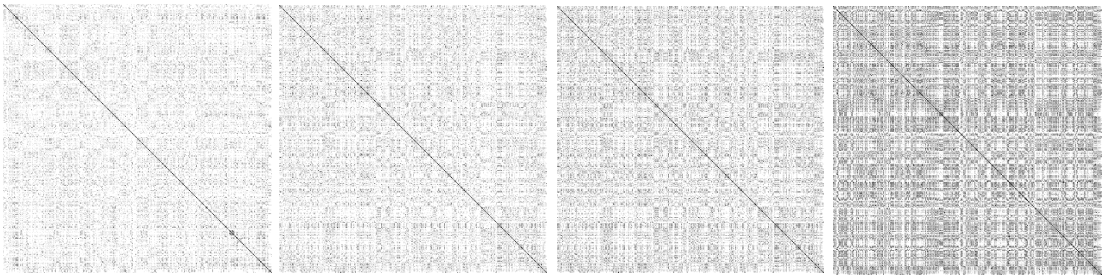


Figure 4.4 NGSIM input distance matrix generated by LCSS, threshold setting from left to right: 0.5, 1, 2, 5.

Being compared with Figure 4.2, these images look very different from the distance matrices generated by DTW, it seems brighter and has lower contrast. This is due to the feature of the data source, i.e., NGSIM was collected by stationary cameras, and the coordinates were based on a local absolute coordinate system, also NuScenes data is based on the ego vehicle as the center of the coordinate system. This kind of coordinate system leads to a consequence of a numerically large dissimilarity of the trajectories. For example, a vehicle driving straight in lane 1 may have a similar shape of trajectory as the vehicle driving straight in lane 6, but the distance measurement would still be high unless the LCSS's thresholds are set high correspondingly.

However, scenarios also happen based on their location. Setting a high threshold may improve the overall contrast but loses the ability of the algorithm to distinguish adjacent trajectories. Thus, even with relatively low contrast, a low threshold is chosen to identify trajectories that are close to the region of interest and exclude trajectories that are too far away. σ and ϵ will be finally decided in Chapter 4.2.4.

4.1.3 Trajectory partitioning

As described earlier, the other way of partly solving trajectory comparison is partitioning on trajectories.

In principle, the partitioning of trajectories is slicing trajectories into segments. For NGSIM data, partitioning is done in the data preprocessing part by only extracting the information that is close to the lane-change maneuvers happening time. The trajectories from NGSIM in this thesis all represent a similar period of driving.

Therefore, considering the data was not collected at the same place and trajectories don't have the same window that represents the objects' moving pattern, choosing the window of slicing would be important for NuScenes. A sliding window is needed to raise the possibility of getting the interesting parts of trajectories.

By looking into the data, most scenarios could happen in less than 15-20 trajectory points (7.5-10 seconds), to keep more driving information while filtering useless parts, the sliding window is determined to be 20 points. To cover the whole trajectory (maximum 41 valid data points) and save the amount of computation, the sliding window will set a total of 5 with a step size of 5 on the entire trajectory. In a trajectory $L = (l_0, l_1, l_2, \dots, l_{40})$, the sliding window would be $W = (l_{0+5*k}, l_{1+5*k}, \dots, l_{19+5*k})$, $k \in [0,1,2,3,4]$, where $l_i = (x_i, y_i)$.

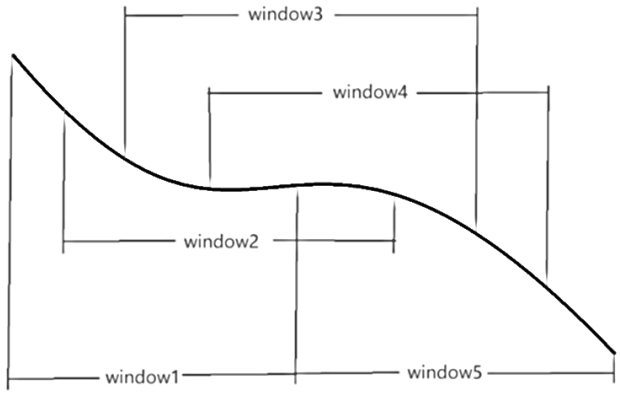


Figure 4.5 Sliding windows.

In this case, each trajectory will be partitioned into 5 segments. Segments of different trajectories will be compared using DTW and LCSS, which generates 25 distances in total for one comparison of a pair of trajectories. In order to reduce the impact of noise—that is, pairs of segments with large dissimilarity or distance—on the similarity measure of the entire trajectory, this paper will discard some distances with large values. Similarly, in order to avoid some segment pairs with quite high similarity influencing the judgement, distance with small values should also be discarded. The specific operation is, in the distance sequence $D = (d_0, d_1, d_2, \dots, d_{24})$ arranged in ascending order, the distance value of this pair of trajectories is determined as:

$$D_{i,j} = \sum_{n=z*5}^{z*5+4} d_n + \sum_{n=z*5+5}^{z*5+9} d_n / 5 \Big/ 6, z = 0,1,2,3 \quad (4.2)$$

Where i, j is the index of the two trajectories being compared. $D_{i,j}$ gives similar parts more weight than the dissimilar parts, thus noises will be partly excluded.

There is no doubt this way of partitioning is not the best. In fact, it is very likely that this method cannot detect the most similar segment pairs. But this is a compromise of computational speed and accuracy because doing trajectory partitioning would increase the amount of computation exponentially.

4.2 HDBSCAN clustering

HDBSCAN [21] is introduced as a Python library in this thesis. It can take a distance matrix as the input and make it precomputed metric. The clustering result will be generated based on the distance matrix. Note that the distance matrix should always be a square matrix.

