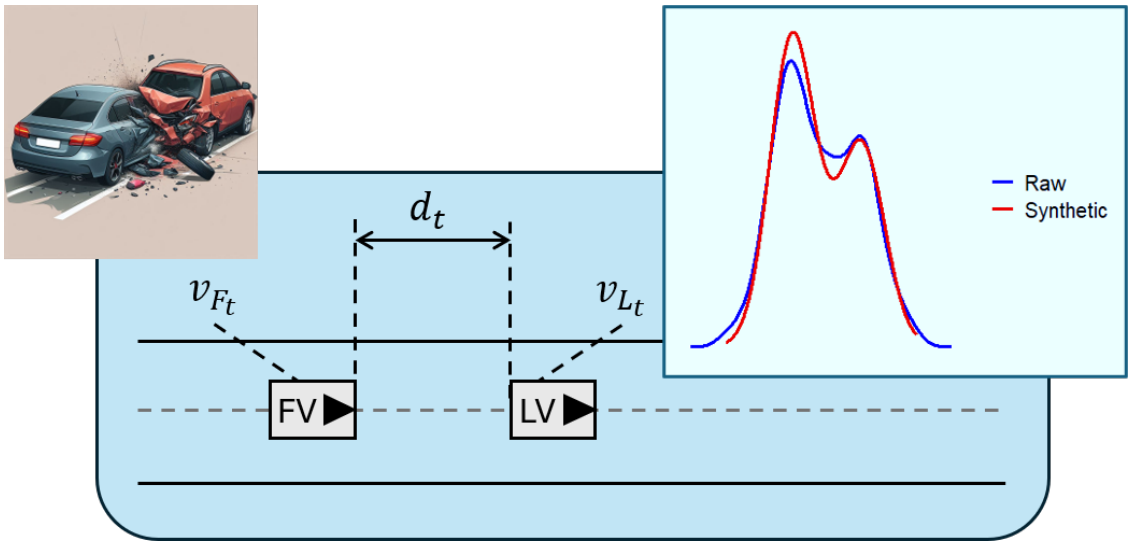


Using Behavioral Cloning for Rear-End Crash Scenario Generation

a feasibility and sensitivity study

Master's thesis in Mobility Engineering

MINXIANG ZHAO

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Using Behavioral Cloning for Rear-End Crash Scenario Generation

a feasibility and sensitivity study

Minxiang Zhao

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Vehicle Safety
Unit of Crash Analysis and Prevention
Chalmers University of Technology
Gothenburg, Sweden 2024

Using Behavioral Cloning for Rear-End Crash Scenario Generation
a feasibility and sensitivity study
MINXIANG ZHAO

Supervisor: Jian Wu, Department of Mechanics and Maritime Sciences
Examiner: Jonas Bärgman, Department of Mechanics and Maritime Sciences

Master's Thesis 2024
Department of Mechanics and Maritime Sciences
Division of Vehicle Safety
Unit of Crash Analysis and Prevention
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Rear-end crash scenario with both cartoon illustration (top left, generated by Bing Image Creator) and simplified illustration (centre), and probability distribution of both raw and synthetic kinematics (top right).

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Using Behavioral Cloning for Rear-End Crash Scenario Generation
a feasibility and sensitivity study
MINXIANG ZHAO
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

Autonomous Driving (AD) technology has made significant breakthroughs in recent years. Virtual safety assessment is a primary method for evaluating the safety of the AD functions due to its efficiency, high experimental control, low risk, and cost-effectiveness. However, the number of cases in existing crash datasets is insufficient to provide an adequate evaluation of critical scenarios. Therefore, it is necessary to generate synthetic crashes with high fidelity. Rear-end crashes, which has relatively low complexity (only involves the longitudinal movement of two vehicles) and high frequency (the most common crash type), is suitable as the first type for crash generation. Behavior Cloning (BC), which can reproduce vehicle behavior using a simple neural network structure, has the potential to generate synthetic crash scenarios with high fidelity. This thesis proposes a crash generation framework that combines a BC-based crash generation model with Hierarchical Clustering-based post-processing to generate crashes with the same distribution as the given crash dataset. In addition, as there is currently no established methodology for the assessment of generated cases, evaluation methods that combine both general statistical metrics and domain knowledge are introduced and implemented. In general, the proposed framework is capable of generating crash kinematics with high fidelity, as the generated cases could pass both visual inspection and statistical evaluation. Future work should focus on enhancing the robustness of the generation framework. The generated synthetic crashes could be employed either for virtual safety assessments of AD functions or rear-end crash kinematic analysis for research purposes.

Keywords: Rear-end Crash, Scenario Generation, Virtual Safety Assessment, Behavior Cloning, Data Synthesis, Deep Learning.

Acknowledgements

First, I would like to thank my examiner, Associate Prof. Jonas Bärgman, for giving me the opportunity to conduct research for my master's thesis. I appreciate him for taking his precious time to ensure the regular weekly meetings and for his patient guidance. I also thank him for his valuable suggestions to ensure that the thesis is headed in the right direction. I've learned a lot from this study, and I believe it's a precious treasure that motivates me to move forward.
I would like to thank my supervisors, Jian Wu and Torrin Raoufi-Danner from Volvo Car, for their participation in this study and their pertinent advice. When I faced some difficulties and confusion, Jian was always there to discuss with me to solve the problems and encourage me. I've also learned much from Jian about polishing presentation slides and presenting. As the machine learning expert in this team, Torrin always gives us valuable suggestions in the machine learning domain, which is very important as my background is in mechanical engineering. I also appreciate Torrin's guidance on academic writing and pertinent comments on my thesis.
Thanks to all my friends who advised me on my thesis and shared the daily joys with me. Thanks to all my opponents and audiences for attending my thesis defense. Lastly, please allow me to dedicate my sincerest wishes to my parents. Thank you for supporting me to study abroad, for taking care of me all the time, and for your efforts in raising me. I love you all!

Minxiang Zhao, Gothenburg, 11th June 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AD	Autonomous Driving
BC	Behavior Cloning
CISS	Crash Investigation Sampling System
DGM	Deep Generative Model
FV	Following Vehicle in a rear-end crash
GAN	Generative Adversarial Net
LV	Lead Vehicle in a rear-end crash
NDD	Naturalistic Driving Dataset
NDE	Naturalistic Driving Environment
NDS	Naturalistic Driving Study
SAE	Society of Automotive Engineers
SHRP2	Second Strategic Highway Research Program

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

c	Index for cluster
t	Index for time
$t_1 : t_2$	Indexes from time t_1 to time t_2 (with time interval Δt)

Parameters

T	Duration of the crash kinematics (in this study $T = 5s$)
Δt	Sampling time interval (in this study $\Delta t = 0.05s$)

Variables

a_F	Acceleration of FV (m/s^2)
a_L	Acceleration of LV (m/s^2)
d	Distance between LV and FV (m)
e	Coefficient of restitution (m/s)
I_F	Initial condition of FV (including acceleration and velocity at collision)
I_L	Initial condition of LV (including acceleration and velocity at collision)
v_F	Velocity of FV (m/s)
v_L	Velocity of LV (m/s)
Δv	Velocity difference between LV and FV (m/s)

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

In recent years, Autonomous Driving (AD) technology has advanced rapidly as a key area of transport research, with the potential to make autonomous driving part of our everyday lives. [1] However, the inadequate validation of the safety of existing autonomous driving technologies constitutes a critical impediment to the deployment of advanced autonomous vehicles (over SAE Level 3 [2], refers to Conditional Automation, where a vehicle can handle all driving tasks under certain conditions, but still requires a human driver to take over when requested). Research indicates that autonomous vehicles would have to be driven potentially hundreds of billions of miles in naturalistic driving environment (NDE) to establish their reliability regarding fatalities and injuries [3]. The extensive test mileage required for AD safety assessment will result in a significant investment of time and resources, which is infeasible for commercial development and deployment. In contrast to the test in real traffic, virtual safety assessment is a primary method for evaluating the safety of AD functions due to its efficiency, high experimental control, low risk, and cost-effectiveness [6, 7]. However, critical scenarios are notably scarce in the real-world traffic environment [3]. Consequently, rather than modeling the NDE, which also requires significant time and computational resources, creating specific safety-critical scenarios for safety assessment is also an important component for verifying the safety of AD functions.

The kinematic information of traffic objects represents a pivotal element in the description of a driving scenario. Three principal methods exist for generating collision kinematics information: Direct Sampling (from real cases), Case Modification, and Full Synthesis.[8]. Sampling real events directly from real data, such as in-depth crash datasets or NDD, is the most intuitive method to obtain rear-end crash samples. As Direct Sampling does not modify the event, the fidelity of the events is the same as the real data. However, it's also important to know that the recons The quality of the samples is determined by the real data. Yet, the number of events in the current datasets is typically insufficient. The current datasets could be expanded by modifying the original case to introduce additional variations. However, the resulting variations may not accurately reflect the true distribution, and the diversity of the expanded dataset may be constrained by the limitations of the original cases. Events can also be fully synthesized by using a training-based event generation model. The model could reproduce the distribution of the training dataset, and then generate virtual events which match the distribution of the training dataset.

The generation model could be either a model that generates the entire kinematics of the event directly or a combination of vehicle behavior (or kinematics) models. One model family that can generate the entire event is Deep Generative Models (DGMs) [15]. Generative Adversarial Nets (GANs) and Variational Auto-encoders (VAEs) are some typical algorithms for DGMs. DGMs show great potential mainly in image generation, but also in the traffic domain DGMs have successful applications [16, 17, 18]. DGMs have limited interpretability, and since DGMs generate an entire event rather than individual behavior at a specific timestamp, DGMs are less flexible than vehicle behavior models. Furthermore, DGMs are more complex and have a much greater number of parameters than vehicle behavior models. Consequently, DGMs require a significantly greater quantity of computing power and data to train. In this thesis, a vehicle behavior model aims to predict the behavior of the vehicle based on historical observations and its own actions. The entire event could be synthetically generated by gradually generating vehicle behavior over time, similar to how a driver would drive (e.g., [4, 5]). Imitation learning is a type of data-driven behavior model that reproduces behavior learned from given training data. Imitation learning is a flexible and straightforward approach to reproducing vehicle behavior, but it lacks interpretability and may produce unreasonable behaviors if the training dataset does not cover the input. Within the author's knowledge, there are few examples of the application of behavioral models, in particular imitation learning models, to generate the entire crash event. Behavior Cloning (BC) [22], a basic imitation learning method, with an uncomplicated structure, low computational cost, and high flexibility in crash event synthetics, will be used in this thesis.

1.2 Aim and Objectives

This thesis aims to enable the generation of synthetic rear-end crashes using BC architectures trained with a reference rear-end crash dataset provided by previous research ([9]). To achieve the aim, the project had the following objectives:

1. Develop the BC model by designing an appropriate neural network architecture and selecting the suitable inputs and output parameters to generate actions that reflect the distributions of behavioral patterns in the training dataset.
2. Design appropriate evaluation methods to comprehensively assess whether the generated crash events come from the same distribution as the training dataset.

The research can be summarised into the following research questions:

1. *Can a Behavioral Cloning model trained on a set of crash kinematics be used to generate crashes that are representative of the training set?*
2. *How do different amounts of training data samples affect the generation model performance?*
3. *How sensitive is the model to the underlying training data samples?*

The thesis is organized as follows. Chapter 2 will introduce the relevant theory and algorithms that are used in this thesis. Chapter 3 will introduce the methodology that was used to train the model, generate the synthetic crashes, and evaluate the generation. Chapter 4 will show the results of the generation and the evaluation in multiple dimensions. Chapter 5 will answer the research questions and further

discuss how the model was built, the evaluation results, the limitations of the model, and the outlook.

2
Theory

In the following section, the theory of the relevant concepts and algorithms used in the thesis will be introduced.

2.1 Rear-end Crash

2.1.1 Introduction

A rear-end collision is a type of collision in which a Following Vehicle (FV) collides with the rear of a Lead Vehicle (LV) [10]. According to the crash data collected by NHTSA in 2021 [10], rear-end crashes accounted for 29.0% of the total number of First Harmful Event (The first injury or damage-producing event that characterizes the crash type [11]). A variety of factors, such as driver distraction, bad weather, and brake failure, lead to rear-end crashes. Instead of thoroughly analyzing the specific reason that caused the crash to happen, this thesis is built on a crash generation model based on capturing the statistical distribution of the kinematic information in the rear-end crash.
Some variables worth analyzing in a rear-end collision will be introduced below.

2.1.2 Time to Collision

Time to Collision (TTC) [12] is a key metric for critical safety assessment, which could be calculated by Formula 2.1. TTC estimates the time remaining before the rear-end crash occurs, assuming that the current velocities of both FV and LV remain constant.

$$\mathrm{TTC} = \frac{d}{v_F - v_L} \tag{2.1}$$

2.1.3 Delta-V

Delta-V is one of the most commonly used variables to describe the severity of a two-vehicle collision. According to the theory of both elastic deformation and plastic deformation, the rebound between both FV and LV is affected by the FV velocity. As a result, the value of Delta-V is different from the velocity differential between LV and FV when the collision happened (Δv_0, calculated by 2.2).

$$\Delta v_0 = v_{F_0} - v_{L_0} \tag{2.2}$$

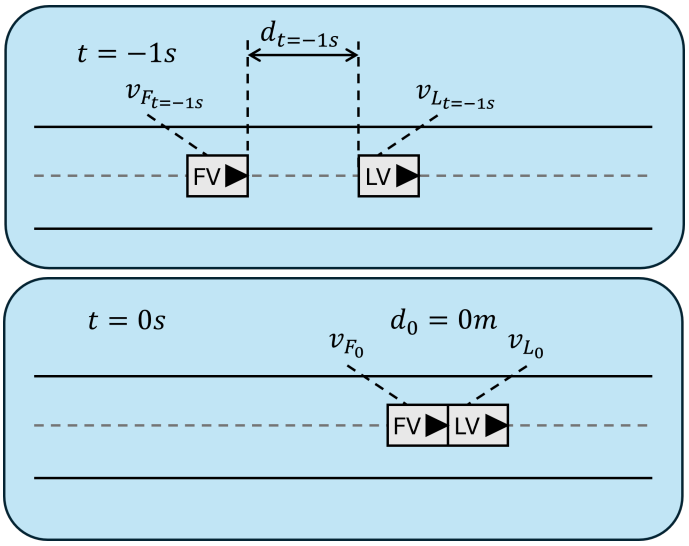


Figure 2.1: An illustration of the scenario of Rear-end Crash, where v_{F_t}, v_{F_t}, d_t represent the velocity of Following Vehicle (FV), the velocity of Lead Vehicle (FV), and the distance between LV and FV at time t, respectively. The upper figure illustrates the scenario that $1s$ before the collision happened. The lower figure illustrates the scenario that the collision happened

Further, a Coefficient of Restitution e was introduced to represent the ratio of the post-impact vehicle-velocity-difference to the pre-impact vehicle-velocity-difference between the two vehicles. The value of e ranges from 0 to 1. When $e = 0$, both vehicles "lock together" post-impact with no rebound. In [14], a best-fit value for e can be obtained by the Equation 2.3.

$$e = 0.47477 - 0.26139 \log_{10} v_{F_0} + 0.03382 (\log_{10} v_{F_0})^2 - 0.1139 (\log_{10} v_{F_0})^3 \quad (2.3)$$

Then, the more accurate Delta-V, effected by e, could be determined by Equation 2.4.

$$\text{Delta-V} = \Delta v_0 (1 + e) \quad (2.4)$$

2.1.4 Point-of-no-retrun

Figure 2.2 categorizes different time periods during the whole crash events. Depending on the level of severity, pre-crash could be categorized into three periods. The first period (the no-conflict zone) represents the period when there is no risk of a crash. The second period (the conflict zone) represents the period in which a crash will occur in a few seconds (or even fractions of a second) if no intervention strategy is adopted. The last period is the period after the point-of-no-return (PONR), defined as the point in time when a crash is no longer avoidable.
In order to develop an effective active safety function, the beginning time of each period in pre-crash is crucial to be determined. As a result, the generated crashes should ideally keep the same distribution of each time period for pre-crash as the real-world crashes to ensure the fidelity of the generation. In this thesis, the no-return time, i.e., the time at the PONR, is one of the metrics that will be determined to evaluate the quality of the generation.

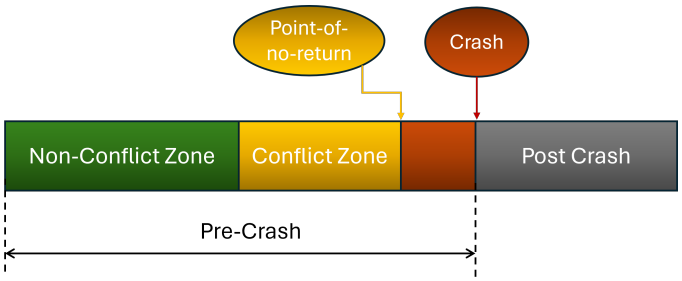


Figure 2.2: Schematic of periods during pre-crash and post-crash.

2.2 Clustering

Clustering, a method of unsupervised learning in machine learning and data analysis, entails grouping a set of objects or data points into clusters [41]. The formation of clusters is such that objects within the same cluster exhibit greater similarity to each other than those in different clusters. The primary objective of clustering is to uncover natural groupings within the data. A variety of clustering algorithms, including K-means [42], DBSCAN [43], and Hierarchical Clustering [44], are employed in diverse domains for the analysis of data and the recognition of patterns. The selection of clustering algorithms should be guided by an understanding of the intrinsic characteristics and distributional properties of the target dataset [45].

2.2.1 Hierarchical Clustering

Hierarchical clustering is a method of cluster analysis that aims to build a hierarchy of clusters. There are two main types of Hierarchical Clustering: Agglomerative Hierarchical Clustering and Divisive Hierarchical Clustering. In this study, Agglomerative Hierarchical Clustering is employed. The step in the Agglomerative Hierarchical could be concluded as below:

1. Start with each datapoint as its own cluster.
2. Find the pair of clusters that are closest together, based on a distance metric (e.g. Euclidean Distance).
3. Merge the two closest clusters.
4. Repeat the above steps until all the data points are in a single cluster or until the desired number of clusters is achieved.

The linkage criteria, which determine how the distances between clusters are calculated, need to be specified before cluster implementation. The linkage method used in this study is Single Linkage, which calculates the minimum distance between any single element in different clusters. The result of hierarchical clustering is typically presented in a tree-like diagram, known as a dendrogram. This illustrates the arrangement of the clusters produced by the corresponding analyses. The height at which two clusters are joined together indicates the distance between them. By cutting the dendrogram at different heights, different numbers of clusters can be formed.

2.2.2 K-Means

K-means [42] is a popular centroid-based clustering algorithm. The goal of K-means is to divide a dataset into K clusters, with each data point assigned to the cluster with the nearest centroid, which is the mean of the cluster. The algorithm aims to minimize the within-cluster sum of squares, which is a measure of the distance between each data point and its assigned centroid.
K-means is efficient and easy to implement. However, K-means is sensitive to initial centroid selection and assumes the shape of clusters to be spherical with equal variance, which introduces limitations when implementing [45].

2.3 The Crash Generation Algorithm

2.3.1 Training-validation-test Dataset

The training dataset, validation dataset, and test dataset represent the fundamental components of machine learning, serving to fit, tune, and evaluate the machine learning model. [38]

1. **Training dataset:** The training dataset represents a subset of the original dataset employed to train the machine learning model. During the training process, the model learns patterns, relationships, and other pertinent information from this data. The training process involves adjusting the model's parameters to minimize errors and improve accuracy [39].
2. **Validation dataset:** The validation dataset is employed during the training process to evaluate the model's performance and to tune the training hyperparameters (e.g., learning rate, number of layers in a neural network). The validation dataset helps to prevent overfitting, where the model performs well on the training data but poorly on unseen data. The validation dataset acts as an unseen set of data that the model does not directly learn from, providing an unbiased evaluation of the model's performance during training [40].
3. **Test dataset:** The test dataset represents a subset of the data utilized to assess the final model's performance after the model has been trained and validated. This dataset is only used once the model has been fully trained and all hyperparameter tuning is complete. The test dataset provides an unbiased evaluation of the model's ability to generalize to new, unseen data, and thus offers an estimation of how the model will perform in the real world.