4.2.1 Evaluation metric

Given the rule-based results, the clustering result will be compared against it to check if the scenario is still in the same class in the clustering result. The clustering model will be considered successful when all same-class scenarios are in one class of the clustering results, and the class has only a few false positives. Recall, precision, and F_β-score will be used as the evaluation metric of the clustering result.

$$Recall = \frac{TP}{TP+FN} \tag{4.3}$$

$$Precision = \frac{TP}{TP+FP} \tag{4.4}$$

$$F_\beta = (1+\beta^2) * \frac{Precision*Recall}{(\beta^2*Precision)+Recall} \tag{4.5}$$

Where TP is the number of all true positive samples, FP is the number of all false positive samples, FN is the number of all false negative samples. The samples of the same class may be distributed in different clusters in the clustering result, and the cluster with the largest number of samples of the same class in rule-based results will be regarded as the true cluster. Since successfully recognizing scenarios in one cluster is more significant than only having part of rule-based results in one cluster, recall should have a higher weight than precision, β is set as 2 here. It should have some tolerance for misclassifying samples other than rule-based results in the positive cluster, since not all trajectories have been reviewed, it cannot be fully guaranteed that rule-based results are the only scenarios that appeared in either dataset.

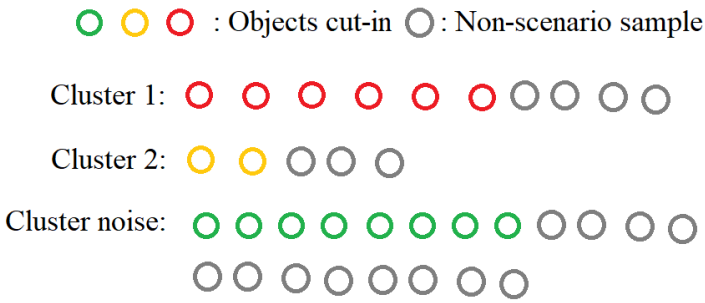


Figure 4.6 Clustering result example.

For instance, in the figure above, cluster 1 has 6 cut-ins, cluster 2 has 2, and the noise cluster has 8. The noise cluster has the greatest number of cut-ins, but the noise will not be taken into account as a positive classification. Therefore, the true positive cluster would be cluster 1, it has 6 true positive samples out of 16 in total. The recall would be

$6/16 = 0.375$. Since cluster 1 is regarded as a positive cluster, the total amount of positive samples is 10, and the precision would be $6/10 = 0.6$. Consequently, the F_2-score would be $F_2 = (1 + 2^2) * \frac{0.6*0.375}{2^2*0.6+0.375} = 0.405$.

4.2.2 Input setting

For NuScenes data, it is also worth mentioning that the entire trajectory set is not considered as input. This thesis only retains the scenes that the rule-based result scenarios originally belong to. This is because considering that the rule-based classification results only account for a small part of the overall sample size (less than 0.5%), if all trajectories are included for clustering, a large number of scenario-irrelevant trajectories will be included in the overall sample space, thus negatively affecting the classification results. Therefore, this thesis considers excluding scenes that do not contain scenarios and reducing the total number of clustering input samples. This will increase the calculation speed of the distance matrix and theoretically improve the performance of clustering to a certain extent.

Finally, only 2283 trajectories will be taken into account, 81 of them are classified as scenarios in rule-based results. Among these 81 scenarios, after manual checking, 40 are recognized as correct classification results and will be used as ground truth. Therefore about 1.75% of trajectories of 2283 are scenarios. Regarding fast-approaching scenarios, in order to make them more distinguishable, they are further split into three subcategories, app_left, app_right, and app_front, which means after approaching, the ego vehicle overtakes the object while the object is on the left, right, or the ego never overtakes. Cut-in and cut-out were also supposed to be further split, but to keep more samples in a class, their labels are kept. The ground truth of 40 labels will be compared with clustering results to generate evaluation metrics.

For the NGSIM data, although each trajectory is only intercepted at most 80 points (8 seconds) since the total amount of data is 86074 trajectories, this is still a huge number for the construction of the distance matrix and the clustering itself. The sample space needs to be reduced accordingly.

As mentioned in Chapter 2, these trajectory samples are extracted based on lane-changing behavior, so there may be several trajectories belonging to the same vehicle. However, since the ego vehicle will change every time, even if the trajectory comes from the driving behavior of the same vehicle, their positions, and shapes reflected in the relative coordinate system are still different. Therefore, this thesis chooses the first 10000 trajectories as input of clustering, which includes 160 scenarios of the label “left cut in”, “right cut in”, “left cut out”, and “right cut out”. In the extracted NGSIM data, because each lane-change behavior will be checked, every “ego cuts in” is an “object cut in” to another vehicle. Therefore, no distinction will be made between ego and objects, all labels belong to the category objects cut in and objects cut out. Left and right indicate the moving direction of the ego vehicle, i.e., “left cut in” means the ego vehicle changes to the left lane and cuts in front of an object, “left cut out” means the ego vehicle cut out from an object behind and changes to the left lane.

Table 4.1 Input settings.

Dataset	NuScenes	NGSIM
Maximum trajectory length	41 points	80 points
Maximum recording time	20 seconds	8 seconds
Trajectory numbers	2283	10000
Scenario numbers	40	160
Scenario proportion	1.75%	1.60%

4.2.3 Hyperparameter tuning

The $\overline{F_2}$-score will be used as the only numerical evaluation metric for hyperparameter tuning, it will be computed by the following formula.

$$\overline{F_2} = \sum(F_{2_i} * sum(i)) \,/\, \sum sum(i) \tag{4.6}$$

Where i is one scenario, $sum(i)$ is the total number of the scenario i in the ground truth. Thus, a combination of hyperparameters gives the highest F_2-score will be chosen as the final setting for clustering.

A hyperparameter optimization framework Optuna [39] is used to choose the best hyperparameter combination. Users can delf-define the search space and choose the search method. Optuna will judge the changes in performance indicators brought about by each combination of hyperparameters and evaluate the importance of each hyperparameter. In addition, it will choose the hyperparameter change direction that will lead to positive performance improvement, thereby reducing the number of trials and calculations. Eventually, within a limited number of trials, the algorithm chooses the combination of hyperparameters that provides the best evaluation metric.

In this thesis, the search space is defined respectively for four kinds of input settings of HDBSCAN clustering, distance matrix built by DTW, DTW with partitioning, LCSS, and LCSS with partitioning. Four hyperparameters will be considered to be tuned, they are shown in the table below.

Table 4.2 Hyperparameter search space.