2.3.2 Behavior Cloning

Behavior Cloning (BC) refers to a model that learns the policy of the driver behavior from the given driver behavior dataset. In this context, a policy refers to a function that dictates the behaviors a driver should adopt based on their observations. In 1989, Pomerleau [22] trained a neural network, shown in Figure 2.3, using video signals and rangefinder signals as input and output steering signals, thereby providing the earliest example of the use of BC. In Figure 2.3, the Direction Output Units represent the probability distribution of the turn curvature along which the

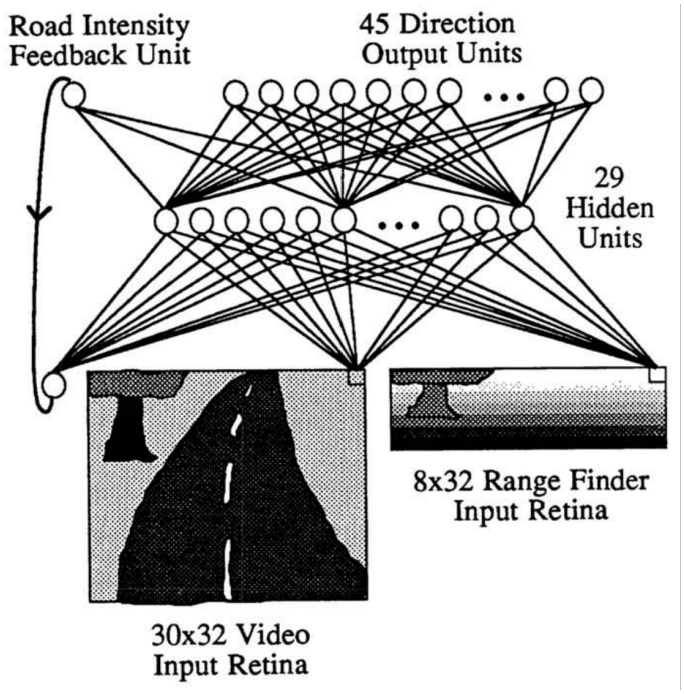


Figure 2.3: ALVINN Architecture. Reprinted from "ALVINN: an autonomous land vehicle in a neural network" by Dean A. Pomerleau., 1988, Proceedings of the 1st International Conference on Neural Information Processing Systems (NIPS'88). MIT Press, Cambridge, MA, USA, 305–313. Reprinted with permission.

vehicle should travel. By deriving the expectation value of the distribution, the turn curvature could be predicted. This form of probability distribution prediction serves as the foundation for the modeling approach presented in this thesis. However, the model in this thesis differs from Pomerleau's approaches by setting the distribution as the output rather than the expectation value of the distribution, which allows for the generation of diverse outputs for a single input. Waymo [24] proposed a car-following BC model, whose inputs also inspired the model input in this thesis. In [24], the inputs of the car-following BC include the historical time-series of the FV's velocity v_F, the velocity difference between FV and LV Δv, the distance between FV and LV d, and the FV's action (i.e., acceleration) a_F. The car-following BC model outperformed the rule-based Intelligent Driver Model (IDM) [25] (a commonly used car-following model in traffic research) on all metrics, indicating that BC has a high potential to reproduce human driving behavior.

2.3.3 Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP), a fundamental building block in the field of neural networks and deep learning, is a class of feedforward artificial neural networks [28]. In general, an MLP consists of at least three layers of nodes: an input layer, one or more hidden layers, and an output layer. Each node (except input nodes) is a neuron that uses a nonlinear activation function, which will be introduced in Section 2.3.4.

MLPs are versatile and can be applied to a wide range of tasks, including classification, regression, pattern recognition, time series prediction, etc [29]. However, the efficacy of MLPs is affected by the selection of architectural parameters, including the number of layers and the number of neurons per layer. Additionally, the choice of hyperparameters, such as the learning rate and batch size, can significantly influence the performance of MLPs [27]. As a result, understanding the structure and training process of MLPs is essential for successful training of MLPs [26].

2.3.4 Activation Function

The purpose of activation functions is to introduce non-linearity into the model, thereby enabling the neural network to learn complex patterns and perform more sophisticated tasks. The activation functions used in this study are introduced in this Section.

2.3.4.1 Rectified Linear Unit

Rectified Linear Unit (ReLU) [33] is one of the most popular activation functions used in neural networks. The output of ReLU is the input itself if the input is positive and zero otherwise. Mathematically, ReLU can be expressed as Formula 2.5.

$$\mathrm{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{2.5}$$

2.3.5 Softmax and LogSoftmax

Both Softmax and LogSoftmax [34] are activation functions employed in neural networks, particularly in the context of multi-class classification problems. The output values of Softmax are in the range $(0, 1)$ and sum to 1, which can be interpreted as a probability distribution. As a result, Softmax is typically used in the final layer of a network designed for classification tasks. Mathematically, Softmax can be expressed as Formula 2.6, where x is the input vector.

$$\mathrm{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{2.6}$$

LogSoftmax is defined as the logarithm of the Softmax, which can be expressed as Formula 2.7.

$$\mathrm{LogSoftmax}(x_i) = \log\left(\frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}\right) = x_i - \log\left(\sum_{j=1}^{n} e^{x_j}\right) \tag{2.7}$$

By combining the softmax and logarithm in one step, LogSoftmax helps to avoid numerical issues that arise from taking the logarithm of very small probabilities. As a result, LogSoftmax exhibits greater numerical stability than Softmax.

2.3.6 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) [30] is a variant of the Recurrent Neural Network (RNN), which is commonly used in the field of deep learning for processing sequential data. The GRU was introduced to address some of the limitations associated with traditional RNNs, particularly the issues of vanishing and exploding gradients that make training difficult over long sequences. In contrast to Long Short-Term Memory (LSTM) networks [32], which comprise three gates (input, forget, and output gates) and a separate cell state, the GRU integrates these functionalities into a more streamlined structure with only two gates, a Reset Gate and an Update Gate. The Reset Gate controls how much of the previous information to forget, and the Update Gate determines how much of the past information needs to be passed along to the future. GRUs often perform similarly to LSTMs on various tasks. With fewer parameters, GRUs are more computationally efficient and require less training time [31]. GRUs are widely used in various applications involving sequential data, especially in time-series prediction [31]. In the field of driver behavior modeling, Waymo [24] utilized the GRU to build a car following model, as previously mentioned, which demonstrated relatively good performance.

2.3.7 Kullback-Leibler Divergence

Kullback-Leibler Divergence (KL-Divergence) [35] is a measure from information theory that quantifies the difference between two probability distributions. Given two probability distributions P and Q defined on the same probability space, the KL-Divergence from Q to P is defined as equation 2.8 for discrete distributions.

$$D_{KL}\left(P\|Q\right) = \Sigma_{x\in X} P\left(x\right) \log \frac{P\left(x\right)}{Q\left(x\right)} \tag{2.8}$$

where $P\left(x\right)$ and $Q\left(x\right)$ are the probability density functions of P and Q, respectively. KL-Divergence is a reliable method for measuring the discrepancy between the distribution estimated by statistical models and the distribution of real-world data, and has broad applications in various scientific and engineering disciplines [36]. In this study, the KL-Divergence is employed as the loss function in the BC training process.

2.4 Evaluation Algorithms

In this section, the theory of the algorithms for evaluating the generation quality will be introduced. The specific methodology for utilizing these algorithms will be explained in Section 3.3.

2.4.1 T-Distributed Stochastic Neighbor Embedding

T-Distributed Stochastic Neighbor Embedding (T-SNE) [13] is a machine learning algorithm developed for dimensionality reduction and is commonly employed in data

analysis to explore and interpret high-dimensional data. T-SNE is particularly well-suited for exploring complex datasets and uncovering patterns that may not be apparent in the original space, which is the high-dimensional space where the data points are originally located. T-SNE projection is a scatter plot in which each point represents a data instance that has been condensed to two or three dimensions, with inter-point distances reflecting the degree of similarity between instances. In summary, T-SNE projection is a visual aid for conveying complex data relationships and structures in a more digestible format.

2.4.2 Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov (KS) Test, which in this thesis is specifically referred to as the Two-sample KS Test, is a non-parametric statistical method that compares the distributions of two datasets. The p-value is one of the fundamental outputs of the KS Test. A p-value below the pre-defined threshold (typically 0.05) indicates that the null hypothesis can be rejected, suggesting a significant difference between the distributions. In order to calculate the p-value, one first calculates the empirical distribution functions of two datasets. The test statistic is then computed, which is the maximum absolute difference between the two samples' empirical distribution functions, to determine the p-value. The non-parametric nature of the KS Test renders it highly versatile and applicable across a wide range of data types and fields of study.

3
Methodology

This chapter will introduce the dataset used, the behavior generation model, and the evaluation. The section Dataset will describe the dataset used in this work and how we transform the dataset for training, validation, and testing. The section Synthetic Dataset Generation Framework will introduce our generation framework, including the model structure for both LV and FV, and the cluster post-processing methodology. The evaluation part will introduce the data evaluation methods used to compare the generated data with the raw data.

3.1 Dataset

The dataset used to train, validate, and test the model is the synthetic rear-end crash dataset from Wu [9]. The dataset includes 5000 rear-end crashes with different sample weights, that describe the time-series kinematic information of both the LV and the FV longitudinally (along the road) in a duration $T = 5s$ before the collision happened. The time-series kinematic information used to describe the crashes includes LV's velocity ($v_{L_{T:0}}$), FV's velocity ($v_{F_{T:0}}$), and the distance between LV and FV ($d_{T:0}$). Comparing the synthetic dataset and the original dataset (rear-end crash from the CISS and SHRP2 NDS) of Wu's work [9], the synthetic dataset is substantially larger. Training could start with a larger dataset to validate the feasibility of the model, as a large dataset is more diverse and comprehensive, helping to train a more effective model.
The dataset should be pre-processed to split the training-validation-test dataset and to derive the input and output for model training. Firstly, the training-validation-test dataset should be uniformly sampled from the raw dataset thus representing a distribution consistent with the pattern of the dataset. According to the process in Figure 3.1, by uniformly sampling from each cluster, the consistency of pattern distribution could be controlled. What's more, the inputs and outputs of the model need to be derived as they cannot be obtained directly from the original dataset.

3.1.1 Clustering

The raw dataset does not include any specific categorization of the 5000 raw cases. Consequently, the behavioral patterns and the pattern distribution of cases in the raw dataset remain unknown. This is where clustering comes in. Clustering is an unsupervised learning technique that groups data points into subsets based on similarities. By clustering the kinematics, (including the time-series of normalized

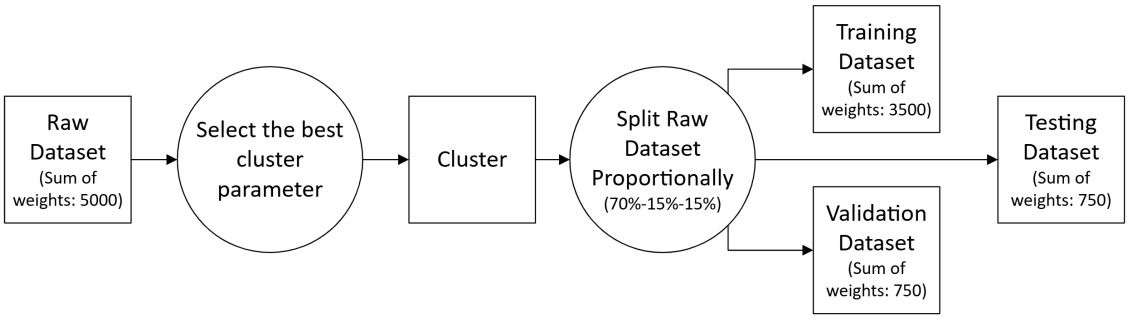


Figure 3.1: Process of dataset splitting based on clustering. The cases for which the sum of weights constituted 70% of the raw dataset were sampled as the training datasets. The remaining cases in the raw dataset were distributed equally to the validation and test datasets, based on the sum of weights.

LV velocity, FV velocity, and the distance between LV and FV) of the cases in the training dataset, the behavior patterns and the pattern distribution in the training dataset can be derived.

3.1.1.1 Normalization

Before clustering, the time-series crash kinematics should be normalized. The normalization method used here is the Z-score, the formula for which is shown in the equation 3.1.

$$\boldsymbol{x}^* = \frac{\boldsymbol{x} - \mu_{\boldsymbol{x}}}{\sigma_{\boldsymbol{x}}} \tag{3.1}$$

where the $\mu_{\boldsymbol{x}}$ and $\sigma_{\boldsymbol{x}}$ are the mean and standard deviation of the time-series $\boldsymbol{x}$ respectively. Z-score normalization removes the mean and scales the variance, allowing the clustering algorithm to focus on the shape rather than the absolute values of the kinematic time-series. The three dimensions of kinematics, i.e., $v_{L_{T:0}}$, $v_{F_{T:0}}$, and $d_{T:0}$, need to be normalized separately to ensure that all time series contribute equally to the similarity measurement.

3.1.1.2 Inferring the clustering parameters

The clustering algorithm selected for the kinematics data of crashes in this thesis is Hierarchical Clustering. The selection process is discussed in Section 5.5.1.1. The single linkage cluster, which is a linkage method for Hierarchical Clustering, merges the clusters that have the smallest minimum pairwise dissimilarity. Thus, the single linkage cluster works well for data with elongated or irregularly shaped clusters, and the weight of cases does not influence the clustering result. By trying different linkage methods, the single linkage cluster shows the relatively best results. The dendrogram shown in the upper graph of Figure 3.2 illustrates the hierarchical relationships between the various levels of clusters. According to the dendrogram, six clusters, whose structure is shown in the lower graph of Figure 3.2, is a relatively optimal number of clusters. The introduction of clusters larger than six would result in the emergence of additional outlier clusters, comprising a relatively small number

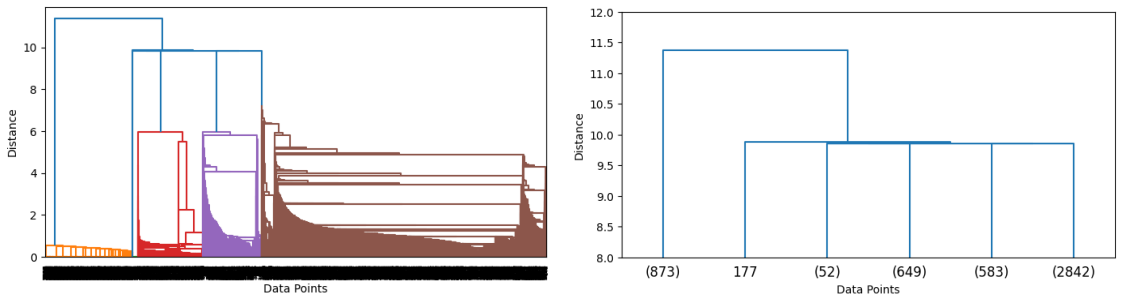


Figure 3.2: Dendrogram of the Hierarchical Clustering (Linkage method: Single). The upper graph shows all hierarchical levels (no truncation is performed). The lower graph illustrates the initial six clusters, with the largest cut-off distance threshold. The label with a number in brackets indicates the number of cases in a cluster, while the label without brackets denotes the index of the case in a cluster.

of cases within each cluster. The clustering results will be visually verified in the next section (Section 3.1.1.3).

3.1.1.3 Cluster inspection

Figure 3.3 and 3.4 illustrate the result of Hierarchical Clustering with a single linkage method when the cluster number is six.
Based on Figure 3.3 and 3.4, the pattern meaning, which is the kinematic properties of each cluster, can be described as follows:

- Cluster 1: The LV remains stationary while the FV accelerates from stationary. The distance between the FV and LV remains constant before gradually decreasing.
- Cluster 2: Both LV and FV keep a constant velocity. The distance between FV and LV decreases linearly.
- Cluster 3: The LV keeps a constant velocity, while the FV brakes when collision approaches. The distance remains linearly decreasing initially and the decrease rate becomes smaller as the collision approaches.
- Cluster 4: The LV velocity decreases, while the FV velocity remains constant. The distance between the FV and LV varies slightly initially and then decreases.
- Cluster 5: Both the LV and FV brake, with the FV braking later. The decreasing of distance is increasing as time approaches the collision.
- Cluster 6: (An outlier) The LV accelerates and then decelerates to stationary. The FV accelerates initially and then keeps a constant velocity. The distance decrease rate decreases first and then increases.

Although the single linkage cluster is less sensitive to outliers, the outliers would be produced as chains of clusters. The outlier (Cluster 6) exists when the maximum number of clusters is three (according to the dendrogram in the lower graph of Figure 3.2). As a result, Cluster 6 is hard to integrate into other clusters. Nevertheless, upon visual inspection, the result of Hierarchical Clustering is logical and has high interpretability.

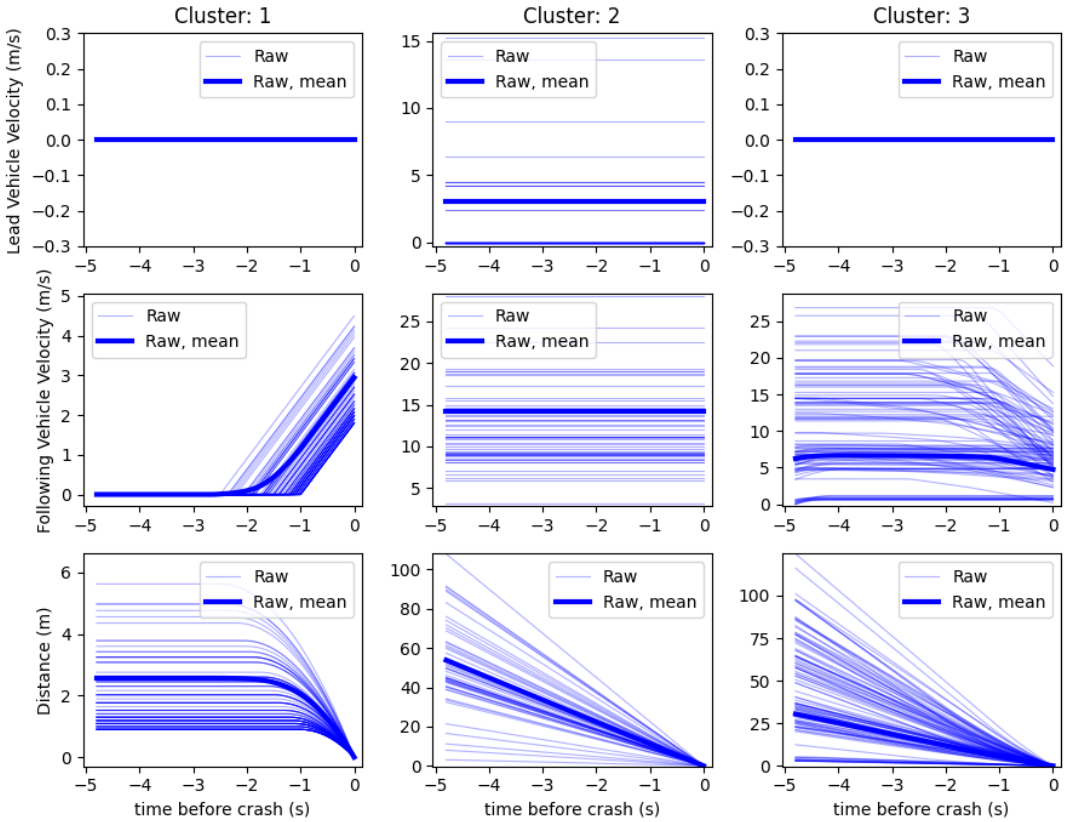


Figure 3.3: Visual inspection of the clustering result when the number of clusters is six (Cluster 1-3)

Table 3.1: The proportion of the sum of the weights in each cluster.

Cluster	1	2	3	4	5	6
Sum of weights	479.84	154.98	910.00	679.86	2765.99	9.32
Proportion	9.60%	3.10%	18.20%	13.60%	55.32%	0.19%

3.1.1.4 Sampling of the training, validation, test datasets

The distribution of patterns between the training and test datasets is shown in Table 3.1, which shows the sum of the weights of all samples in each cluster as a percentage of the total weight.

The training, validation, and test datasets in this work are sampled from each cluster uniformly according to the proportion 70%-15%-15%, that Figure 3.1 shows. Since only one case is in Cluster 6, the case in Cluster 6 could be assigned to training-validation-test datasets. The sample weight of the case in Cluster 6 should also be multiplied by a number equal to the split proportion of the dataset (e.g., multiply the weight of the case in Cluster 6 by 70% when assigning this case to the training dataset) to maintain the cluster proportion in all datasets. Although the case in Cluster 6 is an outlier compared to the other cases in the raw dataset, the case in Cluster 6 still represents one of the patterns, and thus the model has no reason to be trained without this case. What's more, the case in Cluster 6 should also be

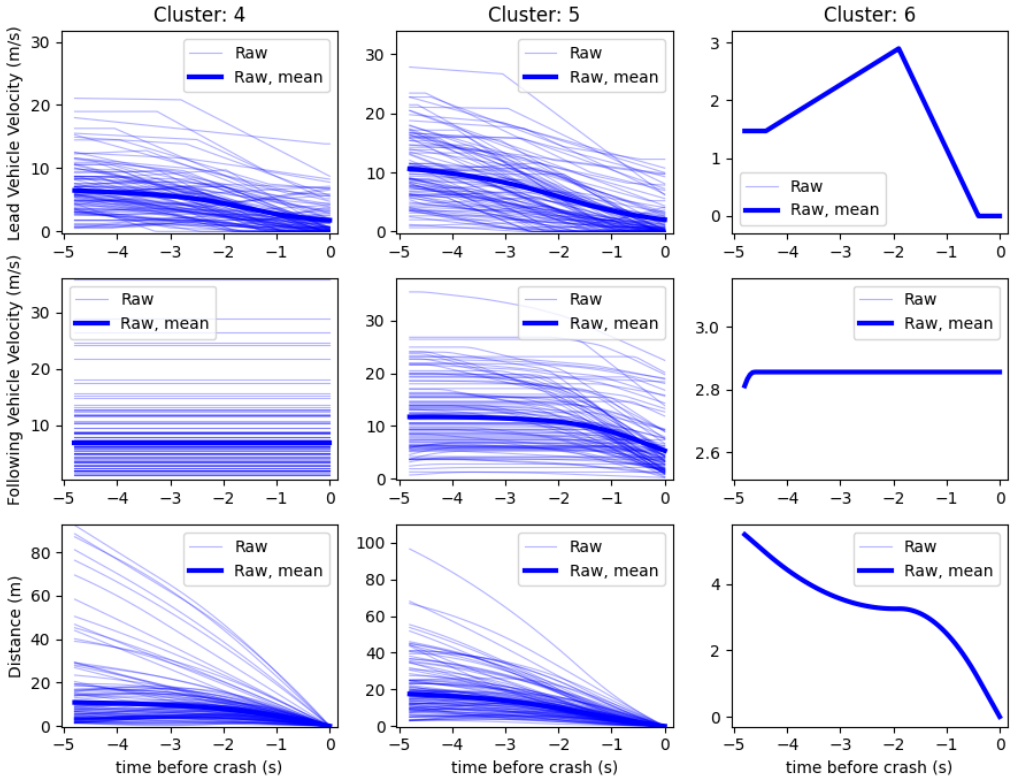


Figure 3.4: Visual inspection of the clustering result when the number of clusters is six (Cluster 4-6)

included in the validation and test dataset to verify that the generation model can reproduce the pattern of Cluster 6.

3.1.2 The behavior model inputs and output

The vehicle action, specifically the acceleration of the vehicle, is both one of the inputs and the final output of the behavior model. The vehicle action a_t is here derived from the velocity time-series by the equation 3.2.

$$a_t = \frac{v_{t+\Delta t} - v_t}{\Delta t} \tag{3.2}$$

The output of the behavior model is a distribution of the vehicle action $P(a_t)$. Since continuous distribution is not feasible to be built (i.e. the number of output units of the neural network cannot be infinite), a discrete distribution is instead created to represent the distribution of actions. The upper bound and the lower bound of the discrete distribution are derived from the maximum and minimum values of the vehicle acceleration in the raw dataset. To simplify the discrete distribution, the discrete interval was set to $0.25 m/s^2$. The parameters of both LV and FV behavior models are shown in Table 3.2. The action ground truth, which is a single numerical value, should be converted to the discrete distribution according to equation 3.3. Firstly, the probability density value in the discrete distribution is derived from the normal distribution of the action value. The mean value of the normal distribution

Table 3.2: Parameters of the discrete distribution for both LV and FV behavior models

	Upper bound	Lower bound	Interval	Length
LV model	$7m/s^2$	$-9.5m/s^2$	$0.25m/s^2$	67
FV model	$4m/s^2$	$-9.5m/s^2$	$0.25m/s^2$	55

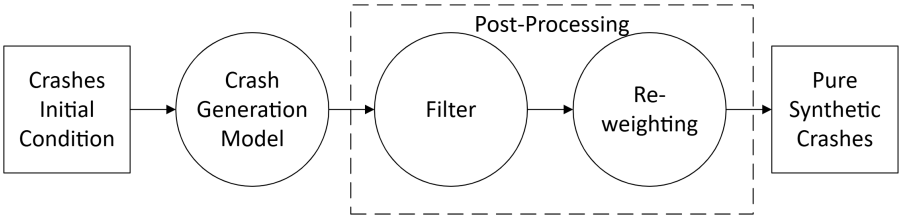


Figure 3.5: A schematic view of the synthetic dataset generation framework.

is the action value, and the standard deviation of the normal distribution is set to 0.07. Then, the discrete distribution should be normalized to sum up to 1.

$$P(a_t) = \text{Normalize}(\mathcal{N}(a_t, \sigma = 0.07)) \tag{3.3}$$

3.1.3 Derive the model input set

In the raw dataset, the values of four parameters v_{F_0}, v_{L_0}, a_{F_0}, a_{L_0} which describe the kinematics on the timing of collision, form a joint distribution. By modeling and sampling the joint distribution, a set of new values for the input variables of the BC model can be derived, which will then be used to generate a greater number of synthetic crashes.

$$v_{F_0}, v_{L_0}, a_{F_0}, a_{L_0} \sim P(v_{F_0}, v_{L_0}, a_{F_0}, a_{L_0}) \tag{3.4}$$

3.2 Synthetic Dataset Generation Framework

The structure of the framework to generate the synthetic crashes dataset is shown in Figure 3.5. Firstly, the Crash Generation Model, which consists of both the FV and LV models, generates the "raw" synthetic crash dataset. Then, through the Cluster Post-Processing, some low-quality cases are removed, and the remaining cases in the synthetic dataset are re-weighted to balance the kinematics pattern. The following sections will introduce each part in detail.

3.2.1 Behavior Generation Model

In the following sub-section, the structure of the model for both the Lead Vehicle and the Following Vehicle will be introduced. Both LV model and FV model are modified from Behavior Cloning. In contrast to most of BC, the behavior of both LV and FV will be generated backward, i.e., using the kinematics at the time of the

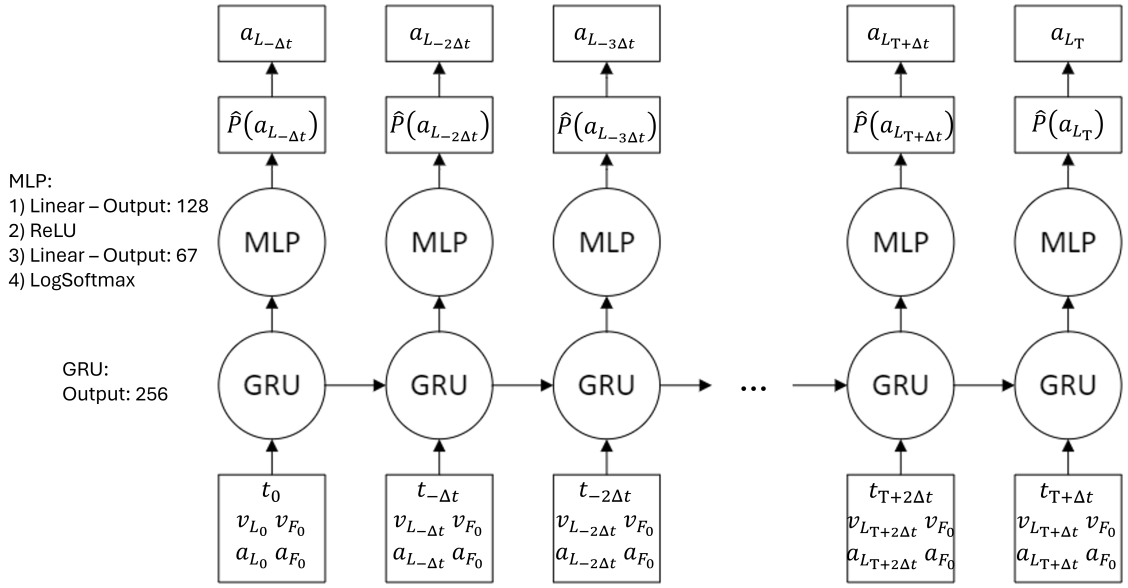


Figure 3.6: A schematic view of the LV model architecture.

collision to derive the trajectory during the 5 seconds before the collision happened. The reason will be discussed in Section 5.3.1.

3.2.1.1 Lead Vehicle

Figure 3.6 shows the architecture of the Lead Vehicle Behavior Generation Model, where $T = -5s$ represents the prediction duration, i.e. predict the kinematics $5s$ before the collision happened, and $\Delta t = 0.05s$ represents the prediction time interval, which is the same as the time interval in the raw dataset. The mathematical representation of the LV model is given by Equation 3.5. The first input to the GRU includes both LV's initial condition I_L, FV's initial condition I_F, and a time indicator $t_0 = 0s$. Both velocity and acceleration on $t = 0s$ of the vehicle consist of the vehicle's initial condition. Counterintuitively, I_F will influence the LV generation quality. The rationale of using I_F as the input of the LV model will be discussed in Section 5.3.2 with the analysis in both specific instances and statistics.

$$\hat{P}\left(a_{L_t}\right) = BC_{LV}\left(t_{t+\Delta t:0}, I_F, v_{L_{t+\Delta t:0}}, a_{L_{t+\Delta t:0}}\right) \tag{3.5}$$

The output of the GRU is used as input to the MLP. The ReLU activation function was introduced into the MLP in order to introduce non-linearity into the model. The output of the MLP is a probability distribution of the LV's behavior on timestamp t, i.e., the acceleration distribution ($\hat{P}\left(a_{L_t}\right)$). As a result, the size of the output is set equal to the length of the discrete distribution in Table 3.2, and a LogSoftmax activation function was introduced as the last layer of the MLP to ensure that the sum of the output nodes is equal to one. The reason for using LogSoftmax instead of Softmax is that LogSoftmax has higher numerical stability because LogSoftmax avoids direct exponentiation of large values.
By randomly sampling from $\hat{P}\left(a_{L_t}\right)$, an a_{L_t} can be computed. Through the motion model shown in Equation 3.6, the kinematics at timestamp t can be calculated, thus

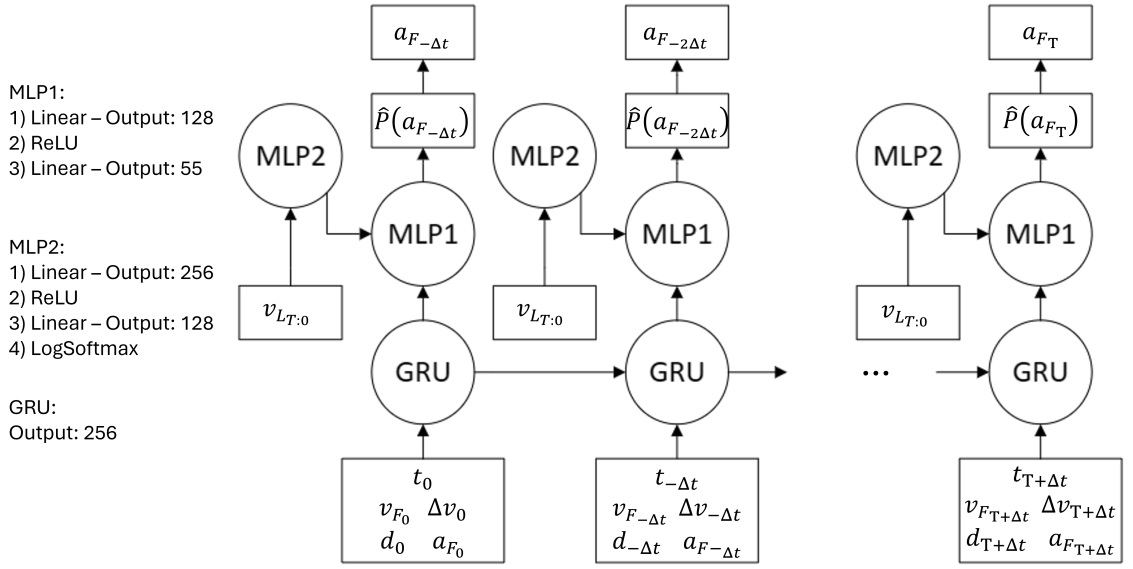


Figure 3.7: A schematic view of the FV model architecture.

the input of the next GRU could be derived. By repeating the above process, the LV kinematics during $5s$ could be generated.

$$\begin{bmatrix} x_{p_t} \\ x_{v_t} \\ x_{a_{t-\Delta t}} \end{bmatrix} = \begin{bmatrix} 1 & -\Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & -\Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{p_{t+\Delta t}} \\ x_{v_{t+\Delta t}} \\ x_{a_t} \end{bmatrix} \tag{3.6}$$

To train the model, the output generated over the entire $5s$, computed by taking the entire trajectory of LV from the training dataset as the model input, was compared with the ground truth derived in Section 3.1.2. The algorithm to calculate the loss between the prediction and ground truth is KL-Divergence, as the Equation 3.7 shows.

$$Loss\left(a_{T:0}\right) = D_{KL}\left(\hat{P}\left(a_{T:0}\right) \| P\left(a_{T:0}\right)\right) \tag{3.7}$$

3.2.1.2 Following Vehicle

Figure 3.7 shows the architecture of the Following Vehicle Behavior Generation Model, and the mathematical representation of the FV model is given by Equation 3.8. According to the car following model proposed by Waymo [24], which yields a relatively good performance, the velocity of FV v_F, the velocity difference Δv, and the distance d between both vehicles could influence the kinematics of FV. Thus, the Δv, d, the FV's historical kinematics v_F and a_F, and the timestamp indicator t are set as the inputs of the GRU. The behavior prediction quality could be improved by introducing the whole LV trajectory $v_{L_{T:0}}$ as one of the inputs. Since $v_{L_{T:0}}$ contains about 100 elements (while the current number of input elements for one GRU is 5), inputting $v_{L_{T:0}}$ to the GRU directly will increase the training time. Thus, in contrast to the LV model, the FV model has one more MLP network (MLP2 in Figure 3.7) to extract the features of the entire LV trajectory.

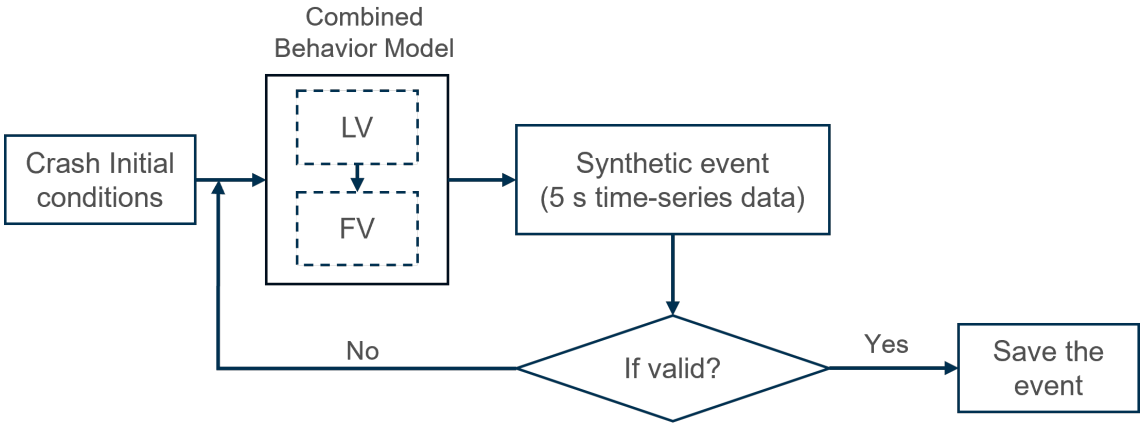


Figure 3.8: Structure of the crash generation model

$$\hat{P}\left(a_{F_t}\right) = BC_{FV}\left(t_{t+\Delta t:0}, v_{F_{t+\Delta t:0}}, \Delta v_{t+\Delta t:0}, d_{t+\Delta t:0}, a_{F_{t+\Delta t:0}}, v_{L_{T:0}}\right) \tag{3.8}$$

The training process and generation process for the FV model are the same as for the LV model.

3.2.2 Crash Generation Model

The entire structure of the crash generation model is illustrated in Figure 3.8. Firstly, the trajectory of LV is generated based on the input derived from both I_L and I_F. Then, the trajectory of FV can be generated, and the entire kinematics can be derived. However, the synthetic case may be unrealistic (i.e., a secondary collision happened before the collision at $t = 0$). By setting a validation check for the generated synthetic event, unrealistic cases could be detected, and the case will be re-generated using the same input as before.

3.2.3 Cluster Post-Processing

An ideal behavioral generation model should generate cases that accurately reflect the distribution of the orignal behavior of both the LV and the FV. However, due to the limited performance of the model, the generated cases may exhibit distorted shapes or a biased distribution of behavioral patterns. By removing cases that are too distant from the cluster, which are assumed to have distorted shapes, and re-weighting the cases to ensure that each kinematic pattern has the same proportion of the sum of weights as the training dataset, the generation quality is expected to improve.

Synthetic cases whose inputs are identical to the initial condition of the training dataset could be used to derive the parameter of both Cluster Filter and Cluster Re-weighting. The inputs identical to the initial condition of the training dataset could be used to generate synthetic cases to verify the generation quality. Here, synthetic cases with the sum of weight $15 \times 3500 = 52500$ (i.e., every initial condition is generated 15 times) are generated.

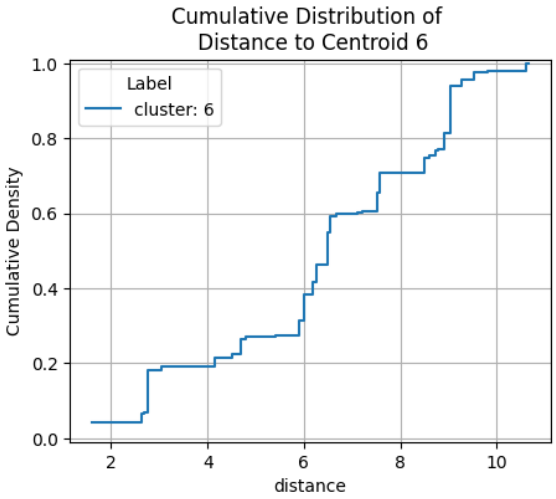


Figure 3.9: Cumulative density graph of the distance of the cases in Cluster 6 to the single raw case in Cluster 6

3.2.3.1 Cluster Filter

One of the key characteristics of unrealistic cases is that they are far from clusters. The distance threshold is used to filter the cases that are far from clusters. The distance threshold of each cluster could be derived from the minimum distance between cases in each cluster. Firstly, the minimum distance of each case in the cluster from the cases in the cluster (other than cases coinciding with itself) is calculated. Then, the weighted mean μ_c and weighted standard deviation $3\sigma_c$ of the minimum distance of each cluster could be computed. According to the Three Sigma Rule, which indicates that the range in Three Sigma could cover the majority of datapoints (99.7% for Gaussian distribution), the distance threshold d_c for each cluster could be set as Equation 3.9.

$$d_c = \mu_c + 3\sigma_c \tag{3.9}$$

However, according to Equation 3.9, the distance threshold of Cluster 6 is 0 since only one case is in Cluster 6. The new proper distance threshold of Cluster 6 can be derived from the elbow point of the cumulative density graph (Figure 3.9), in which the x-axis is the distance between Cluster 6 and the synthetic cases closest to Cluster 6. However, Figure 3.9 does not have an explicit elbow point. Instead, the weighted mean value of the distance in Figure 3.9, which is equal to 6.4, could be set as the distance threshold in the beginning. The distance threshold could be further adjusted based on visual inspection.
The distance thresholds of each cluster are shown in Table 3.3.
Then, the synthetic cases can be assigned to each cluster to visualize the generation after the filter. Although Hierarchical Clustering can not give labels to new data points directly, some algorithms (e.g., K-Nearest Neighbors (KNN)) based on the distance between cases and clusters can assign labels to the synthetic cases. In this thesis, the labels of synthetic cases are assigned based on the minimum distance between cases and clusters and the distance threshold of each cluster. Cases will

Table 3.3: Cut-off threshold for each cluster

Cluster	1	2	3	4	5	6
Weighted Mean μ_c	0.360	0	0.170	1.23	1.18	-
Weighted Standard Deviation σ_c	0.0896	0	0.443	1.04	0.917	-
Distance Threshold d_c	0.628	0	1.50	4.35	3.93	6.4

be priority allocated to the closest cluster. If the minimum distance between the case and the raw cases in the closest cluster exceeds the distance threshold of the cluster, the case will be reassigned to the next closest cluster if the case is within the distance threshold of the cluster. The rest of the cases that cannot be assigned to any cluster will be filtered out. The rationale for removing cases to improve the generation quality is based on the assumption that the case far from the cluster has an unrealistic shape.
Figure 3.10 shows both the remaining and removed synthetic cases compared to the raw case in Cluster 6. The shape of the cases in the second column of Figure 3.10 looks similar to the shape in the first column, but the shape in the third column looks distorted from the first column. The above discovery illustrates that the cluster filter with the distance threshold in Table 3.3 can filter out noisy synthetic cases and clean up the generation. Figure A.1, A.2, and A.3 in the Appendix shows the results of the rest Cluster (Cluster 1~5).

3.2.3.2 Cluster Re-weighting

In order to achieve a more similar distribution of patterns between the training dataset and the synthetic dataset, it is essential to re-assign the sample weight in each cluster. Each case remains after the filter in cluster c could be re-weighted by multiplied with a coefficient α_c.

$$\alpha_c = \frac{\Sigma W_{\text{raw}_c}}{\Sigma W_{\text{cutoff}_c}} \tag{3.10}$$

where the ΣW_{raw_c} represents the sum of weights of all raw cases in cluster c, and the $\Sigma W_{\text{cutoff}_c}$ represents the sum of weights of all remaining synthetic cases in cluster c. The re-weighting result can be verified in Table 3.4. The overall performance of the distribution is improved, especially for Clusters 2-4. As a result, the re-weighting function can balance the proportion of the sum of weights of each cluster, thereby modifying the proportion of the synthetic cases to align with that of the training dataset.