Name	Meaning	Search space	Search rule
min_cluster_size	The minimum size a final cluster can be, the higher this parameter is, the bigger the clusters will be.	[2, 15]	Integer, step size = 1

min_samples	The minimum number of neighbors needed for a core point to detect. The higher this parameter is, the more points will be considered noise.	[2, 15]	Integer, step size = 1
cluster_selection_epsilon	Clusters below this threshold will be merged. The epsilon used for clustering is at least this number.	[0.0001, 10]	Float, log
cluster_selection_method	‘eom’ tends to choose a few big clusters and then choose smaller ones. ‘leaf’ gives more small homogeneous clusters. Both of them give variable density clusters.	[‘eom’, ‘leaf’]	Category, pick one of two

Where if the search rule is step size, the algorithm will divide the search space equally according to the step size, and then randomly try any one of the possibilities. If the log flag is true, the search space will be divided in the log domain instead of the linear domain.

4.2.4 Tuning results and discussion

100 attempts were made for each optimization, the best set of hyperparameters was selected, and the corresponding F_2-score value was displayed. The objective value in Figure 4.7 is the F_2-score of this trial. From this plot it can be seen that the objective value has reached the top and can barely grow further before 50 trials, so 100 trials are set for the rest tuning processes to save computational time. This hyperparameter tuning process will make sure that the performance of clustering cannot be further improved with only changing hyperparameters.

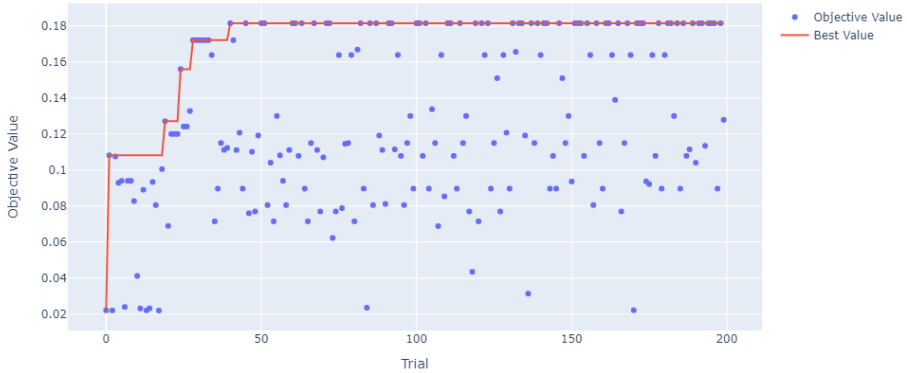


Figure 4.7 Hyperparameter optimization history plot.

First, the LCSS threshold σ and ϵ need to be determined. Multiple thresholds have been tested; the results are listed below:

Table 4.3 LCSS threshold testing.

Dataset	Threshold	Best F2-score

NuScenes	0.5	0.175499404
	1	0.181564038
	2	0.147443534
	5	0.047352862
NGSIM	0.5	0.072291867
	1	0.119762641
	2	0.080596246
	5	0.062432712

Therefore, σ and ϵ are set as 1 according to the results above. Even if the result cannot be improved a lot, it can still be a baseline to be compared with. The following content will be based on the condition $\sigma = \epsilon = 1$.

NuScenes

Different values of z were tried in Equation 4.2, but since the algorithm did not provide a sufficiently discriminative distance matrix when $z = 0$ (most of the trajectories have some similar segments and the 5 smallest distances have very low values), the results of this attempt are not presented in the table below. z will be shown in the column of the building rule.

Table 4.4 Tuning result of NuScenes input.

Distance matrix algorithm	Building rule	Best F_2-score	Best hyperparameter
LCSS	$z = 1$	0.043654135	'min_cluster_size': 9, 'min_samples': 14, 'cluster_selection_epsilon': 0.01112, 'cluster_selection_method': 'leaf'
	$z = 2$	0.191149824	'min_cluster_size': 3, 'min_samples': 5, 'cluster_selection_epsilon': 0.0007052, 'cluster_selection_method': leaf
	$z = 3$	0.177504006	'min_cluster_size': 2, 'min_samples': 13, 'cluster_selection_epsilon': 0.0002465, 'cluster_selection_method': 'leaf'
	No partitioning	0.181564038	'min_cluster_size': 5, 'min_samples': 7,

			'cluster_selection_epsilon': 0.0008821, 'cluster_selection_method': 'leaf'
DTW	$z = 1$	0.043654135	'min_cluster_size': 3, 'min_samples': 6, 'cluster_selection_epsilon': 6.9226, 'cluster_selection_method': 'leaf'
	$z = 2$	0.124264645	'min_cluster_size': 3, 'min_samples': 13, 'cluster_selection_epsilon': 0.0005508, 'cluster_selection_method': 'leaf'
	$z = 3$	0.166612258	'min_cluster_size': 2, 'min_samples': 14, 'cluster_selection_epsilon': 0.0001083, 'cluster_selection_method': 'leaf'
	No partitioning	0.158638688	'min_cluster_size': 2, 'min_samples': 12, 'cluster_selection_epsilon': 0.001080, 'cluster_selection_method': 'leaf'

NGSIM

Since in the NGSIM data input, partitioning has been done manually by only extracting the time frame around the lane-change behavior, z will not be set.

Table 4.5 Tuning result of NGSIM input.

Distance matrix algorithm	Best F_2-score	Best hyperparameter
LCSS	0.119762640	'min_cluster_size': 3, 'min_samples': 3, 'cluster_selection_epsilon': 8.5620e-05, 'cluster_selection_method': 'leaf'
DTW	0.303631634	'min_cluster_size': 3, 'min_samples': 6, 'cluster_selection_epsilon': 3.5217e-05, 'cluster_selection_method': 'eom'

As the results showed, in general, using the distance matrix obtained by DTW as input can achieve a higher F_2-score, while clustering performs better in the input provided by NGSIM. Several comparisons for the data will be discussed below:

1) As the z increases, the F_2-score also increases.

Having trajectories partitioning in the NuScenes dataset, each comparison between two trajectories would have 25 distance values. z affects the value position of the final distance among this 25-distance sequence. The results show that a larger z value is better because most of the values in the sequence are relatively small at the numerical

level. So if the chosen values are the first few of the sequence, the discrimination between trajectories will be reduced.

Part of the reason for this result is that after partitioning, the trajectory segments are shorter and are more likely to be judged as similar by the algorithm and given a small distance value. For example, in Equation 1.2, $\min(n, m)$ is much lower than the whole trajectory, the proportion of similar subsequences in $\min(n, m)$ will also tend to be higher.