3.3 Evaluation

As there is no standardized method for evaluating the performance of synthetically generated cases, the selection of appropriate metrics to evaluate the quality of synthetic crashes is a crucial but challenging task [19, 20]. One of the most common and intuitive evaluation methods is to visually inspect the synthetic crashes directly. The

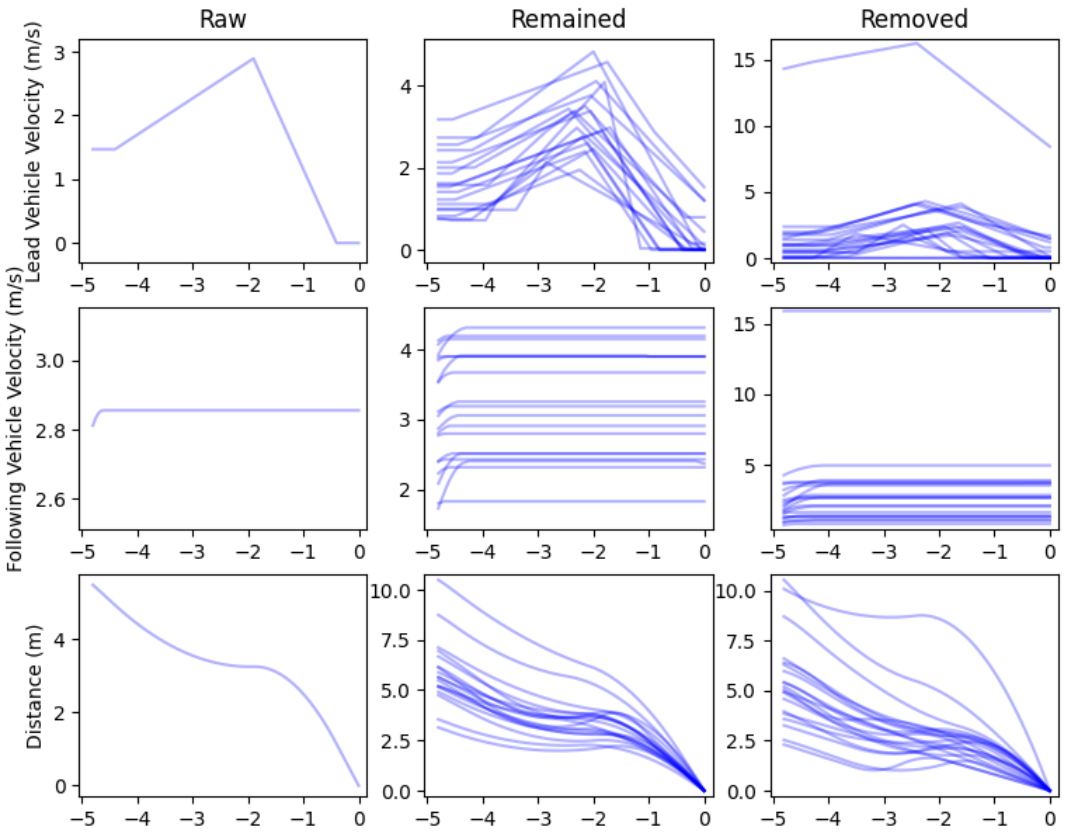


Figure 3.10: Visual comparison between the raw case(s), the remaining generation, and the removed generation of Cluster 6.

visualization can be conducted in several different dimensions to inspect the synthetic cases comprehensively. However, as visual inspection cannot be rigorously quantified, visual inspection is a subjective evaluation metric. A relatively objective method of evaluation is the distribution test. Since the goal of this thesis is to generate synthetic crashes that come from the distribution of real-world crashes, if the distribution test results indicate that both the synthetic and real-world crashes come from the same distribution, the generation quality can be considered satisfactory.

3.3.1 Visualization

The crash kinematics could be visualized in several different dimensions.

3.3.1.1 Trajectory Shape

The most intuitive method for inspecting the generation quality is to visualize the trajectory shape directly. In order to prevent the plot becoming excessively crowded (i.e., with too many trajectories in one plot), the synthetic trajectories could be randomly sampled from different clusters and visualized separately for each cluster. What's more, the proportional distribution of the number of trajectories within each cluster could also be checked by comparing the proportional distribution of both the raw and synthetic cases in a histogram.

Cluster	1	2	3	4	5	6
Raw (Validation)	9.49	3.08	18.09	13.65	55.48	0.19
Synthetic (w/o pp)	10.04 (+5.8%)	6.33 (+104.8%)	13.19 (−27.1%)	17.85 (+30.8%)	52.28 (−5.8%)	0.31 (+64.4%)
Synthetic (w/ pp)	10.30 (+8.4%)	2.66 (−14.0%)	18.88 (+4.37%)	15.86 (+16.1%)	52.28 (−5.8%)	0.02 (−89.1%)

Table 3.4: Proportion of the sum of weights of each cluster. Both synthetic cases before post-processing (w/o pp) and synthetic cases after post-processing (w/ pp) are tested. The percentages in the table indicate the ratio between the proportion of synthetic cases and raw cases in each cluster.

3.3.1.2 T-SNE Projection

As previously mentioned (Section 2.4.1), T-SNE projection is a scatter plot that plots the resulting low-dimensional representation of the crash kinematics to interpret the crash. The raw cases and synthetic cases can be plotted as points using different colors. Suppose the occupation areas of the points of the generated cases and the points of the original cases coincide with each other. In that case, the generated cases are considered to have the same pattern as the original cases. In contrast, the points of synthetic cases that are situated at a considerable distance from the points of raw cases could be regarded as diverging from the original distribution.

3.3.1.3 Patterns Distribution

Ideally, the proportion of the sum of the weights in each cluster of the synthetic cases should have the same proportion as the raw cases, as shown in Table 3.1. Moreover, the performance of the re-weighting function proposed in Section 3.2.3.2 should also be verified for the test data.

3.3.2 Distribution

To demonstrate that the generated trajectories are reasonable, the distribution of the parameters of the generated trajectories should be similar to the original trajectories. Making comparisons between individual trajectories is inappropriate since multiple plausible trajectories can be generated under the same initial conditions. The method used in this thesis to determine if two groups of data are drawn from the same distribution is the KS-test. The following section introduces the parameters that will be tested.

3.3.2.1 Velocity & Distance on each timestamp

A time series trajectory consists of several data points depending on the duration of the trajectory. If the generated trajectories match the distribution of the original trajectories, the value distribution at each timestamp between the original and the

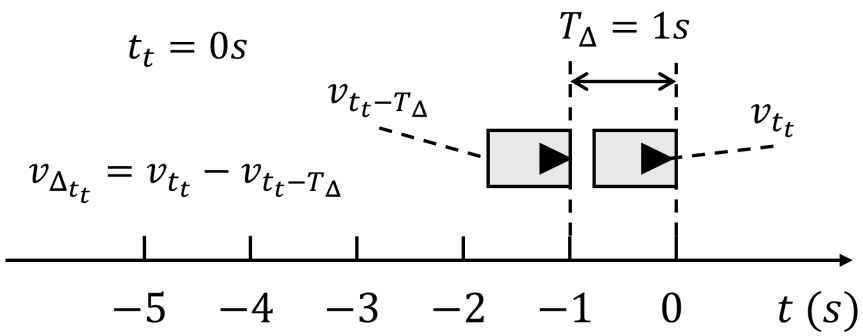


Figure 3.11: Vehicle velocity difference v_Δ with a time interval $T_\Delta = 1s$ at test time $t_t = 0s$

generated trajectories should be similar. The distribution test can be used to evaluate the quality of the generated trajectories, including the time-series of the velocity of both vehicles ($v_{F_{T:0}}$ and $v_{L_{T:0}}$) and the distance ($d_{T:0}$).

3.3.2.2 Velocity Difference in a time interval

As Figure 3.11 illustrated, velocity difference v_Δ in a time interval T_Δ is the difference between the velocity at test time t_t (v_{t_t}) and the velocity at the time that T_Δ before t_t ($v_{t_t - T_\Delta}$), (i.e., the velocity at the time $t_t - T_\Delta$). Although the evaluation introduced in Section 3.3.2.1 could test the value distribution on each timestamp, the distribution of variation of the value over time couldn't be tested. By testing the v_Δ with a time interval $T_\Delta = 1s$, the distribution of velocity variation of both vehicles could be tested.

3.3.2.3 Minimum Acceleration

The minimum acceleration, i.e., the maximum deceleration that the vehicle applies, is an important indicator that describes driver behavior in rear-end crash analysis. Therefore, the distribution of the minimum acceleration is worth testing.

3.3.2.4 Delta-V

Although the Delta-V of each synthetic crash is fixed and determined before generation since Delta-V can be fully derived from the initial condition I_F and I_L, the distribution of Delta-V still requires testing since the cluster post-processing removes and re-weights the generation dataset. The Delta-V could be calculated according to the Equation 2.4 shown in Section 2.1.3.

3.3.2.5 No-return time

The no-return time could be determined by predicting both vehicle positions in the future. However, during the prediction process, the FV should apply the maximum braking deceleration. According to the definition of the PONR, the FV is assumed to brake with acceleration equal to $-9m/s^2$, which could be recognized as the maximum acceleration that the FV is able to apply. Since the post-collision trajectory

of the LV could not be accessed, the post-collision trajectory of the LV could be made up by assuming that the LV maintains the same acceleration as a_{F_0} after the collision. A constant acceleration motion model (Equation 3.11) could be employed to predict both vehicles' positions. By searching the timestamp backward from the first timestamp (i.e. the first timestamp before the collision happened), if the predicted distance between two vehicles is never less than or equal to zero, the previous timestamp could be considered as the no-return time.

$$\begin{bmatrix} x_{p_{t+\Delta t}} \\ x_{v_{t+\Delta t}} \\ x_{a_{t+\Delta t}} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{p_t} \\ x_{v_t} \\ x_{a_t} \end{bmatrix} \tag{3.11}$$

4
Results

This section presents the results of the synthetic collisions generated using the methodologies outlined in this paper, along with an evaluation of their fidelity. The results are divided into two categories: a visual representation and probability distributions.
The initial condition used to generate the synthetic cases presented in this section is identical to that of the test dataset of the raw data.

4.1 Visual Representation

4.1.1 Synthetic Crash Kinematics Visualization

The kinematics of the synthetic cases and the raw cases, both labeled Cluster 5, are visualized together in Figure 4.1. The red bold lines in Figure 4.1 represent the weighted mean of all synthetic cases in Cluster 5, while the blue bold lines represent the raw cases. The red and blue bold lines are close, indicating that the generation quality for the synthetic cases in Cluster 5 is satisfactory.

4.1.2 T-SNE Projection

Figure 4.2 shows the T-SNE projection, where the red dots represent the raw cases and the blue and green dots represent the synthetic cases before and after post-processing. Generally, most of the blue dots are far away from the red dots, which indicates that the post-processing can remove the unrealistic cases. Moreover, the coverage areas of the green dots are similar to the coverage areas of the red dots, representing synthetic cases with a similar pattern to the raw cases. The location of both red and green dots could be further interpreted. Some of the green dots are situated close to one or a group of red points, yet do not overlap. This may be an indication that the synthetic cases have a similar pattern to the raw cases, but the latent parameters of the synthetic cases vary from the raw cases. The discovery of green points near the red points serves to illustrate that the model is capable of increasing the quantity of the dataset by expanding the cases for specific data patterns. Moreover, some green dots are situated between two groups of red dots. This discovery demonstrates that the generation model is capable of generating patterns that interpolate between two patterns of the raw dataset, thereby expanding the diversity of the raw dataset.

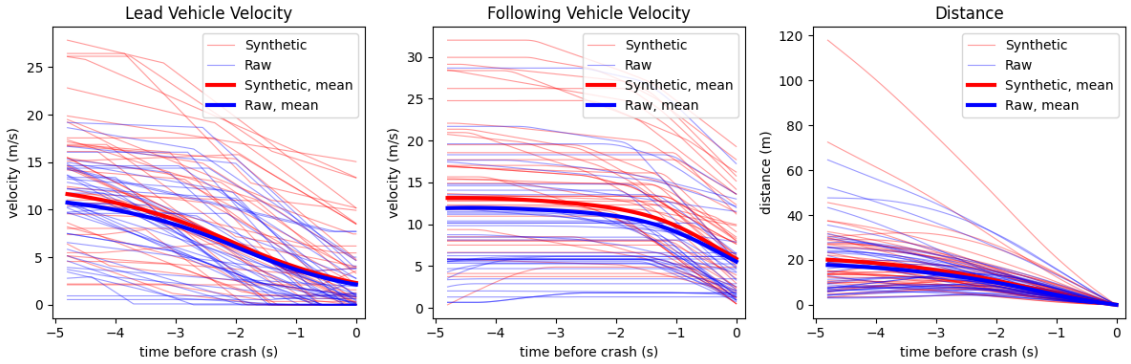


Figure 4.1: Crash kinematics of both raw and synthetic crashes in Cluster 5 (40 randomly sampled for the raw and synthetic crashes respectively)

Cluster	1	2	3	4	5	6
Raw (Test)	9.67	3.03	18.26	13.45	55.40	0.19
Synthetic (w/o pp)	10.13 (+4.7%)	5.36 (+77.1%)	12.22 (−33.1%)	18.23 (+35.5%)	54.03 (−2.5%)	0.04 (−78.5%)
Synthetic (w/ pp)	9.12 (−5.7%)	2.39 (−20.9%)	17.73 (−2.9%)	15.97 (+18.7%)	54.78 (−1.1%)	0.01 (−93.5%)

Table 4.1: Proportion of the sum of weights of each cluster. Both synthetic cases before post-processing (w/o pp) and synthetic cases after post-processing (w/ pp) are tested. The percentages in the table indicate the ratio between the proportion of synthetic cases and raw cases in each cluster.

4.1.3 Patterns Distribution

The proportion of the sum of weights of each cluster is shown in Table 4.1. In general, the proportion distribution of the synthetic cases after post-processing is similar to that of the raw cases. After post-processing, the ratios between the proportions of synthetic cases and raw cases are closer to 100%, indicating that the post-processing improves the proportion distribution.

4.2 Probability Distributions

4.2.1 Kinematics at every timestamp

Figure 4.3 shows the p-value computed by the KS-test, which compares the kinematics between the test dataset and the synthetic dataset generated by using the same initial condition as the test dataset. The blue curve in Figure 4.3 represents the p-value results for the synthetic cases before the cluster post-processing, and the red curve represents the p-value results for the synthetic cases after the cluster post-processing. In Figure 4.3, all curves are always above the line of threshold ($p = 0.05$), meaning that the kinematics of synthetic cases, including $v_{L_{T:0}}$, $v_{F_{T:0}}$, and $D_{T:0}$, come from the same distribution as the test dataset of the raw cases

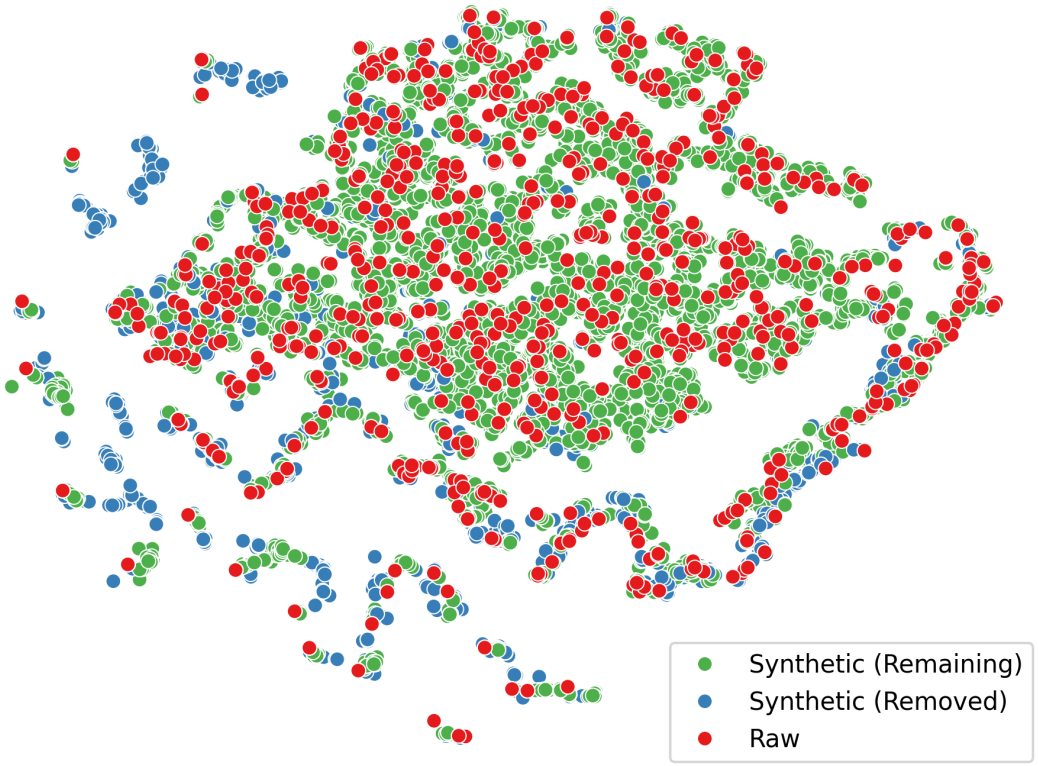
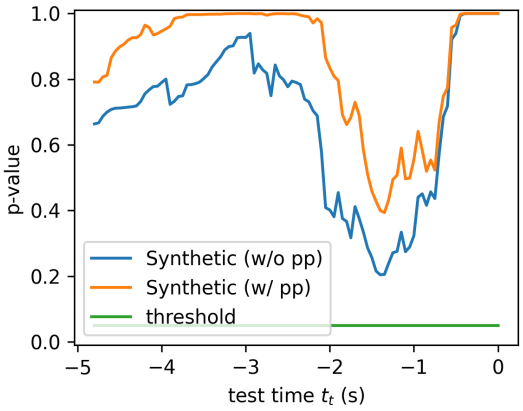


Figure 4.2: T-SNE projection of the raw and synthetic cases

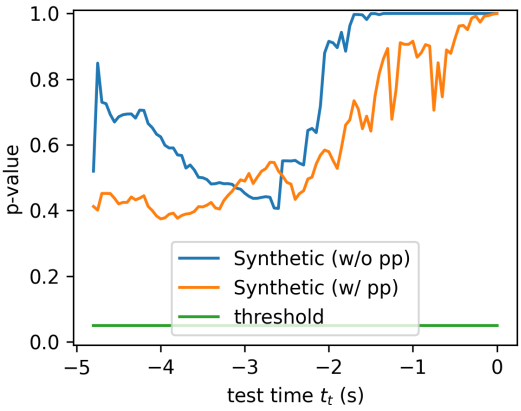
during $5s$ before the collision happened.
Figure 4.4 compares the cumulative distribution of the kinematics at $t = -3$ (i.e. $3s$ before the collision occurred), showing that there is no significant difference between the distributions of the raw cases and the synthetic cases.

4.2.2 Minimum acceleration

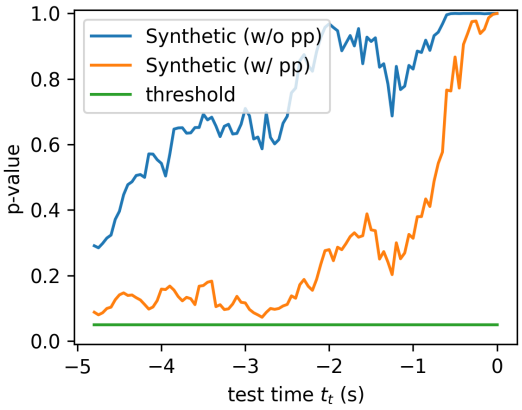
Figure 4.5 illustrates the cumulative distribution of the minimum acceleration of both LV and FV between the raw and synthetic datasets, where the blue curve represents the raw dataset, and the orange and green curves represent the synthetic dataset before and after cluster post-processing, respectively. According to the left graph in Figure 4.5, the distribution of the minimum acceleration of synthetic LV is close to the raw dataset. However, the distribution of the synthetic FV data is distorted from the raw FV data. In the synthetic cases before post-processing, the proportion of $-0.25m/s^2$ is too large compared to the raw cases, but the sum of the proportion of both $0m/s^2$ and $-0.25m/s^2$ is almost equal to the proportion of $0m/s^2$ of the raw cases. The post-processing could filter and re-weight the cases that have a minimum acceleration $-0.25m/s^2$, but the proportion of $0m/s^2$ still has a big difference compared to the raw cases. Nevertheless, according to Table 4.4, both LV and FV distribution of minimum acceleration pass the KS-test, indicating that the distribution of minimum acceleration of synthetic crashes has the same distribution



(a) LV Velocity (V_L)



(b) FV Velocity (V_F)



(c) Distance between LV and FV (D)

Figure 4.3: KS-Test for the kinematics, including V_L, V_F, and D, in $5s$ before the collision happened. Both synthetic cases before post-processing (w/o pp) and synthetic cases after post-processing (w/ pp) are tested.