2) Overall, the input distance matrix provided by DTW resulted in a higher F_2-score.

First, this could be due to the threshold setting of LCSS. Since the construction of the distance matrix consumes a lot of computing power, the threshold optimization of LCSS is limited. Therefore, the threshold setting is very likely not the optimal setting, which may bring a certain loss of clustering performance.

Second, in NGSIM data, DTW shows a great performance advantage over LCSS, which is related to the data characteristics of NGSIM. Since there is no data scaling, the vehicle trajectories of NGSIM will be distributed in different positions of the coordinate system according to their lanes. At the same time, the coordinate system is established by taking the vehicle with a lane-changing maneuver as the reference point. Logically speaking, the cut-in and cut-out scenes will only exist in the same lane as the reference vehicle and will be located behind.

However, LCSS performs similar-dissimilar binary judgments on the points in the trajectory according to the threshold setting, which will cause the two trajectories to be judged as completely similar if the distance between all point pairs of the trajectory is smaller than the threshold, and vice versa. Thus, LCSS cannot recognize the shape of trajectories that are too close or too far, which is critical in unscaled data. DTW will calculate the distance between two trajectories according to its algorithm rules. Trajectory pairs with close distances will have the same magnitude and different distance values, which helps to identify trajectory shapes within a certain range.

3) HDBSCAN performs better on NGSIM, but not well overall.

NuScenes data has only a few ground truth samples for cut-in and cut-out, the amount is not enough for clustering to identify the vehicle's moving pattern. Besides, in order to keep more samples in one class to reduce the impact of lack of samples, cut-in and cut-out were not further split into left and right classes. The difference between left and right lane changes may interfere with the accuracy of clustering.

On the other hand, as mentioned above, the dissimilarity of the trajectories in NGSIM is more reflected in the difference of the lane, that is, the difference of the position in the local coordinate system, rather than the discrimination of its shape. Because the distance between lane positions may be tens of meters, the distance value brought by different driving maneuvers in the same lane will not be so distinct. It's much easier for clustering to classify locations and hence better evaluation metrics appear. While in NuScenes all data represent urban driving environments, objects are closer to the base point (ego vehicle), and the location of trajectories is not as informative as NGSIM.

4.3 Data scaling effects

As mentioned above, the large numerical difference in trajectory coordinates due to different lanes will affect the discrimination of the trajectory shape by clustering. To solve this problem, and check the effects of location information, data scaling will be used. The main purpose is to limit the numerical range of the trajectory points to a similar interval.

Python library *tslearn* [38] is used to scale the data. After scaling, all points $c_n = (x_n, y_n)$, from the trajectory t_i will fulfill the conditions below:

$$\mu_{t_i} = 0, \qquad \sigma_{t_i} = 0 \tag{4.7}$$

Where μ_{t_i} is the mean of t_i, and σ_{t_i} is the standard deviation of t_i.

As shown in Figure 4.6, two trajectories could represent the same maneuver. But because of the different locations, they cannot be recognized as similar without data scaling. Scaling could help the similarity measurement algorithms focus on the trajectories' shape and their behavior patterns.

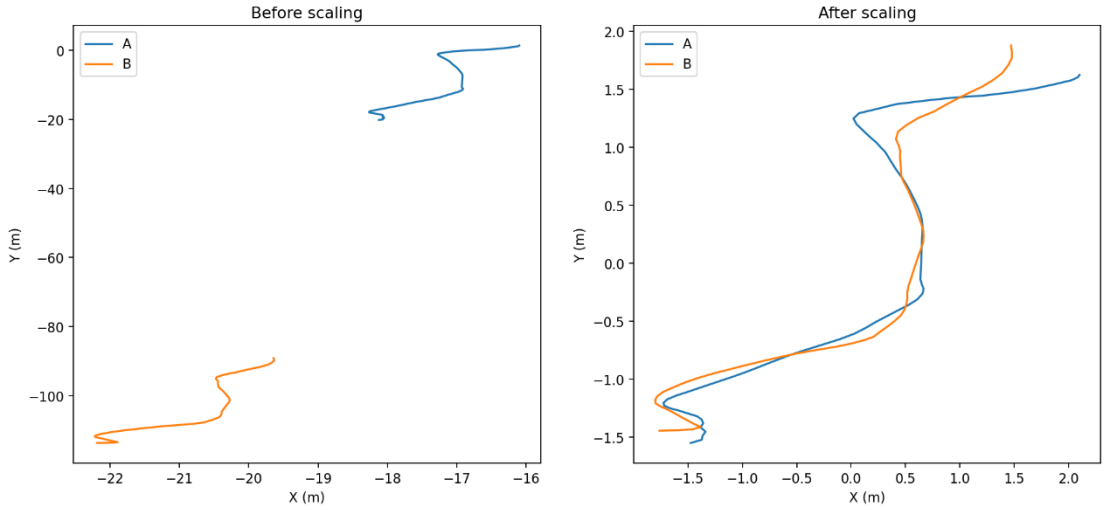


Figure 4.8 Scaled data and unscaled data comparison.

HDBSCAN clustering results after scaling are shown in the following table:

Table 4.6 Result of scaled NuScenes input.

Distance matrix algorithm	Building rule	Scaling	Best F_2-score	Best hyperparameter
LCSS	z = 3	Yes	0.02521	'min_cluster_size': 10, 'min_samples': 11, 'cluster_selection_epsilon': 0.02225, 'cluster_selection_method': 'eom'

		No	0.1775	'min_cluster_size': 2, 'min_samples': 13, 'cluster_selection_epsilon': 0.0002465, 'cluster_selection_method': 'leaf'
	No partitioning	Yes	0.04975	'min_cluster_size': 6, 'min_samples': 15, 'cluster_selection_epsilon': 0.0001762, 'cluster_selection_method': 'leaf'
		No	0.1816	'min_cluster_size': 5, 'min_samples': 7, 'cluster_selection_epsilon': 0.0008821, 'cluster_selection_method': 'leaf'
DTW	z = 3	Yes	0.1695	'min_cluster_size': 2, 'min_samples': 12, 'cluster_selection_epsilon': 0.0001527, 'cluster_selection_method': 'leaf'
		No	0.1666	'min_cluster_size': 2, 'min_samples': 14, 'cluster_selection_epsilon': 0.0001083, 'cluster_selection_method': 'leaf'
	No partitioning	Yes	0.1724	'min_cluster_size': 2, 'min_samples': 7, 'cluster_selection_epsilon': 0.0001462, 'cluster_selection_method': 'leaf'
		No	0.1586	'min_cluster_size': 2, 'min_samples': 12, 'cluster_selection_epsilon': 0.001080, 'cluster_selection_method': 'leaf'

Table 4.7 Result of scaled NGSIM input.

Distance matrix algorithm	Scaling	Best F_2-score	Best hyperparameter
LCSS	Yes	0.1144	

			min_cluster_size': 6, 'min_samples': 3, 'cluster_selection_epsilon': 0.0001498, 'cluster_selection_method': 'eom'
	No	0.1198	'min_cluster_size': 3, 'min_samples': 3, cluster_selection_epsilon': 8.5620e-05, 'cluster_selection_method': 'leaf'
DTW	Yes	0.03031	min_cluster_size': 2, 'min_samples': 6, 'cluster_selection_epsilon': 4.7709e-05, 'cluster_selection_method': 'leaf'
	No	0.3036	'min_cluster_size': 3, 'min_samples': 6, 'cluster_selection_epsilon': 3.5217e-05, 'cluster_selection_method': 'eom'

The results show that data scaling cannot bring massive performance increments, and the clustering results are still not satisfying. Even at certain times, data scaling could bring a negative impact on clustering. It seems if the location information of the trajectories is eliminated, the clustering still cannot recognize the driving behavior from the trajectories' shape. This could be because of the input data's quality. But if the clustering recognizes location more, it could be discussed further if other clustering algorithms from centroid clustering methods work better.

4.4 Different clustering algorithms implementation

To explore the reason for bad performance, this thesis has also tried different types of clustering algorithms, and the results of the attempts are shown in the table below, where each algorithm has implemented the hyperparameter tuning method as the same as Chapter 4.2.3 explained but with different hyperparameters. Among these algorithms, K-means and Gaussian Mixture Model (GMM) are centroid-based flat clustering models, hierarchical clustering is centroid-based but a hierarchical model, and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is the representative of density-based flat clustering model. HDBSCAN is a density-based hierarchical model, these models basically cover the rest of the classic types of clustering. The distance matrix generated by DTW without partitioning and scaling is used as the input.

Table 4.8 Result of different clustering algorithms.

Dataset	Algorithm	Best F_2-score	Hyperparameter choosing

NuScenes	K-means	0.1688	n_clusters': 95, 'init': 'random'
	GMM	0.145	'n_components': 89
	Hierarchical	0.133	n_clusters': 94, 'linkage': 'complete'
	DBSCAN	0.09566	min_samples': 2, 'eps': 0.004001
	HDBSCAN	0.1586	min_cluster_size': 2, 'min_samples': 12, 'cluster_selection_epsilon': 0.001080, 'cluster_selection_method': 'leaf'
NGSIM	K-means	0.4119	n_clusters': 94, 'init': 'random'
	GMM	0.3827	'n_components': 66
	Hierarchical	0.3885	n_clusters': 96, 'linkage': 'average'
	DBSCAN	0.4653	min_samples': 7, 'eps': 0.0001352
	HDBSCAN	0.3036	min_cluster_size': 3, 'min_samples': 6, 'cluster_selection_epsilon': 3.5217e-05, 'cluster_selection_method': 'eom'

Apparently, HDBSCAN does not provide the best metric, this could be figured out in the following figures.

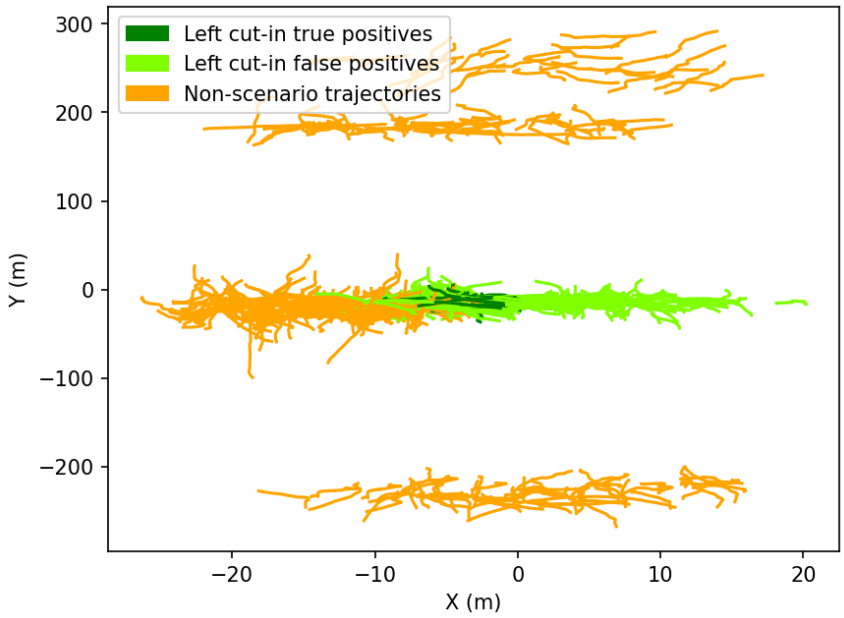


Figure 4.9 Parts of NGSIM K-means clustering result.

This plot partly shows how K-means clusters classify trajectories. Although these results are based on the distance matrix instead of the original trajectories, the clustering pattern could still be seen. Large position differences lead to large distance values for trajectory pairs, making it easy for trajectories with close occurrences to be classified into one class. This is the best application scenario of K-means. It can iteratively find the best center point of each category to classify the overall position space. Because the place of centroids is random, the metric of K-means could be affected.

Even if the label is not marked, the four categories of the orange trajectories in the figure can be easily recognized by the human eye, they are distributed by their different y coordinates. At the same time, it can be seen that there are indeed similar-shaped trajectories in each cluster, but the similar shape is not the main feature of this cluster.

However, it's very hard to argue that these performance gains are meaningful for driving scenario classification tasks. Because locations can be simply filtered by a rule-based method, with fewer parameters, computational time, and higher accuracy. The comparison will be further discussed in Chapter 5.