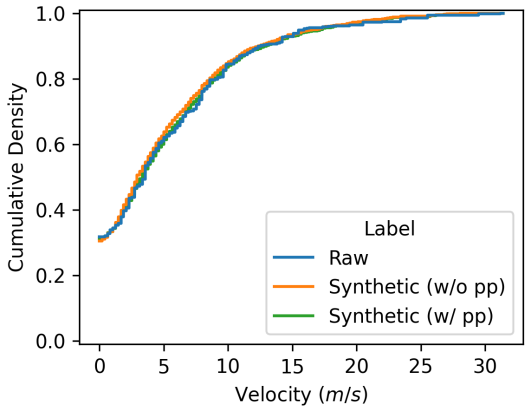
as the raw cases.

4.2.3 Delta-V

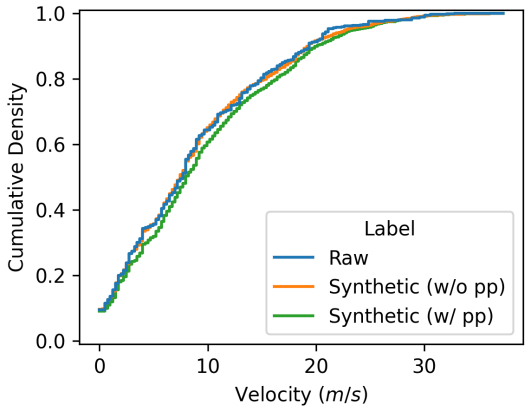
Although Delta-V could be controlled to be identical to the raw cases at the beginning of the case generation since the Delta-V could be fully derived from the model inputs, the distribution of Delta-V still needs to be tested since the post-processing varies the distribution of Delta-V. Figure 4.6 compares the distributions between the raw cases and the synthetic cases, and Table 4.3 shows the KS-test results of the distributions of Delta-V. Both Figure 4.6 and Table 4.3 indicate that the distribution of Delta-V of the synthetic cases is highly similar to the raw cases.

4.2.4 Velocity Difference

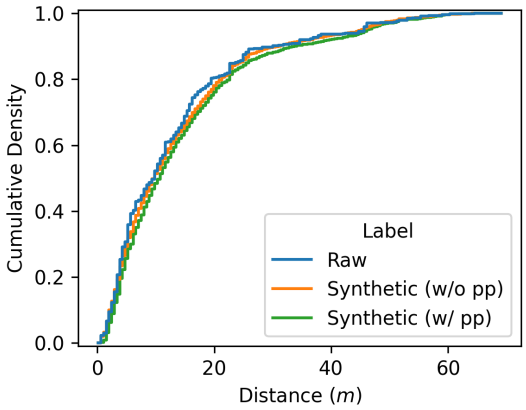
Figure 4.7 shows the distribution of velocity difference in $1s$ between raw and synthetic cases. The meaning and the calculation method of velocity difference are



(a) LV Velocity (V_L)



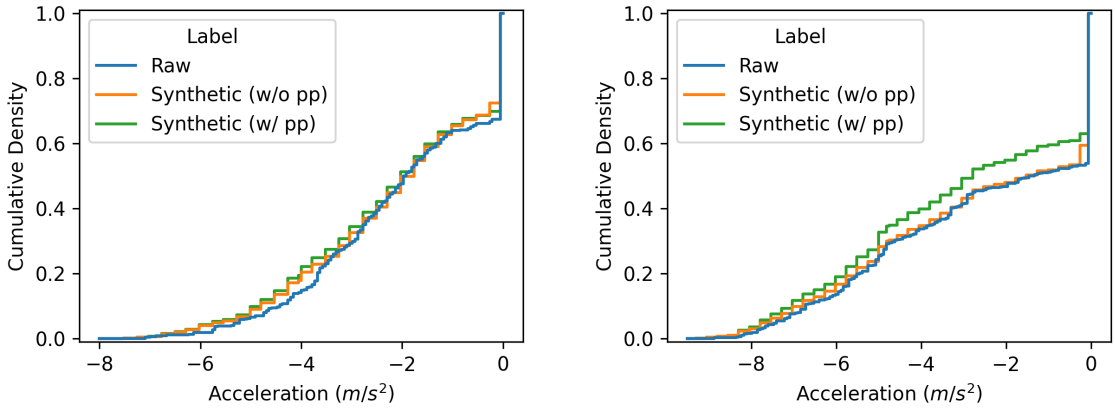
(b) FV Velocity (V_F)



(c) Distance between LV and FV (D)

Figure 4.4: Cumulative distribution of the kinematics, including V_L, V_F, and D, on $t = -3s$ before the collision happened, including raw cases, synthetic cases before post-processing (w/o pp), and synthetic cases after post-processing (w/ pp).

explained in Section 3.3.2.2. The curves of the synthetic cases in the left graph of Figure 4.7 are always above the threshold, indicating that the distribution of the velocity difference of the LV passes the KS-test throughout the crash. However, in the last $1s$, the distribution of the velocity difference of the synthetic FV could not pass the KS-test. What's more, the FV's distribution of the synthetic crashes after the post-processing is on the edge of the threshold and seems to be even worse than the crashes before the post-processing. One hypothesis for the cause of this phenomenon is that the cluster removes some specific cases that undermine the overall distribution. Figure 4.8 shows the distribution of velocity difference between $t = 0s$ and $t = -1s$. According to the right graph of Figure 4.8, some $v_\Delta = 0$ cases are removed by the cluster post-processing. As a result, the post-processing process still needs to be improved to avoid removing cases that undermine the distribution.



(a) LV (b) FV

Figure 4.5: Cumulative distribution of the minimum acceleration of LV and FV, including raw cases, synthetic cases before post-processing (w/o pp), and synthetic cases after post-processing (w/ pp).

Table 4.2: P-value of KS-test for the cumulative distribution of the minimum acceleration of LV and FV.

	Before Post-processing	After Post-processing
LV a_{min}	0.332	0.103
FV a_{min}	0.459	0.054

4.2.5 No-Return-time

No-return-time refers to the timestamp on the PONR, which is a metric introduced by domain knowledge. The distribution of no-return-time is worth testing to validate the fidelity of the synthetic crashes. Figure 4.9 illustrates the distribution of no-return-time and Table 4.4 shows the KS-test result of the distribution of no-return-time. Both Figure 4.9 and Table 4.4 indicate that the distribution of no-return-time between the raw and the synthetic dataset is highly similar.

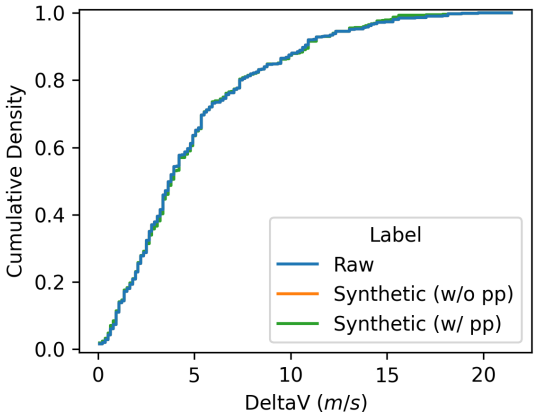


Figure 4.6: Cumulative distribution of Delta-V of both Raw and Synthetic dataset, including raw cases, synthetic cases before post-processing (w/o pp), and synthetic cases after post-processing (w/ pp).

Table 4.3: P-value of KS-test for the cumulative distribution of the Delta-V

	Before Post-processing	After Post-processing
Delta-V	1.000	1.000

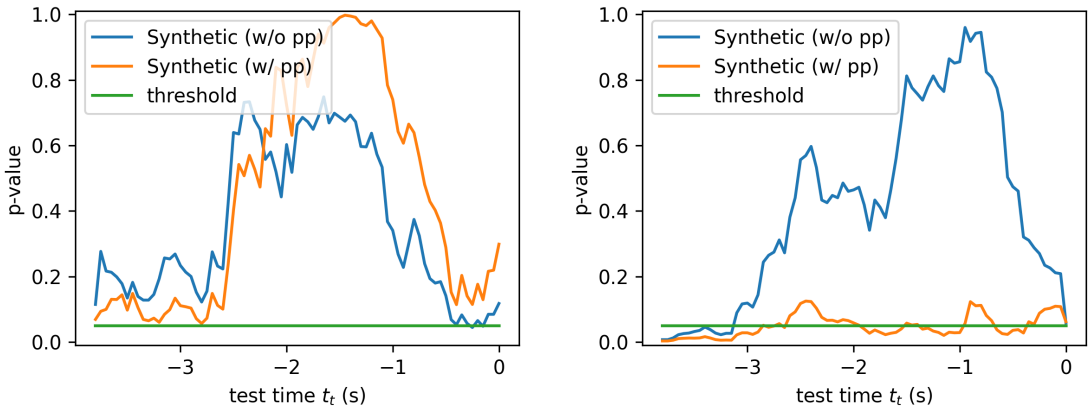


Figure 4.7: KS-Test for the velocity difference with a time interval $T_\Delta = 1s$ during $5s$ before collision. Both synthetic cases before post-processing (w/o pp) and synthetic cases after post-processing (w/ pp) are tested.

Table 4.4: P-value of KS-test for the cumulative distribution of the no-return-time

	Before Post-processing	After Post-processing
no-return-time	0.999	0.999

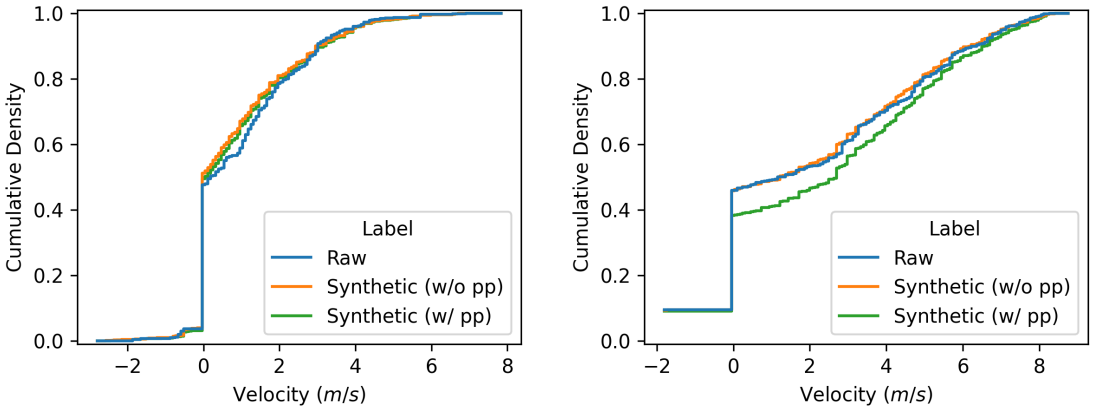


Figure 4.8: Cumulative distribution of the velocity difference between $t = 0s$ and $t = -1s$ (at test time $t_t = 0s$), including raw cases, synthetic cases before post-processing (w/o pp), and synthetic cases after post-processing (w/ pp).

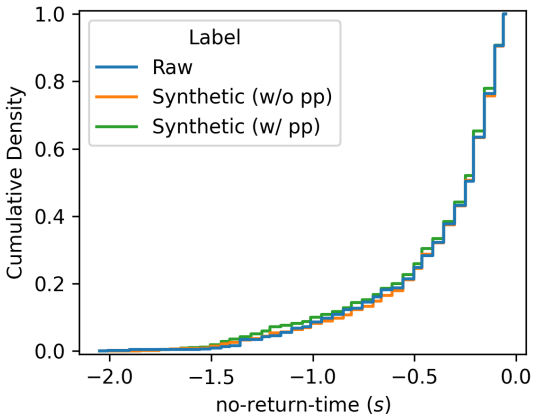


Figure 4.9: Cumulative distribution of no-return-time of both Raw and Synthetic dataset, including raw cases, synthetic cases before post-processing (w/o pp), and synthetic cases after post-processing (w/ pp).

5

Discussion

5.1 Answers to research questions

5.1.1 Can a Behavioral Cloning model trained on a set of crash kinematics be used to generate crashes that are representative of the training set?

The Behavioral Cloning model can be used to generate representative crashes. The number of cases in the raw dataset may be insufficient to derive an accurate distribution for representative case analysis. According to the evaluation result, the proposed model shows its potential to generate synthetic crash kinematics with high fidelity. As a result, the proposed model can generate high-fidelity cases to supplement the raw dataset and complement patterns that have few cases or introduce more patterns. The supplementary data could enhance the robustness of the distribution of the raw dataset to derive accurate representative cases.

One of the most intuitive methods for identifying representative crashes is to utilize an appropriate clustering algorithm to identify all patterns within the raw dataset. However, this method may be too difficult to implement. One of the most important reasons is that it's hard to provide an objective assessment of the final representative result. Different clustering setups and algorithms yield different representatives, but the results are difficult to compare since there is either no recognized algorithm for comprehensive evaluation or a recognized conclusion on how many representative cases should be found. Consequently, the result should be visually inspected to verify and thus subjectively interpreted. However, the number of representatives could be too large (e.g. over 20), making visual inspection difficult. Moreover, all of the results could be interpreted rationally (i.e., there are several reasonable results), making it hard to determine the optimal set of representative crashes.

Nevertheless, the question can be addressed through simplification. The crash generation framework presented in this thesis allows the generation of an infinite number of crashes for one specific initial condition (I_F and I_L). Since the crash patterns for one specific initial condition would be much less complex than the patterns in the entire raw dataset, the above clustering method could effectively discover the representative crashes for the given initial condition. By analyzing the joint distribution of I_F and I_L, the representative initial conditions could be derived. Then, the representative crashes of the entire raw dataset could be derived by analyzing the representative crashes of each representative initial condition.

5.1.2 How do different amounts of training data samples affect the generation model performance?

In general, the larger the training dataset, the more effective the generation model will be, as a larger dataset provides a more diverse and comprehensive learning environment, which in turn leads to enhanced performance [21].

Section A.4 and A.5 in the Appendix shows the evaluation results when the size of the training dataset is 30% and 20% of the raw dataset, respectively. The results shown in Section A.4 are similar to Section 4, which is generated by the model that trained by 70% of the raw dataset. However, according to the results shown in Section A.5 (1000 cases), the distribution of generated cases is distorted from the distribution of the test dataset. This finding indicates that the model capable of generating high quality synthetic crashes requires at least 1500 cases to train.

5.1.3 How sensitive is the model to the underlying training data samples?

According to the consensus, a biased training set will distort the model output. Therefore, the distribution of the patterns in the training dataset should be carefully controlled. Ideally, the model will reflect the same pattern distribution as the training dataset. However, according to the results shown in Table 4.1 the distribution of the patterns before the cluster post-processing distorted from the raw dataset, indicates that the BC model could not replicate the same distribution of patterns in the training dataset. The possible reasons could be:

- **Model issue:** The model has an upper limit that cannot reproduce the same pattern distribution as the training dataset.
- **Input issue:** The initial condition used for crash generation is not the same as the initial condition of the raw dataset, causing the model to generate a distorted distribution of patterns.

The model could be modified to mitigate the problem of model limitations. Several specific models could be trained for different kinds of crash patterns. Moreover, indicators that refer to the pattern categories could also be added as one of the model inputs. However, to implement these strategies, the joint distributions between the model pattern categories and the initial condition should be modeled, as the crash categories may not uniformly distribute for one specific initial condition. Furthermore, the classification of patterns should be accurate, since poor classification quality will accumulate the error in the model.

In general, the input issue seldom happens since the input for crash generation is identical to the initial condition of the test dataset, which is randomly sampled from the raw dataset. However, currently, the sampling process only controls the distribution of patterns. To be more rigorous, the distribution of initial conditions of the train/validation/test dataset should also be controlled to be the same as the entire raw dataset.

5.2 A mathematical analysis of the Behavior Cloning for vehicle kinematics modeling

The thesis aims to generate synthetic rear-end crashes that obey the distribution of the existing crash dataset. Instead of directly modeling the distribution of the kinematics over the entire duration of the LV and FV ($P\left(\boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}}\right)$), the distributions of the kinematics action of the LV and FV at each timestamp (i.e., the acceleration at each timestamp) under different given backgrounds were modeled. The distribution of the kinematics action of the LV and FV at time t could be represented as Formula 5.1 and 5.2, respectively.

$$P\left(a_{L_t} | v_{L_{t+\Delta t:0}}, a_{L_{t+\Delta t:0}}, v_{F_0}, a_{F_0}\right) \tag{5.1}$$

$$P\left(a_{F_t} | v_{F_{t+\Delta t:0}}, a_{F_{t+\Delta t:0}}, v_{L_{T:0}}, a_{L_{T:0}}\right) \tag{5.2}$$

where v_{L_t} and v_{F_t} represents the velocity of LV and FV on time t, and a_{L_t} and a_{F_t} represents the acceleration of LV and FV on time t.
The modelling of both Formula 5.1 and 5.2 could indirectly reflect $P\left(\boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}}\right)$, and the mathematical derivation is shown in Section A.1 in the Appendix.

5.3 Model Set-up

5.3.1 Behavior Generation Direction

Generally, the input of the driver behavior model is the history observation and action of the driver. Therefore, it is natural to think that the model takes the information 5s before the collision as the initial input to the model, and then the model generates the behavior step by step, thus composing a 5 seconds trajectory. However, the most readily available and important data for crash analysis is the data at the moment of collision, including the velocity of the LV and FV, as well as the Delta-V. As a result, in contrast to most of the BC models, the model in this thesis generated the vehicle behavior backward.

5.3.2 Input of the Model

The choice of input of the neural network should be made with care. The absence of essential inputs can result in the model output being biased or overly generalized far from desired, yet the addition of non-pivotal inputs could lead to the overfitting of the model and makes the model difficult to capture the true relationship between the inputs and the outputs.
Based on common sense, the behavior of the LV is independent of the FV, but the model of LV in this thesis takes the initial condition of FV as one of the inputs. To explain the reason specifically, an initial condition of LV corresponds to different kinds of LV trajectories, but while given a corresponding initial condition of FV, some LV trajectories are either impossible or too rare to occur in the real-world.

According to the mathematical analysis in 5.2 (specifically, the Equation A.6), both I_F and I_L should be set as the input of the LV model to generate the behavior the same as the distribution of the raw dataset. However, the mathematical equation could not prove that I_F affects the distribution of the LV's behavior, since the equation still works even if I_F is independent of the LV's behavior. To rigorously prove that I_F affects the distribution of the LV's behavior, the correlation test between I_F and the LV trajectory could be implemented. Since the length of the LV's velocity time-series (equal to 100) is too long for testing, the mean value of the LV's velocity is used instead. Table 5.1 shows the correlation test result. The measured variables are highly correlated if the absolute correlation value is greater than 0.5 [37]. Table 5.1 shows that the mean value of the LV's velocity is correlated to v_{F_0} and a_{F_0}, which is the I_F. Thus, I_F correlates with LV's trajectory. As a result, I_F should be included in the input of the LV model.

Table 5.1: Weighted correlations between initial conditions of the following vehicle and the lead's speed profile

	a_{F_0}	v_{F_0}	μ_{v_L}
a_{F_0}	1	−0.184	−0.57
v_{F_0}	−0.184	1	0.579
μ_{v_L}	−0.57	0.579	1

In the early stages of this thesis work, the LV model input did not include I_F. Sometimes, the crash could not be generated because some of the elements in $d_{T:-\Delta t}$ are smaller than 0. By introducing I_F into the LV's model, the successful generation rate (i.e., no element in $d_{T:-\Delta t}$ is smaller than 0) improved from 80% to 95%, which means that the generation quality was improved. As a result, introducing I_F as one of the LV model inputs is reasonable.

5.4 Analysis of the advantages of the model

5.4.1 High transparency and a certain degree of interpretability in vehicle kinematic generation

In contrast to DGMs, which directly generate the entire crash kinematics, the crash generation model proposed in this thesis generates the vehicle action to synthesize the entire crash. The generation methodology allows us to have access to the action-decision process. Although how the neural network computes the vehicle action distribution is still too difficult to interpret, at least the proposed model could visualize the action distribution and thus visualize how the crash kinematics are generated. As a result, compared to DGMs, the proposed model has a higher transparency and interpretability in the vehicle kinematic.

5.4.2 High controllability

High controllability refers to a feature of the model that can reproduce crashes with specific settings, rather than randomly generate one crash case [1]. Different crash kinematics could be generated by the proposed model when some specific parameters are fixed. Here are several potential scenarios that the proposed model could apply.

- **The model could generate synthetic cases with control of the initial condition.**
  The inputs of the proposed model are the kinematics of LV and FV when the collision happens (i.e., I_L and I_F), which could directly control the initial condition of the crashes. In contrast, most DGMs generate cases by inputting a (Gaussian) noise vector, thus the initial condition of the generated cases cannot be controlled.
- **The model could generate variations of the existing crashes.**
  Since the output of the BC model is a probability distribution of actions, the kinematic generation model could generate different actions even if the inputs are identical. Thus, the crash generation model could generate an infinite number of crashes for a given initial condition, and then the variations of crashes for this initial condition could be fully explored.
  What's more, action generation can start at any timestamp before the collision, allowing for controlling longer crash kinematics before the collision, rather than just when the collision happens. An example could be used to describe this scenario. Given kinematics on the last $1s$ before the collision, the model could start generating a $4s$ crash kinematics at $t = -1s$, thus fixing the kinematics on the last $1s$. This generation method allows the analysis of the variations of crash kinematics that have fixed kinematics in a few seconds before the collision.
- **The model could elongate the existing crashes.**
  The proposed model can start generating the vehicle's action at any timestamp before the collision to synthesize the entire crash. As a result, the model could complement the existing cases that has short crash kinematics (e.g., only $1s$ $3s$ kinematics before the collision is available).
- **The model could modify the action on a specific timestamp to modify the subsequent generation.**
  Since the action-decision process of the proposed model could be accessed, the action generated by the model could be controlled. As a result, the action generated by the model could be modified by some predefined rules.
  Currently, the action is generated by randomly sampling the probability distribution. The sampling rule could also be modified, e.g., sampling the maximum expectation of the distribution as the action.

5.5 Limitation and Outlook

The methodology presented in this thesis has many shortcomings and potential areas for improvement.

5.5.1 Cluster tuning

5.5.1.1 Cluster Algorithm Selection

Both K-means and Hierarchical Clustering are popular clustering algorithms for time-series clustering. K-means has several advantages, including a faster clustering speed, a simple implementation, and a high degree of interpretability for centroids. However, the inherent limitations of K-means render it unsuitable as a clustering method for the specific nature of the crash kinematics data.

Firstly, K-means assumes clusters are spherical (or globular), which may not be true for crash kinematics data. The quality of the generated case closer to the centroid may not be better than the case that is further away from the centroid, which means that the distance between the case and the corresponding centroid is not a rigorous metric for evaluating the quality of the generated case. Thus, the cluster filter based on one single threshold distance for each cluster is not robust. Moreover, the parameters of k-means, including the number of clusters and initialization, should be well-tuned to determine the optimal cluster. However, the tuning process is challenging and cannot be guaranteed to be rigorous. Generally, the number of clusters could be determined based on human domain knowledge or some commonly used metrics (eg., elbow method, silhouette score, etc.). However, crash kinematics does not have a recognized classification standard, which means the number of clusters needs to be determined by numerical metrics. Although different metrics evaluate the cluster quality in different dimensions, there is no general metric to evaluate the cluster quality comprehensively. What's more, the outputs of some specific metrics represent the average performance of each cluster number rather than the peak performance. Tuning the number of clusters and initialization should therefore be done at the same time, otherwise, the optimal combination of the number of clusters and initialization that represents peak performance may be missed. Last but not least, the optimal parameters selected by the metric do not represent the cluster as optimal, which means that the parameter selection should be visually inspected based on the underlying meaning of the results. In conclusion, the tuning process is quite complex, and the optimal tuning could not be rigorously proven.

Hierarchical Clustering is a better choice for crash kinematics data. Firstly, Hierarchical Clustering can capture complex cluster shapes and nested structures, which is beneficial for time-series data with varied patterns. Moreover, in contrast to K-Means, where the result is affected by initialization, the results of the Hierarchical Clustering remain unchanged when the internal algorithm and parameters (e.g., the number of clusters) are determined. What's more. the dendrogram provides a visual representation of the hierarchical structure of the crashes kinematics data, making it easier to understand the relationships between clusters and to determine the number of clusters. As a result, Hierarchical Clustering has the potential to produce superior performance on crash kinematics data.

The application of Hierarchical Clustering is also associated with some challenges. Firstly, Hierarchical Clustering could not be directly applied to new datapoints (i.e., assign labels to new datapoints). By training some classification model (e.g., k-nearest neighbors) or set some self-defined rules based on the distance matrix, the new datapoints could be assigned with a proper label from the Hierarchical

Clustering. In addition, Hierarchical Clustering has many types of linking algorithms (e.g. single, complete, etc.) that need to be chosen carefully. Each linkage algorithm has its pros and cons, and the choice of linkage algorithm should be based on either understanding the nature of the data or inspecting the cluster results. What's more, Hierarchical Clustering is more computationally intensive than K-means, which means it takes much longer to cluster. Nevertheless, the disadvantages of Hierarchical Clustering could be overcome in several proper ways, and Hierarchical Clustering would still be the optimal choice for crash data clustering.

5.5.1.2 Potential improvements

- **Clustering Algorithm:** The classification given by the current clustering algorithm is still too general, which means that more detailed data patterns could be divided from the current classification, especially Cluster 5 (Figure 3.4). A cluster that is too general cannot fully reflect the internal structure and nuances of the data within the cluster, meaning that the internal diversity could not be discovered.
  The rough cluster affects the sampling of the training dataset, making the training dataset biased. What's more, the reliability of the re-weighting process is diminished when the cluster is too general, since applying a uniform coefficient to different patterns in a rough cluster will result in an imbalanced distribution of the weight of the pattern.
  However, simply adding the number of clusters cannot find more detailed data patterns from the current cluster result. Due to the inherent limitations of the Single Linkage Hierarchical Clustering, a larger number of clusters results in the emergence of a chain of clusters that are outliers. By using secondary clustering with other cluster algorithms (e.g., Complete Linkage Hierarchical Clustering), the cluster could result in more cluster numbers and bring higher interpretability to each cluster (i.e., each cluster contains more similar objects, resulting in less variation within each cluster and more variation between clusters).
- **Cluster Inputs:** Currently, the inputs of the cluster include the normalized LV velocity $V_{L_{T:0}}$, FV velocity $V_{F_{T:0}}$, and the Distance between two vehicles $D_{T:0}$. Although the velocity implies the information of acceleration, introducing the acceleration of both LV and FV ($a_{L_{T:0}}$ and $a_{F_{T:0}}$) as additional two of the input could improve the clustering result, since the acceleration also implies the behaviour patterns.

5.5.2 Deficiencies of Behavior Cloning

Behavior cloning is relatively simple to implement and has high efficiency and flexibility. However, BC is still faced with the following deficiencies.

- **Unknown generalizability:** The BC model always studies the distribution of the training dataset, rather than the human cognitive decision policy. As a result, the BC model seldom reproduces crash kinematic patterns that do not belong to the training dataset. Therefore, to train an ideal BC model, the training dataset should cover as many patterns as possible. What's more, BC

could also overfit the training dataset, making the model unable to reproduce the general policy.

- **Error Compounding:** The crash generation model could produce biased crash kinematics if the BC generates an unrealistic action during the crash generation process. As the unrealistic action generated does not exist in the training dataset, the action distribution calculated in the next timestamp may be biased, and then the entire crash kinematics will be distorted as the error accumulates.

Determining the bad actions generated by the BC model is challenging due to the poor interpretability of neural networks. Some algorithms, such as Generative Adversarial Imitation Learning (GAIL) [23] and Active Inference Driving Agent (AIDA) [24] could be potential replacements for BC.

5.5.3 Distribution Evaluation Method

5.5.3.1 Analysis of Different Evaluation Methods

KS-test is the algorithm used in current evaluation frameworks to assess the similarity between the distributions of the raw and synthetic cases. However, KS-test also has some limitations. Firstly, neither the KS-statistic nor the p-value - the metrics produced by the KS-test - can be strictly used as metrics to measure the difference between two distributions. KS-statistic measures the maximum difference between cumulative distribution functions (CDFs) derived from the given dataset, focusing on the largest discrepancy at a single point. As a result, the KS-statistic may not fully reflect the difference between the two distributions tested. Furthermore, although the p-value indicates whether the two distributions are the same, the amplitude of the p-value doesn't strictly reflect the difference between the two distributions. Last but not least, the p-value becomes more sensitive to differences between distributions when the number of data points increases (i.e. the increasing of the number of data points leads to smaller p-values). Even if the difference between the two distributions remains the same, the p-value can decrease because the KS-test can more confidently reject the hypothesis that the two distributions are not significantly different.

KL-Divergence, a comprehensive metric for measuring both distribution differences, could be an alternative algorithm to the KS test. However, some limitations make KL-Divergence not the first choice of the evaluation algorithm. Firstly, the result (difference between two distributions) calculated by KL-Divergence has unavoidable errors. The required inputs of KL-Divergence are two probability density functions (PDFs). However, the conversion from datapoints to PDF may not be optimal, potentially introducing errors into the PDF and thus affecting the calculation accuracy of the KL divergence calculation. To derive the PDF, the given datapoints must be discretized, but the degree of discretization, which refers to the bin width of the PDF, affects the quality of the derived PDF. A small bin width can capture detailed information from the given dataset, but introduces more noise, especially when the quantity of data is insufficient. A large bin width makes the PDF smoother, but this may result in a loss of detail. One of the mitigation strategies is using kernel density estimation to smooth the PDF, thereby providing a relatively reliable

input for KL-Divergence. In addition, there are also some existing algorithms for searching the optimal bin width. Nevertheless, the conversion error still exists and always affects the KL-Divergence calculation. Last but not least, in contrast to the KS-test, KL-Divergence is unable to determine whether the two distributions have a significant difference. There is no fixed threshold to examine whether the two distributions are the same.

5.5.3.2 Future Improvements

- **Evaluation method:** Both the KS-test and KL-Divergence could be combined in the future to provide a comprehensive assessment of the distributions between the raw and synthetic datasets. The KL-divergence test result could be used to compare the performance of different generation models or different training datasets, while the KS-test indicates whether the model could generate the crash dataset with the same distribution as the raw dataset. What's more, an integrated evaluation metric needs to be designed, as there are currently too many evaluation indicators to be examined simultaneously.
- **The dataset/distribution for comparison:** A 'standard' dataset/distribution should be set as the dataset/distribution to be used for evaluation. Currently, the test dataset, which is split from the raw dataset, is the dataset used for evaluation. However, the distribution of the test dataset is not strictly identical to the raw dataset, making the test less reliable. What's more, the KS-test is affected by the size of the dataset used for evaluation. As a result, the size of the test dataset should be fixed.

6
Conclusion

The thesis proposed a rear-end crash scenario generation framework that can generate a set of synthetic rear-end crashes with a similar distribution to a reference rear-end crash dataset. The generation framework combines both Behavior Cloning and Hierarchical Clustering to control the generation process at both the vehicle behavior level and the whole scenario level, respectively. In addition, variant evaluation methodologies that integrate both general statistical methods and domain knowledge of virtual safety assessment are introduced and implemented to evaluate the crash generation quality. The evaluation results show that the proposed generation framework is feasible to generate satisfactory synthetic rear-end crashes. With a single behavior generation model, synthetic crashes with the same distribution as the raw dataset could be generated, but some synthetic crashes checked out by visual inspection are unrealistic. The post-processing could remove some unrealistic crashes to clean up the generation, but the distribution of synthetic crashes after filtering was distorted from the raw dataset. Enhancing the quality of the clustering to achieve a more precise classification could improve the performance of the existing generation framework. An integrated quantitative evaluation metric also needs to be developed for a comprehensive evaluation.

Bibliography

[1] Zhao, J., Zhao, W., Deng, B., Wang, Z., Zhang, F., Zheng, W., Cao, W., Nan, J., Lian, Y., Burke, A.F. (2023). Autonomous driving system: A comprehensive survey. Expert Systems with Applications. doi: 10.1016/j.eswa.2023.122836.

[2] International, S. (2023). SAE Standards News: J3016 automated-driving graphic update, Retrieved from https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic. Accessed July 6, 2023.

[3] Kalra, N., Paddock, S. M. (2016). Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability? RAND Corporation. doi: 10.1016/j.tra.2016.09.010

[4] Tan, S., Wong, K., Wang, S., Manivasagam, S., Ren, M., Urtasun, R. (2021, June). SceneGen: Learning to Generate Realistic Traffic Scenes. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 892–901. doi:10.1109/CVPR46437.2021.00095

[5] Suo, S., Regalado, S., Casas, S., Urtasun, R. (2021, June). TrafficSim: Learning to Simulate Realistic Multi-Agent Behaviors. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 10395–10404. doi:10.1109/CVPR46437.2021.01026

[6] Đ. Nalić, T. Mihalj, M. Baeumler, M. Lehmann, A. Eichberger, S. Bernsteiner (2020). Scenario Based Testing of Automated Driving Systems: A Literature Survey. doi: 10.46720/f2020-acm-096.

[7] Riedmaier, S., Ponn, T., Ludwig, D., Schick, B., Diermeyer, F. (2020). Survey on Scenario-Based Safety Assessment of Automated Vehicles. IEEE Access, 8, 87456-87477.

[8] Wimmer, P. (2024). HARMONIZED APPROACHES FOR BASELINE CREATION IN PROSPECTIVE SAFETY PERFORMANCE ASSESSMENT OF DRIVING AUTOMATION SYSTEMS. URL: https://www-esv.nhtsa.dot.gov/Proceedings/27/27ESV-000032.pdf

[9] J. Wu, C. Flannagan, U. Sander and J. Bärgman (2024). "Modeling Lead-Vehicle Kinematics for Rear-End Crash Scenario Generation". In: IEEE Transactions on Intelligent Transportation Systems. doi: 10.1109/TITS.2024.3369097.

[10] National Center for Statistics and Analysis. (2023, December). Traffic safety facts 2021: A compilation of motor vehicle traffic crash data (Report No. DOT HS 813 527). National Highway Traffic Safety Administration. URL: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813527

[11] Sullivan, S. (2018, August 28). First harmful event. Mass Crash Report Manual. URL: https://masscrashreportmanual.com/crash/first-harmful-event/

[12] Hydén, C. (1996). Traffic conflicts technique: state-of-the-art. Traffic safety work with video processing, 37, 3-14.

[13] Maaten, L.V., Hinton, G.E. (2008). Visualizing Data using t-SNE. Journal of Machine Learning Research, 9, 2579-2605.

[14] Leifer, J. (2013). A Supplemental Analysis of Selected Two-Vehicle Front-to-Rear Collisions from the NASS/CDS. SAE Technical Paper. doi: 10.4271/2013-01-0223.

[15] Ruthotto L, Haber E. (2021). An introduction to deep generative modeling. GAMM-Mitteilungen. doi:10.1002/gamm.202100008

[16] Krajewski, R., Moers, T., Nerger, D., Eckstein, L. (2018). Data-Driven Maneuver Modeling using Generative Adversarial Networks and Variational Autoencoders for Safety Validation of Highly Automated Vehicles. 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2383–2390. doi:10.1109/ITSC.2018.8569971

[17] Demetriou, A., Alfsvåg, H., Rahrovani, S. et al. (2023). A Deep Learning Framework for Generation and Analysis of Driving Scenario Trajectories. SN COMPUT. SCI. 4, 251. doi: 10.1007/s42979-023-01714-3

[18] van der Heide, A., Tampère, C. M. J., Nicolai, M. (2023). Deep Learning Highway Traffic Scenario Construction with Trajectory Generators. 2023 IEEE Intelligent Vehicles Symposium (IV), 1–8. doi:10.1109/IV55152.2023.10186614

[19] Lucic M, Kurach K, Michalski M, Gelly S, Bousquet O. (2018). Are GANs created equal? a large-scale study. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18). Curran Associates Inc., Red Hook, NY, USA, 698–707.

[20] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X. (2016). Improved techniques for training GANs. Proceedings of the 30th International Conference on Neural Information Processing Systems, 2234–2242. Presented at the Barcelona, Spain. Red Hook, NY, USA: Curran Associates Inc.

[21] Shorten, C., Khoshgoftaar, T.M. (2019). A survey on Image Data Augmentation for Deep Learning. Journal of Big Data. 6. 10.1186/s40537-019-0197-0.

[22] Dean A. Pomerleau. (1988). ALVINN: an autonomous land vehicle in a neural network. In: Proceedings of the 1st International Conference on Neural Information Processing Systems (NIPS'88). MIT Press, Cambridge, MA, USA, 305–313.

[23] Ho, J., Ermon, S. (2016). Generative Adversarial Imitation Learning. Neural Information Processing Systems.

[24] Wei, R., McDonald, A.D., Garcia, A., Markkula, G., Engstrom, J., O'Kelly, M. (2023). An active inference model of car following: Advantages and applications. ArXiv, abs/2303.15201.

[25] Treiber, M., Hennecke, A., Helbing, D. (2000). Congested traffic states in empirical observations and microscopic simulations. Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics, 62 2 Pt A, 1805-24 .

[26] LeCun, Y., Bengio, Y., Hinton, G. (05 2015). Deep Learning. Nature, 521, 436–444. doi:10.1038/nature14539

[27] Bengio, Y., Courville, A.C., Vincent, P. (2012). Representation Learning: A Review and New Perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35, 1798-1828.

[28] Rosenblatt, F. (1957). The Perceptron, a Perceiving and Recognizing Automaton Project Para. URL: https://books.google.se/books?id=P_XGPgAACAAJ

[29] Hornik, K., Stinchcombe, M., White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks, 2(5), 359–366. doi:10.1016/0893-6080(89)90020-8

[30] Cho, K., Merrienboer, B.V., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. Conference on Empirical Methods in Natural Language Processing.

[31] Chung, J., Gulcehre, C., Cho, K., Bengio, Y. (12 2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.

[32] Sepp Hochreiter, Jürgen Schmidhuber. (1997). Long Short-Term Memory. Neural Comput. 9 (8): 1735–1780. doi: https://doi.org/10.1162/neco.1997.9.8.1735

[33] Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems (NIPS), 25, 1097-1105.

[34] Bridle, J. S. (1990). Training Stochastic Model Recognition Algorithms as Networks can Lead to Maximum Mutual Information Estimation of Parameters. Advances in Neural Information Processing Systems (NIPS), 2, 211-217.

[35] Kullback, S., Leibler, R. A. (1951). On Information and Sufficiency. Annals of Mathematical Statistics, 22, 79–86. Retrieved from https://api.semanticscholar.org/CorpusID:120349231

[36] Hinton, G. E., van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. Proceedings of the Sixth Annual Conference on Computational Learning Theory, 5–13. Presented at the Santa Cruz, California, USA. doi:10.1145/168304.168306

[37] Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd ed.). Routledge.

[38] Hastie, T., Tibshirani, R., Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

[39] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11), 2278-2324.

[40] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." Journal of Machine Learning Research, 15(1), 1929-1958.

[41] Jain, A. K., Dubes, R. C. (1988). "Algorithms for Clustering Data." Prentice Hall.

[42] MacQueen, J. (1967). "Some Methods for classification and Analysis of Multivariate Observations." In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, pp. 281-297.

[43] Ester, M., Kriegel, H. P., Sander, J., Xu, X. (1996). "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." In

Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), pp. 226-231.

[44] S. Belongie, J. Malik, J. Puzicha. (2002). "Shape matching and object recognition using shape contexts." IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(4), 509-522.

[45] Estivill-Castro, V. (2002). Why so many clustering algorithms: a position paper. SIGKDD Explor. Newsl., 4(1), 65–75. doi:10.1145/568574.568575

A

Appendix 1

A.1 The mathematical derivation from the distribution of crash kinematics to the Behavior Cloning

The thesis aims to generate synthetic rear-end crashes that obey the distribution of the existing crash dataset. When the distribution is modeled, the crash case could be sampled from the distribution, as the equation A.1 described.

$$\boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}} \sim P\left(\boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}}\right) \tag{A.1}$$

where the variable $\boldsymbol{V}_{\mathrm{FV}}$ is the time-series of FV during the time T before the collision, and the variable $\boldsymbol{V}_{\mathrm{LV}}$ is the time-series of LV during the time T before the collision, and the variable $P\left(\boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}}\right)$ is the distribution of the existing crash dataset.
Assuming that there are 10 possible discrete velocity values at one timestamp (which is super discrete for the real world), and $T = 5s$ with sample frequency $20Hz$ for both $\boldsymbol{V}_{\mathrm{FV}}$ and $\boldsymbol{V}_{\mathrm{LV}}$, the joint distribution $P\left(\boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}}\right)$ requires 10^{200} numbers to describe, which means $P\left(\boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}}\right)$ is impossible to be modeled directly.
To disassemble and simplify $P\left(\boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}}\right)$, the equation A.1 could be expanded as the equation A.2 according to Bayes' theorem.

$$\begin{aligned} P\left(\boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}}\right) &= \frac{P\left(\boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}} | I_F, I_L\right) P\left(I_F, I_L\right)}{P\left(I_F, I_L | \boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}}\right)} \\ &= P\left(\boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}} | I_F, I_L\right) P\left(I_F, I_L\right) \end{aligned} \tag{A.2}$$

where the variable I_F includes both the velocity v_{F_0} and the acceleration a_{F_0} of the FV at the time of the collision, and a similar definition applies to I_L (I_L includes both v_{L_0} and a_{L_0}). The denominator $P\left(I_F, I_L | \boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}}\right)$ is equal to 1 because $\boldsymbol{V}_{\mathrm{FV}}$ and $\boldsymbol{V}_{\mathrm{LV}}$ includes all information about I_F and I_L. The distribution $P\left(I_F, I_L\right)$ is a four-dimensional joint distribution that describes both FV and LV's kinematics when the collision happens. The distribution $P\left(\boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}} | I_F, I_L\right)$ describes the crash distribution given both initial conditions, and the distribution can be further expanded by Bayes' theorem according to equation A.3.

$$P\left(\boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}} | I_F, I_L\right) = P\left(\boldsymbol{V}_{\mathrm{FV}} | \boldsymbol{V}_{\mathrm{LV}}, I_F, I_L\right) P\left(\boldsymbol{V}_{\mathrm{LV}} | I_F, I_L\right) \tag{A.3}$$

The equation A.2 could be re-written as A.4 based on the equation A.3.

$$P\left(\boldsymbol{V}_{\mathrm{FV}}, \boldsymbol{V}_{\mathrm{LV}}\right) = P\left(\boldsymbol{V}_{\mathrm{FV}} | \boldsymbol{V}_{\mathrm{LV}}, I_F, I_L\right) P\left(\boldsymbol{V}_{\mathrm{LV}} | I_F, I_L\right) P\left(I_F, I_L\right) \tag{A.4}$$

Since $\boldsymbol{V}_{\mathrm{LV}} = v_{L_{T:0}}$, the distribution $P\left(\boldsymbol{V}_{\mathrm{LV}}|I_F, I_L\right)$ could be expanded by Bayes' theorem as the equation A.5.

$$\begin{aligned} P\left(\boldsymbol{V}_{\mathrm{LV}}|I_F, I_L\right) &= P\left(v_{L_{T:0}}|I_F, I_L\right) \\ &= \prod_{t=T:\Delta t:-\Delta t} P\left(v_{L_t}|v_{L_{t+\Delta t:0}}, I_F, I_L\right) P\left(v_{L_0}|I_F, I_L\right) \\ &= \prod_{t=T:\Delta t:-\Delta t} P\left(v_{L_t}|v_{L_{t+\Delta t:0}}, I_F, I_L\right) \end{aligned} \tag{A.5}$$

where Δt is the time interval in the time-series $v_{L_{T:0}}$. Since v_{L_t} could be derived from $v_{L_{t-\Delta t}}$ and a_{L_t} (according to the motion equation $v_{L_t} = v_{L_{t-\Delta t}} - a_{L_t}\Delta t$), the distribution $P\left(v_{L_t}|v_{L_{t+\Delta t:0}}, I_F, I_L\right)$ could be derived from the distribution $P\left(a_{L_t}|v_{L_{t+\Delta t:0}}, I_F, I_L\right)$. As a result, the equation A.5 could be rewritten as equation A.6.