4.5 Result analysis

1) Overall, the F_2-score is bad, clustering does not perform well in scenario classification in NuScenes and NGSIM data.

It must be admitted the clustering implementations that have been tried did not give a convincing result. The highest F_2-score that the result has is less than 0.5. This result means that clustering cannot mine as much scenario information as the rule-based method, and there is no way to achieve high accuracy.

2) The reason for clustering's bad performance is the dissimilarity among trajectories.

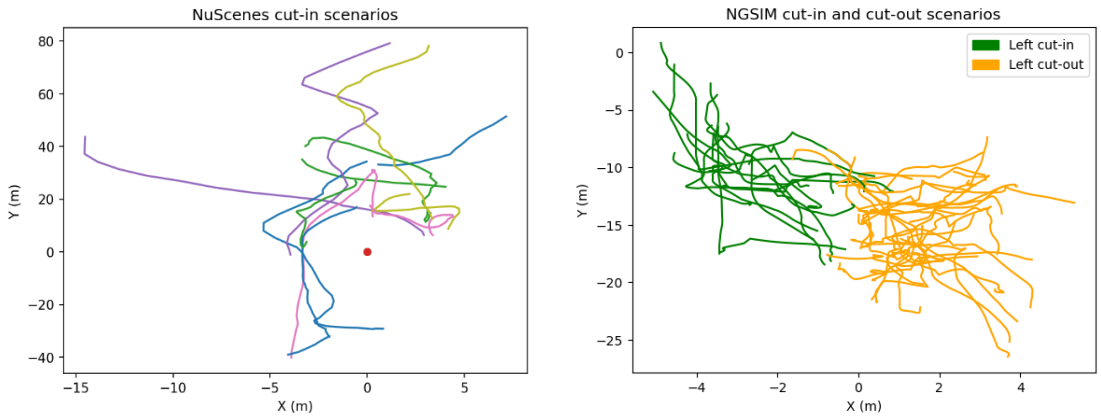


Figure 4.10 Trajectory examples of both datasets.

As shown in the figure above, the cut-in scenarios' trajectories from NuScenes are completely different from each other. To make each trajectory recognizable, they are drawn with different colors. All of them are from the rule-based result and have been checked manually. Same as NuScenes, NGSIM does not have similar cut-in and cut-out either, but they do locate at different relative places.

The reason is that as relative trajectories, these objects may have thousands of moving patterns from the base point or the ego vehicle's view. For instance, for the ego vehicle, other vehicles may cut in 60 degrees from the front right, or 80 degrees from the front left, both of which are called cut-in scenarios. In this case, the "cut-in" scenario can be infinitely differentiated. However, in this thesis, it is only divided into two categories according to the left and right in the clustering of NGSIM, but the trajectories in these two categories are still not similar enough to be identified from the total sample set by similarity measurement.

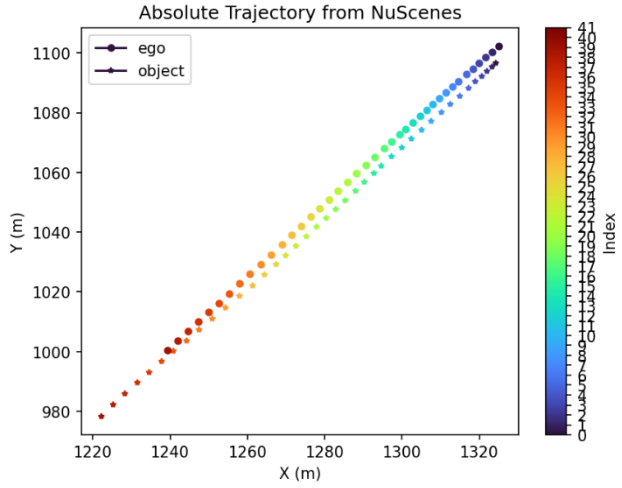


Figure 4.11 Absolute trajectory of a cut-in.

An example of the absolute trajectory of cut-in is shown in Figure 4.10, which represents the cut-in scenario by showing the correlation of the points. But this cannot be the input of the clustering model, it's hard to identify both trajectories' shapes and their correlation and generate the distance matrix. Besides, the absolute trajectory can have any direction in NuScenes since the data was collected in the whole city, this will make the similar problem discussed above.

3) The hyperparameter tuning, distance matrix, and clustering algorithm choosing can increase the evaluation metric to some extent, but it lacks meaning.

Although clustering is bad at identifying scenarios by recognizing the shapes of trajectories, if the location information is well recognized by using the appropriate way to build the distance matrix, the F_2-score can be higher. But this is not the aim of having a clustering model, filtering location can be easily done by using a rule-based method.

5 Conclusion and outlook

This chapter will cover the main conclusion and outlook. It will also explain some specific limitations that have been met in the thesis.

5.1 Conclusion

1) Lane information is crucial for rule-based methods to identify scenarios.

One of the biggest differences between NuScenes and NGSIM is that NGSIM has the Lane_ID information, which indicates the lane to which the vehicle belongs at each moment. Once the Lane_ID is provided, lane change maneuvers can be easily extracted, which is the prerequisite for identifying cut-in and cut-out. Therefore, this information could massively reduce the constructed rules that aim for identifying the lane and correlation between two vehicles. NGSIM also provided the preceding vehicle ID and following vehicle ID, which are useful but also based on lane information. This information could nearly eliminate all cases of wrongly judging cars in other lanes as being in the ego vehicle's lane.

Having the lane information, the rule-based result's precision increased from around 0.5 to 0.9. There is no doubt that the road conditions are much better in NGSIM data since it was collected on a straight highway, while the impact of roads' curvature in NuScenes was magnified without lane information.

2) Relative trajectory is not suitable for trajectory clustering to classify driving scenarios.

This conclusion is made based on the analysis and discussion in Chapter 4, where it can be seen that relative trajectory is not helpful for driving pattern identification. It is for sure that trajectories contain much information on driving patterns, and lane change could also be recognized by looking at absolute trajectories. Then, because the critical driving scenarios for the autonomous driving test will be some interactions between vehicles, to involve the correlation between two vehicles, relative trajectories, and two respective absolute trajectories can be used.