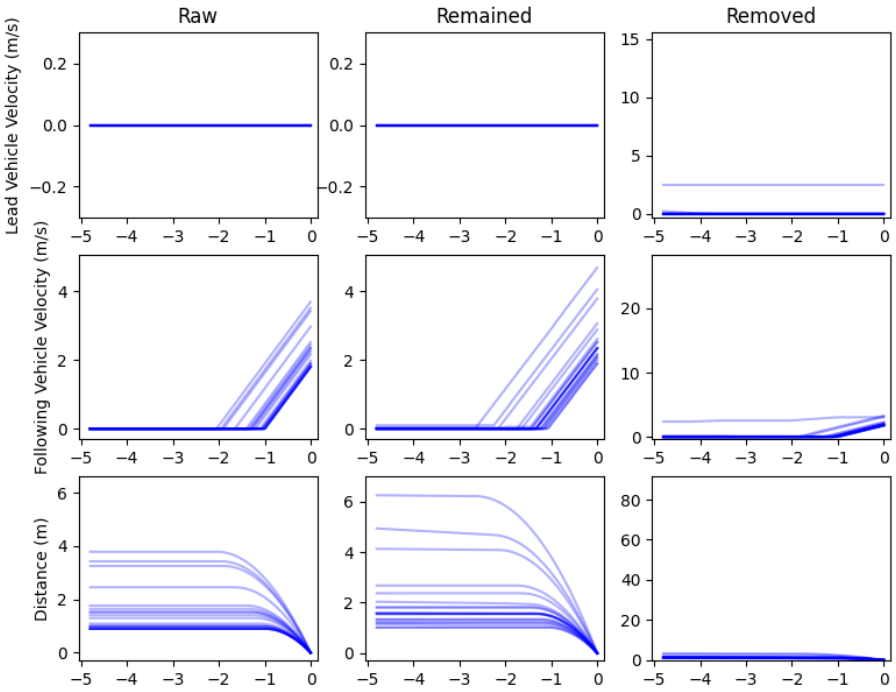
$$P\left(\boldsymbol{V}_{\mathrm{LV}}|I_F, I_L\right) \Leftrightarrow \prod_{t=T:\Delta t:-\Delta t} P\left(a_{L_t}|v_{L_{t+\Delta t:0}}, a_{L_{t+\Delta t:0}}, v_{F_0}, a_{F_0}\right) \tag{A.6}$$

Similar to the LV, the distribution $P\left(\boldsymbol{V}_{\mathrm{FV}}|\boldsymbol{V}_{\mathrm{LV}}, I_F, I_L\right)$ could be expanded as the equation A.7.

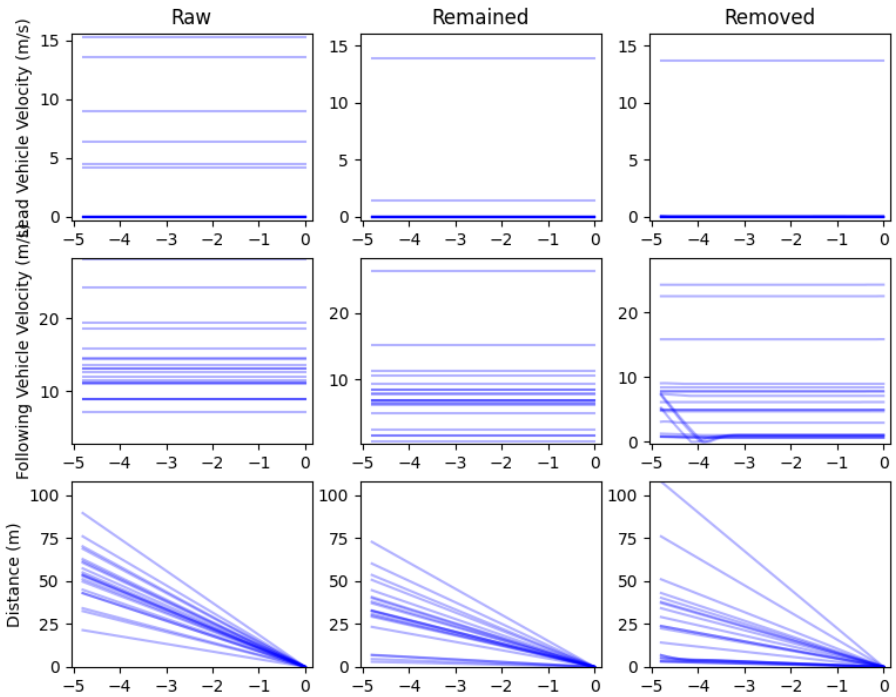
$$\begin{aligned} P\left(\boldsymbol{V}_{\mathrm{FV}}|v_{L_{T:0}}, I_F, I_L\right) &= P\left(v_{F_{T:0}}|v_{L_{T:0}}, I_F, I_L\right) \\ &= \prod_{t=T:\Delta t:-\Delta t} P\left(v_{F_t}|v_{F_{t+\Delta t:0}}, v_{L_{T:0}}, I_F, I_L\right) P\left(v_{F_0}|v_{L_{T:0}}, I_F, I_L\right) \\ &= \prod_{t=T:\Delta t:-\Delta t} P\left(v_{F_t}|v_{F_{t+\Delta t:0}}, v_{L_{T:0}}, I_F, I_L\right) \\ &\Leftrightarrow \prod_{t=T:\Delta t:-\Delta t} P\left(a_{F_t}|v_{F_{t+\Delta t:0}}, a_{F_{t+\Delta t:0}}, v_{L_{T:0}}, a_{L_{T:0}}\right) \end{aligned} \tag{A.7}$$

In conclusion, both Equation A.6 and Equation A.7 illustrate the LV and FV model behavior model in a mathematical representation respectively. The mathematical representations could reveal the essential input parameters for both models.
To be specific, $P\left(a_{L_t}|v_{L_{t+\Delta t:0}}, a_{L_{t+\Delta t:0}}, v_{F_0}, a_{F_0}\right)$ in Equation A.6 describes the distribution of LV behavior on time t given the LV kinematics during the time from the collision time ($t = 0$) to $t + \Delta t$ and the FV's initial condition including v_{F_0} and a_{F_0}. To model the distribution of a_{L_t} that the same as the training dataset, $v_{L_{t+\Delta t:0}}$, $a_{L_{t+\Delta t:0}}$, v_{F_0}, and a_{F_0} should be given as the input, theoretically.
$P\left(a_{F_t}|v_{F_{t+\Delta t:0}}, a_{F_{t+\Delta t:0}}, v_{L_{T:0}}, a_{L_{T:0}}\right)$ in Equation A.7 describes the distribution of FV behavior on time t given LV kinematic during the time from the collision time ($t = 0$) to $t + \Delta t$ and FV kinematic during the time from $t = 0$ to $t = T$. To model the distribution of a_{F_t} that the same as the training dataset, $v_{F_{t+\Delta t:0}}$, $a_{F_{t+\Delta t:0}}$, $v_{L_{T:0}}$, and $a_{L_{T:0}}$ should be given as the input, theoretically.

A.2 Cluster Post-processing

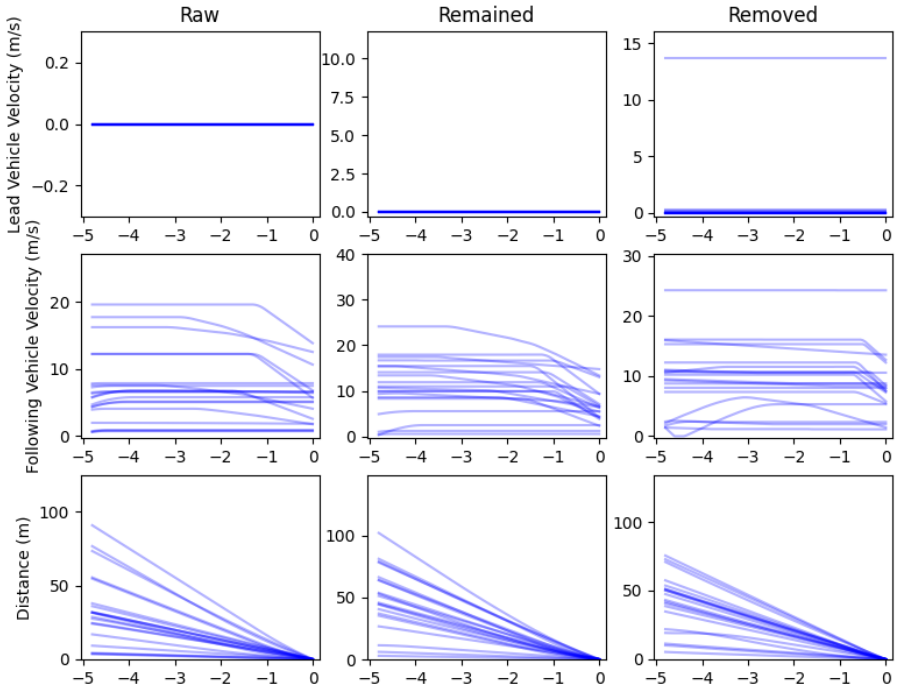


(a) Cluster 1

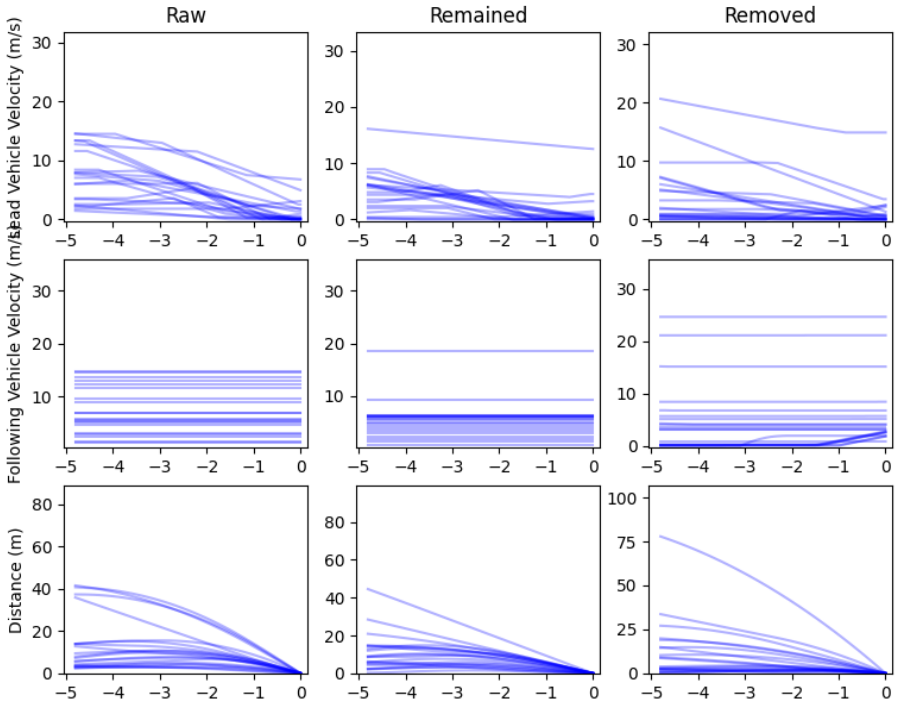


(a) Cluster 2

Figure A.1: Visual comparison between the raw case(s), the remaining generation, and the removed generation of Cluster 1 and Cluster 2.

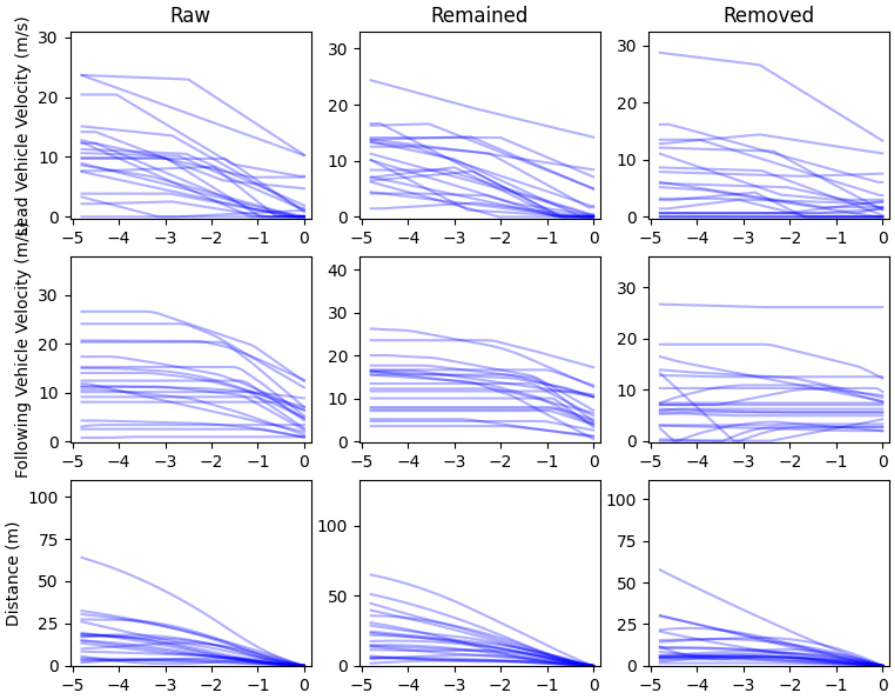


(a) Cluster 3

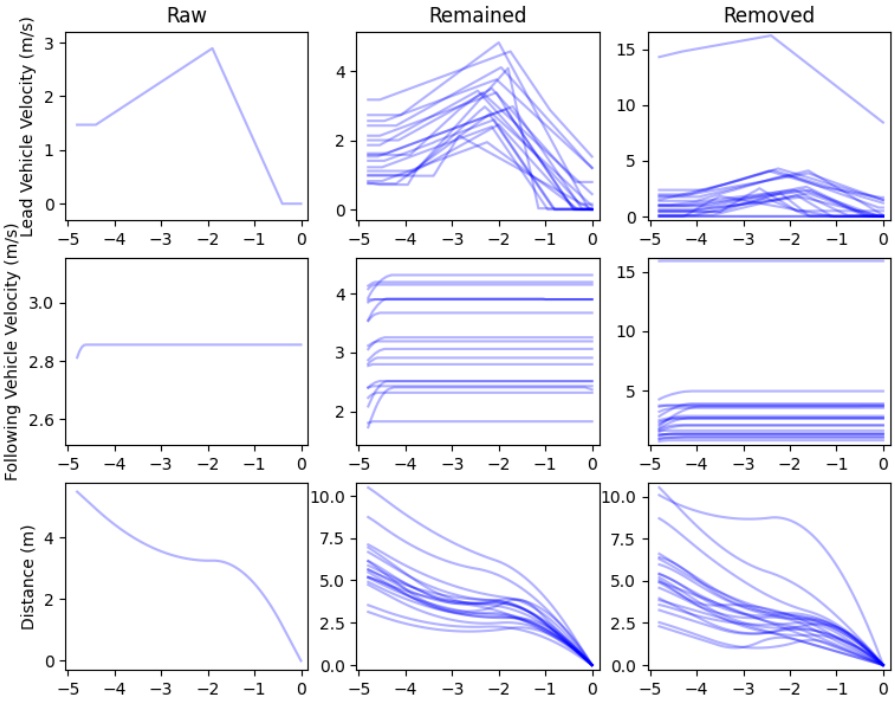


(b) Cluster 4

Figure A.2: Visual comparison between the raw case(s), the remaining generation, and the removed generation of Cluster 3 and Cluster 4.



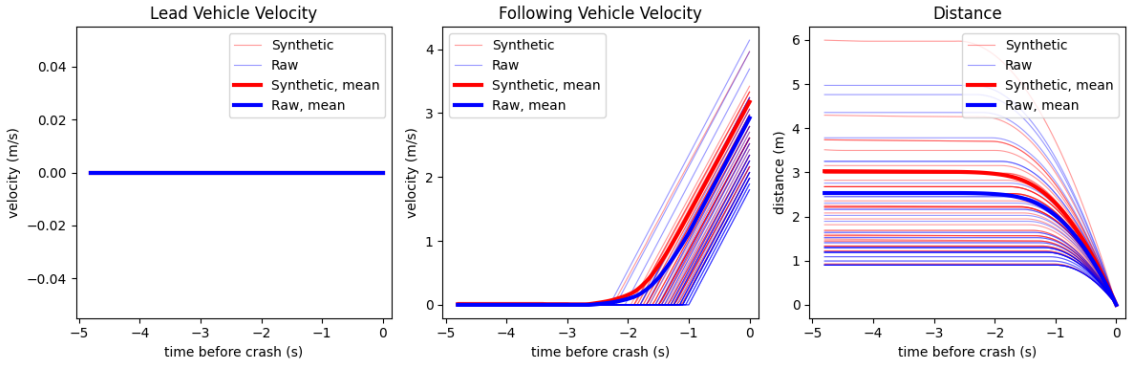
(a) Cluster 5



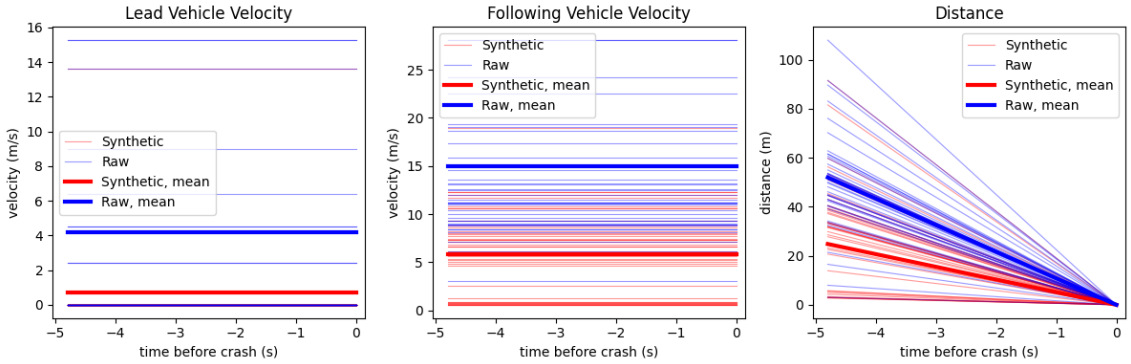
(b) Cluster 6

Figure A.3: Visual comparison between the raw case(s), the remaining generation, and the removed generation of Cluster 5 and 6.