An absolute trajectory is clearer than a relative trajectory, but two trajectories with time-spatial correlation cannot be processed by clustering or the distance matrix algorithms. If relative trajectories are used, one of the biggest differences it makes is that it reduces the size of the space in which the two vehicles move. This can otherwise cause small changes to be very noticeable on relative traces. For example, if a car is driving on the highway with continuous steering wheel fine-tuning, the behavior of driving in a straight line, if viewed from the perspective of a car traveling at the same speed, will become a swaying behavior of the car in place.

Additionally, relative trajectories may take on shapes that do not align with the actual direction of travel. For example, if a car changes lanes while its longitudinal speed decreases, it will be an act of approaching him from the front for the car behind. And if it is typical overtaking behavior, the following car will regard the car in front as moving away. In the above two examples, the lateral lane change behavior that should have

been paid attention to was interfered with by the longitudinal information, and the consistency of trajectories became lower. This could have been resolved using rules in a rule-based approach, however, trajectory clustering cannot do this. To sum up, since the speed and displacement direction of ego and object vehicles are varied, and the relative trajectory is the vector difference between the two, the resulting relative trajectory will also be varied, which makes it almost impossible for clustering model to view them as one cluster.

3) Rule-based method is more efficient for driving scenario classification.

First, the rule-based method is more precise for classifying scenarios. Two clustering algorithms with the highest F_2-scores are picked, and their precision scores are shown below. Obviously, the rule-based method has a higher precision score.

Table 5.1 Different cluster models applied for Nuscenes and NGSIM

Dataset	Algorithm	Precision
NuScenes	K-means	0.10
	HDBSCAN	0.16
	Rule-based	0.51
NGSIM	K-means	0.23
	DBSCAN	0.65
	Rule-based	0.90

Second, the cost of implementing clustering is higher. As discussed in Chapter 4, involving more location information of trajectories could help it cluster with a better evaluation metric, but this could be simply done using a rule-based method without wasting time building the distance matrix and doing hyperparameter tuning.

Third, uncertainty is greater for clustering. The rule-based method should be built based on the known information, while clustering is a method of unknown data mining. The results of the rule-based method will be more predictable than the results of clustering because the changes brought about by adjusting the threshold are intuitive and in line with conventional logic, which is fundamentally different from the hyperparameter tuning of clustering.

In summary, the rule-based method is a more efficient way to classify scenarios when facing a new dataset.

5.2 Limitations

All datasets are given with object detection results, and annotation will not be done by the thesis students, including both object annotation and trajectory annotation.

The NuScenes dataset does not have lane ID information and vehicle ID information, which leads to difficulty in recognizing lane-changing maneuvers, not to mention extracting cut-in and cut-out scenarios, since what could be used to extract scenarios is just coordinates. Besides, the NuScenes data is driving scenes on urban roads, so the driving scenes are more unexpected, and the road conditions are more variable, which further increases the difficulty in identifying scenarios. Also, all data were collected by the ego vehicle, in each scene, the scenario happening time could vary a lot. This is unlike the data from the stationary camera at an intersection, which could give trajectories that are all similar to each other. The NGSIM dataset being used is the reconstructed version, therefore it may cause some differences from the original version.

The thesis tremendously challenges the computer performance, since, among other things, data extraction and storage, distance matrix generation, clustering model training, hyperparameter tuning and so on should be done. All these mentioned tasks are computationally demanding, which requires both powerful CPU and GPU. It is pretty much infeasible by private laptops.

Unfortunately, there is not a large volume of target scenarios extracted by rule-based method, hence supervised machine learning such as neural network could not be applied in this case. The reasons for this are that the number of identified scenarios is not enough to detect key feature patterns existing in relative trajectories and that relative trajectories are too literally dissimilar to be learned with so few scenarios.

If the result is too many, they will not be fully checked. Take the NGSIM data as an example, the number of extracted scenarios is relatively large and it is almost improbable to check the accuracy one by one via watching the video, so what is done is to just make features of partial samples represent the whole samples. Apart from this, the indicator evaluating the extracted results could just be precision as there are so many trajectories that it is impossible to check samples predicted as negative exhaustively since the major portion is negative samples. Consequently, true negative and false negative could not be derived.

Different datasets have different structures and contain different information, for which rules of rule-based method should be set from scratch when it comes to different data. As a result, the generated rule-based method could not be blindly transferred to other datasets since they might be totally incompatible.

Fundamentally, clustering model is inherently not applicable to the relative trajectories. It is eventually realized that relative trajectories present huge dissimilarities in terms of visible patterns. It is absurd to force clustering models to classify trajectories based on shapes when it is not possible with even naked eyes.

5.3 Outlook

Further improvement

1) For the NuScenes dataset, if higher classification accuracy is needed for other research, there are two optional approaches to improve the recognition of vehicle behavior.

One method is to use a mature image semantic recognition algorithm to mark lane information, to assist in judging whether the front/rear vehicle of the Ego vehicle is in the same lane as the ego. Another method is to use the map that comes with NuScenes to integrate the map information with the trajectory to identify the curve. This may exclude interfering targets outside the curve. But it is undeniable that no matter which method is extremely laborious but not necessarily pleasing, it is best to replace the data set for research.

2) A scenario generation tool can be built based on the idea of relative trajectories.

For example, when a relative trajectory is obtained but the real absolute trajectory is not known, the vehicle lane change cut-in/cut-out behavior can be automatically generated according to the number of lanes. This will cover the case where two cars change lanes at the same time. This approach can exponentially expand the number of different scenarios for simulation.

Implementation of this thesis

1) This thesis can help companies clarify the information they need to gather when gathering new datasets for autonomous driving development validation, including lane information.

2) The scene data that this thesis has explored for the two datasets will be used in the following summer work. And it may be further applied to autonomous driving simulation.

6 References

[1] SAE International. (2021). Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. *SAE International Std. J3016*. doi:10.4271/J3016_202104.

[2] Kalra, N., & Paddock, S. M. (2016). Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94, 182-193.

[3] Huang, W., Wang, K., Lv, Y., & Zhu, F. (2016, November). Autonomous vehicles testing methods review. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)* (pp. 163-168). IEEE.

[4] Häger, G., Caesar, L., & Sun, D. (2019). NuScenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*.

[5] Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 3431-3440.

[6] Mao, J., Shi, S., Wang, X., & Li, H. (2022). 3D object detection for autonomous driving: a review and new outlooks. *arXiv preprint arXiv:2206.09474*.

[7] Junietz, P., Wachenfeld, W., Klonecki, K., & Winner, H. (2018, November). Evaluation of different approaches to address safety validation of automated driving. *In 2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (pp. 491-496). IEEE. doi: 10.1109/ITSC.2018.8569959

[8] Junietz, P., Bonakdar, F., Klamann, B., & Winner, H. (2018, November). Criticality metric for the safety validation of automated driving using model predictive trajectory optimization. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (pp. 60-65). IEEE.

[9] Ulbrich, S., Menzel, T., Reschka, A., Schuldt, F., & Maurer, M. (2015, November). Defining and substantiating the terms scene, situation, and scenario for automated driving. *In 2015 IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)* (pp. 982-988). doi: 10.1109/ITSC.2015.164.

[10] Zhang, M., Zhang, Y., Zhang, L., Liu, C., & Khurshid, S. (2018, September). DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering* (pp. 132-142).

[11] Zhao, D., Guo, Y., & Jia, Y. J. (2017). TrafficNet: An open naturalistic driving scenario library. *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. doi: 10.1109/itsc.2017.8317860.

[12] Tenbrock, A., König, A., Keutgens, T., & Weber, H. (2021, July). The conscend dataset: Concrete scenarios from the highd dataset according to alks regulation unece r157 in openx. In *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)* (pp. 174-181). IEEE.

[13] Demetriou, A., Alfsvåg, H., Rahrovani, S., & Chehreghani, M. H. (2020). A deep learning framework for generation and analysis of driving scenario trajectories. *arXiv preprint arXiv:2007.14524*.

[14] Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research, 9*(11).

[15] Martinsson, J., Mohammadiha, N., & Schliep, A. (2018). Clustering Vehicle Maneuver Trajectories Using Mixtures of Hidden Markov Models. *In 2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (pp. 3698-3705). doi: 10.1109/ITSC.2018.8569418

[16] Hoseini, F. S., Rahrovani, S., & Chehreghani, M. H. (2020). A generic framework for clustering vehicle motion trajectories. *arXiv preprint arXiv:2009.12443*.

[17] Ezugwu, A. E., Ikotun, A. M., Oyelade, O. O., Abualigah, L., Agushaka, J. O., Eke, C. I., & Akinyelu, A. A. (2022). A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, *110*, 104743.

[18] Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd* (Vol. 96, No. 34, pp. 226-231).

[19] Yuan, G., Sun, P., Zhao, J., Li, D., & Wang, C. (2017). A review of moving object trajectory clustering algorithms. *Artificial Intelligence Review*, *47*, 123-144.

[20] Ankerst, M., Breunig, M. M., Kriegel, H. P., & Sander, J. (1999). OPTICS: Ordering points to identify the clustering structure. *ACM Sigmod record*, *28*(2), 49-60.

[21] McInnes, L., Healy, J., & Astels, S. (2017). hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, *2*(11), 205.

[22] Healy, J., & McInnes, L., (2018, October). HDBSCAN, Fast Density Based Clustering, the How and the Why. Presented at *PyData NYC 2018*, New York, NY. Retrieved from https://www.youtube.com/watch?v=dGsxd67IFiU&t=445s&ab_channel=PyData.

[23] Müller, M. (2007). Dynamic time warping. *Information retrieval for music and motion*, 69-84.

[24] Bergroth, L., Hakonen, H., & Raita, T. (2000, September). A survey of longest common subsequence algorithms. In *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000* (pp. 39-48). IEEE.

[25] Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., & Keogh, E. (2003, August). Indexing multi-dimensional time-series with support for multiple distance measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 216-225).

[26] Lee, J. G., Han, J., & Whang, K. Y. (2007, June). Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (pp. 593-604).

[27] Nanni, M., & Pedreschi, D. (2006). Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, *27*(3), 267-290.

[28] SAE International. (2018). Terms and Definitions Related to Testing of Automated Vehicle Technologies. *SAE International Std. J3016_201806*. doi: 10.4271/J3016_201806

[29] Scenario Categories for the Assessment of Automated Vehicles Version 1.7 from January 21, 2020 Erwin de Gelder (TNO), Olaf Op den Camp (TNO), Niels de Boer (NTU)

[30] H. Elrofai, J.-P. Paardekooper, E. de Gelder, S. Kalisvaart, and O. Op den Camp, "StreetWise position paper, scenario-based safety validation of connected and automated driving," TNO Integrated Vehicle Safety, www.tno.nl/STREETWISE, Tech. Rep., 2018.

[31] U.S. Department of Transportation Federal Highway Administration. (2016). Next Generation Simulation (NGSIM) Vehicle Trajectories and Supporting Data. [Dataset]. Provided by *ITS DataHub through Data.transportation.gov.* Accessed 2023-04-24 from http://doi.org/10.21949/1504477

[32] Zhang, T. T., & Jin, P. J. (2022). Spatial-temporal map vehicle trajectory detection using dynamic mode decomposition and Res-UNet+ neural networks. *arXiv preprint arXiv:2201.04755.*

[33] U.S. Department of Transportation Federal Highway Administration. (2016). Next Generation Simulation (NGSIM) Program I-80 Videos. [Dataset]. Provided by *ITS DataHub through Data.transportation.gov*. Accessed 2023-04-24 from http://doi.org/10.21949/1504477

[34] Ma, Y., Tang, K., Chen, S., Khattak, A. J., & Pan, Y. (2020). On-line aggressive driving identification based on in-vehicle kinematic parameters under naturalistic driving conditions. *Transportation research part C: emerging technologies, 114*, 554-571.

[35] Transportation Officials. (2011). *A Policy on Geometric Design of Highways and Streets, 2011.* AASHTO.

[36] Brown, T. L., Lee, J. D., & McGehee, D. V. (2001). Human performance models and rear-end collision avoidance algorithms. *Human Factors, 43*(3), 462-482.

[37] Widdows, D. (2004). *Geometry and meaning* (Vol. 773). Stanford: CSLI publications.

[38] Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., ... & Woods, E. (2020). Tslearn, a machine learning toolkit for time series data. *The Journal of Machine Learning Research, 21*(1), 4686-4691.

[39] Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019, July). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2623-2631).