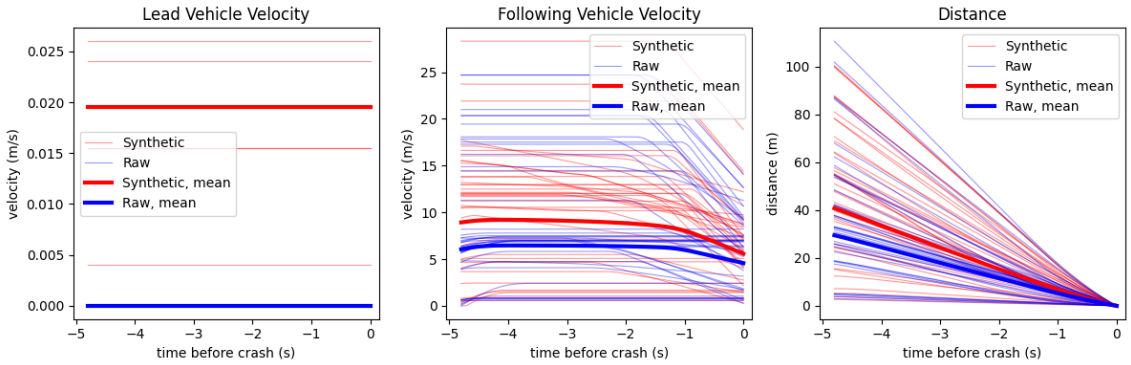
A.3 Synthetic Crash Kinematics Visualization



(a) Cluster 1

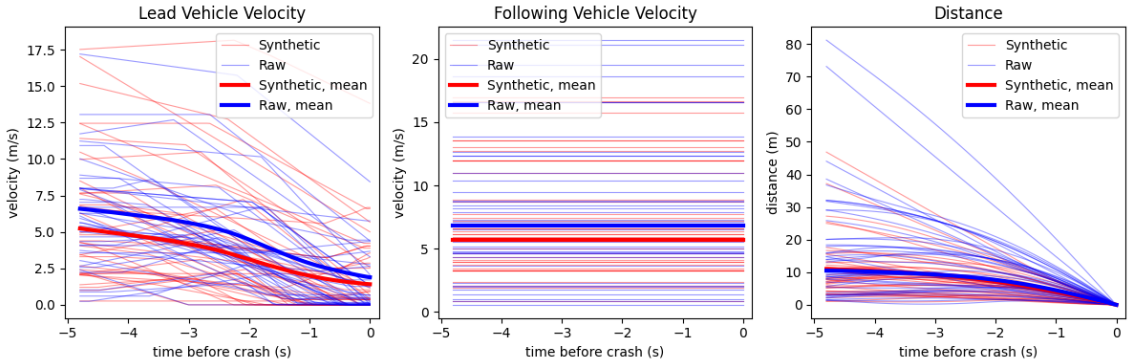


(b) Cluster 2

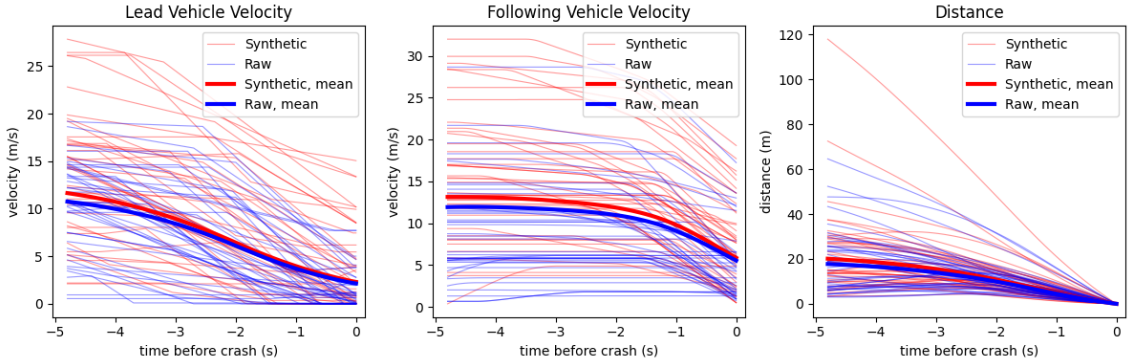


(c) Cluster 3

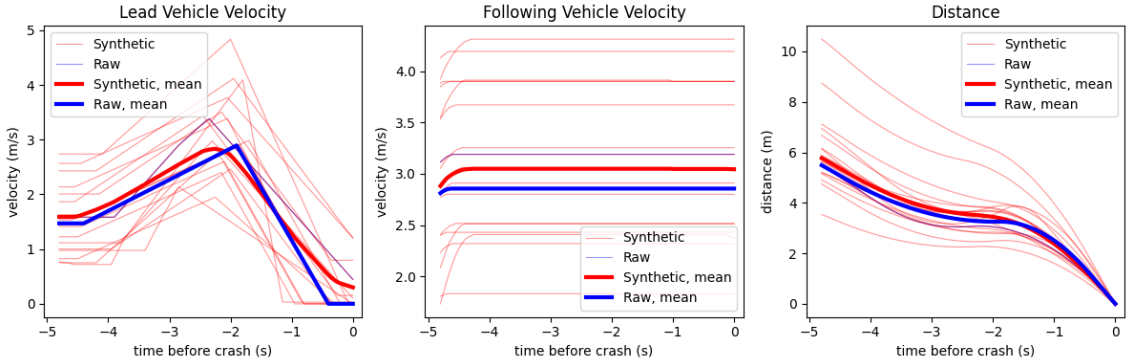
Figure A.4: Crash kinematics of both raw and synthetic crashes in Cluster 1, 2, 3 (20 randomly sampled for the raw and synthetic crashes respectively)



(a) Cluster 4



(b) Cluster 5



(c) Cluster 6

Figure A.5: Crash kinematics of both raw and synthetic crashes in Cluster 4, 5, 6 (20 randomly sampled for the raw and synthetic crashes respectively)

A.4 Probability Distributions (30% training dataset)

A.4.1 Kinematics at every timestamp

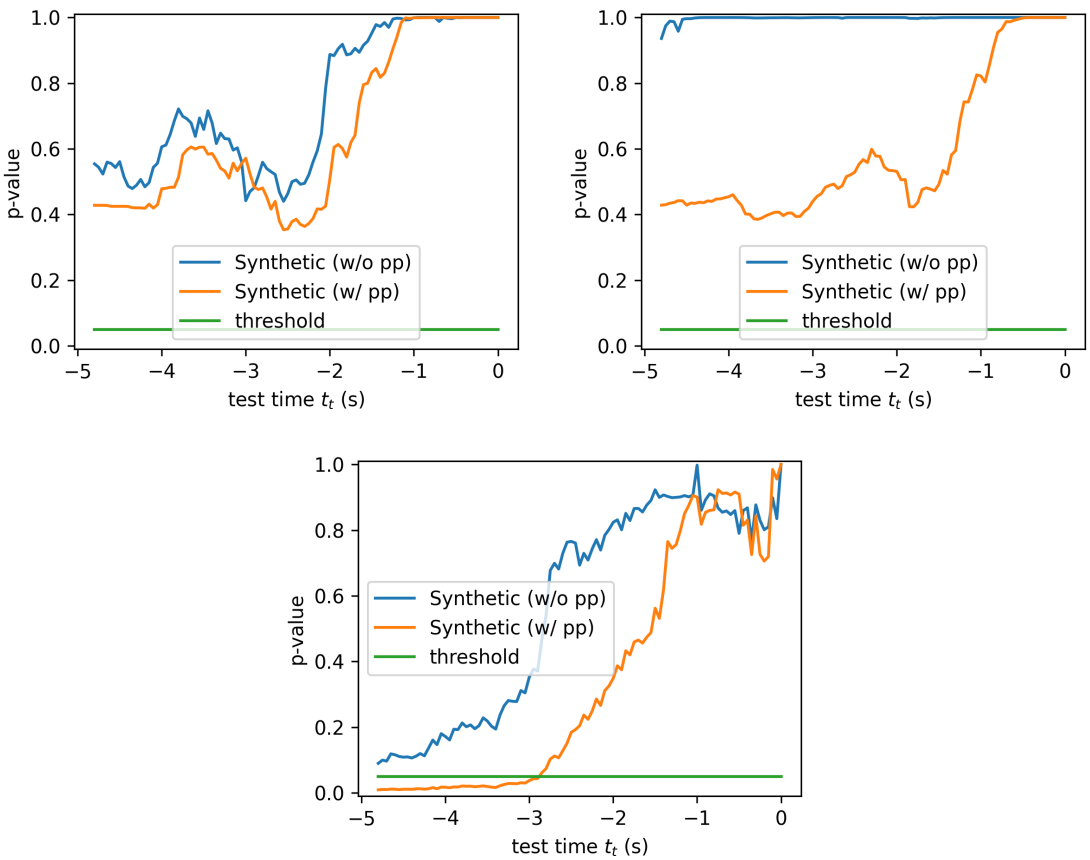


Figure A.6: KS-Test for the kinematics, including LV Velocity, FV Velocity, Distance between LV and FV, in 5s before collision happened

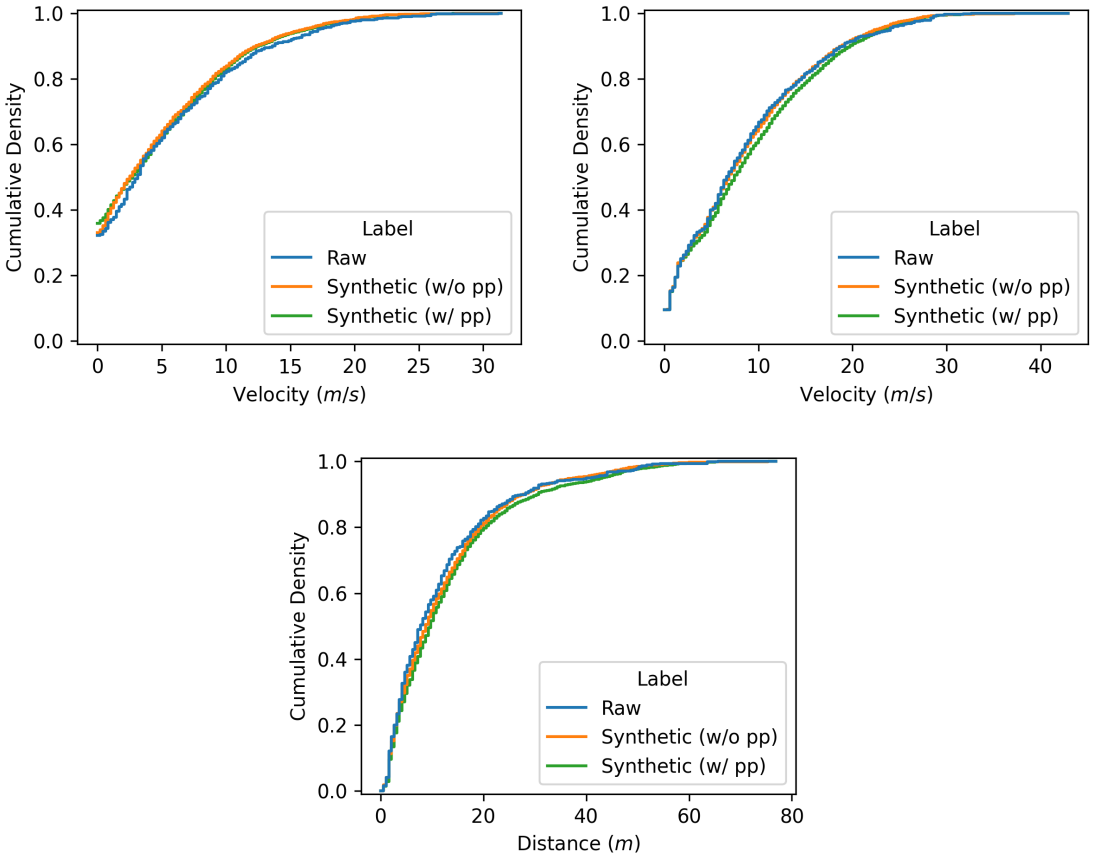


Figure A.7: Cumulative distribution of the kinematics, including LV Velocity, FV Velocity, Distance between LV and FV, on $t = -3s$ before the collision happened

A.4.2 Minimum acceleration

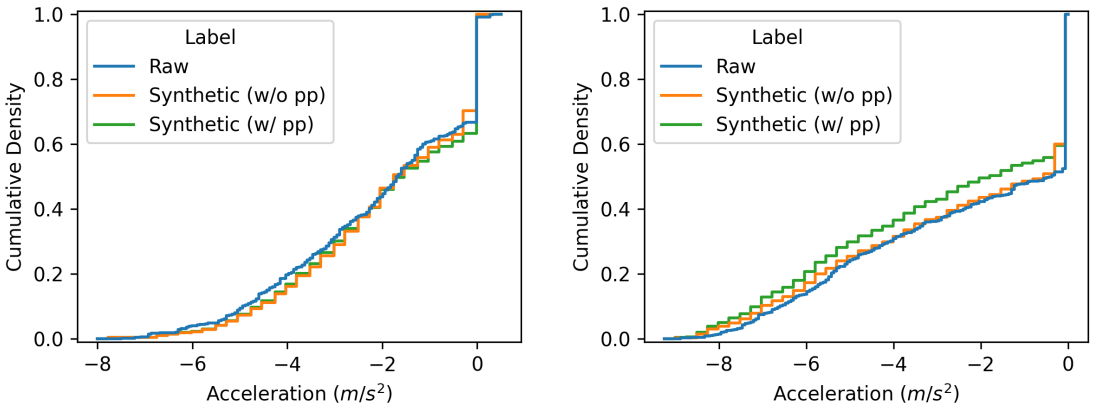


Figure A.8: Cumulative distribution of the minimum acceleration of LV and FV

Table A.1: P-value of KS-test for the cumulative distribution of the minimum acceleration of LV and FV

	Before Post-processing	After Post-processing
LV a_{min}	0.380	0.398
FV a_{min}	0.049	0.077

A.4.3 Delta-V

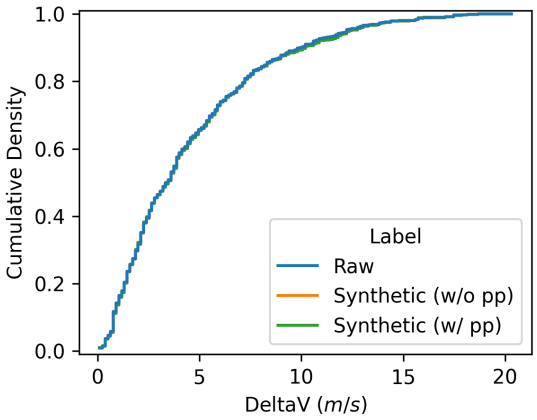


Figure A.9: Cumulative distribution of Delta-V of both Raw and Synthetic dataset

Table A.2: P-value of KS-test for the cumulative distribution of the Delta-V

	Before Post-processing	After Post-processing
Delta-V	1.000	1.000

A.4.4 Velocity Difference

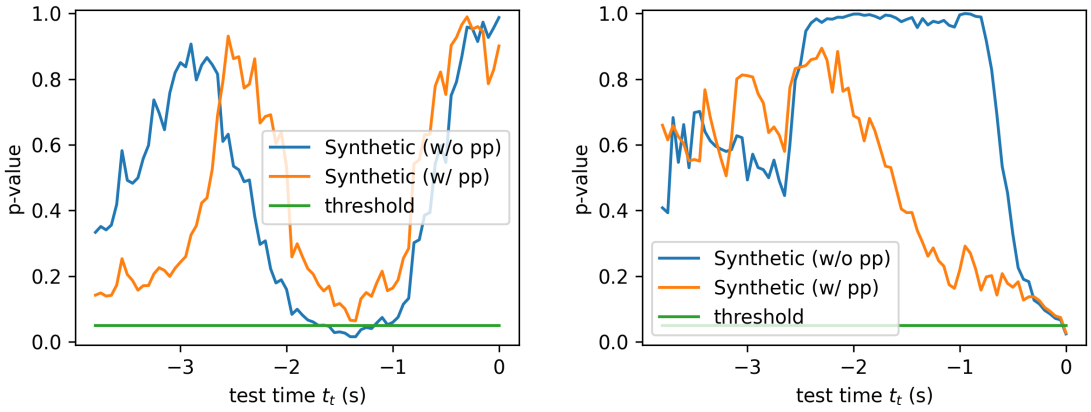


Figure A.10: KS-Test for the velocity difference between $1s$ in $5s$ before crash happened

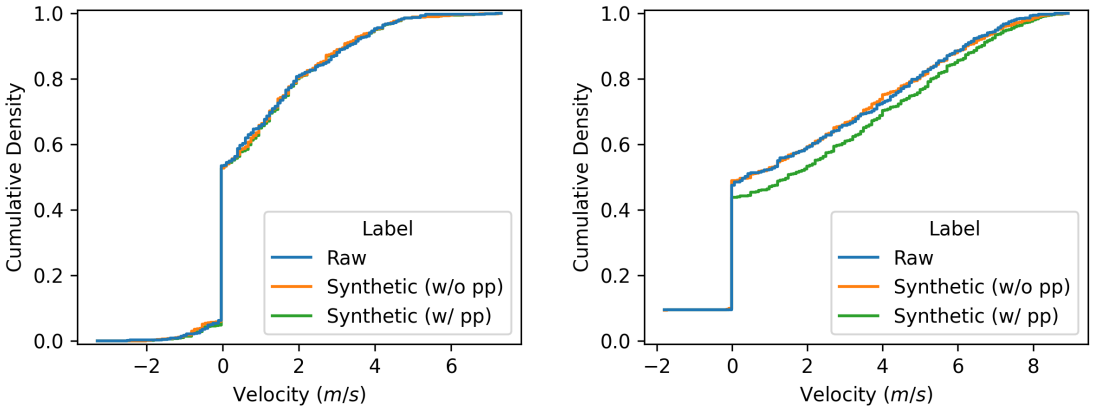


Figure A.11: Cumulative distribution of the velocity difference between $t = 0s$ and $t = -1s$

A.4.5 No-Retrun-time

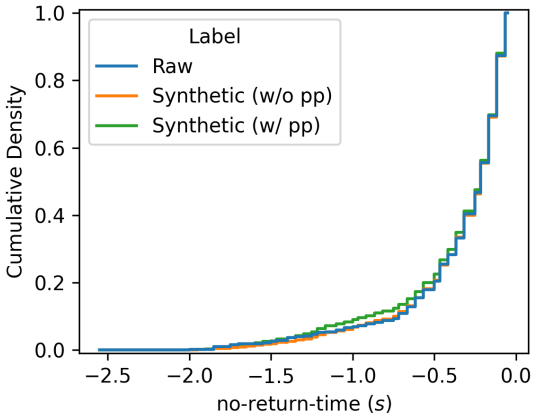


Figure A.12: Cumulative distribution of no-return-time of both Raw and Synthetic dataset

Table A.3: P-value of KS-test for the cumulative distribution of the no-return-time

	Before Post-processing	After Post-processing
no-return-time	1.000	0.980

A.5 Probability Distributions (20% training dataset)

A.5.1 Kinematics at every timestamp

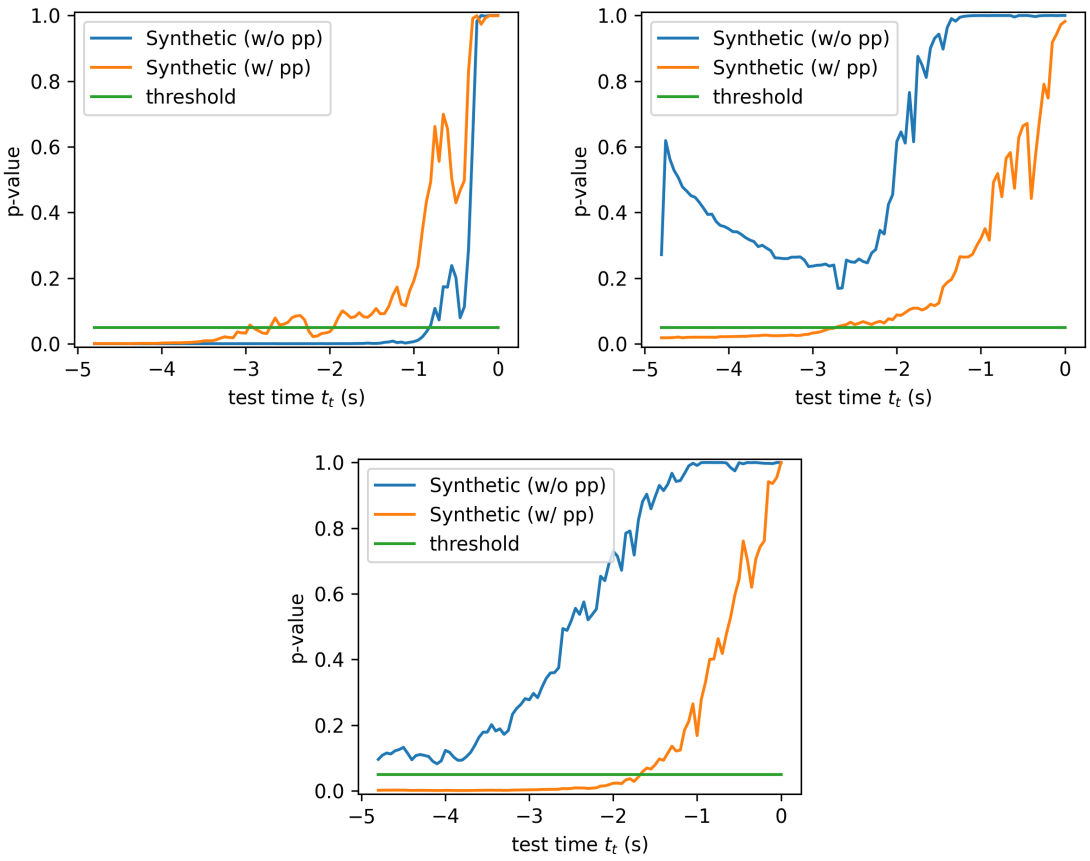


Figure A.13: KS-Test for the kinematics, including LV Velocity, FV Velocity, Distance between LV and FV, in 5s before collision happened

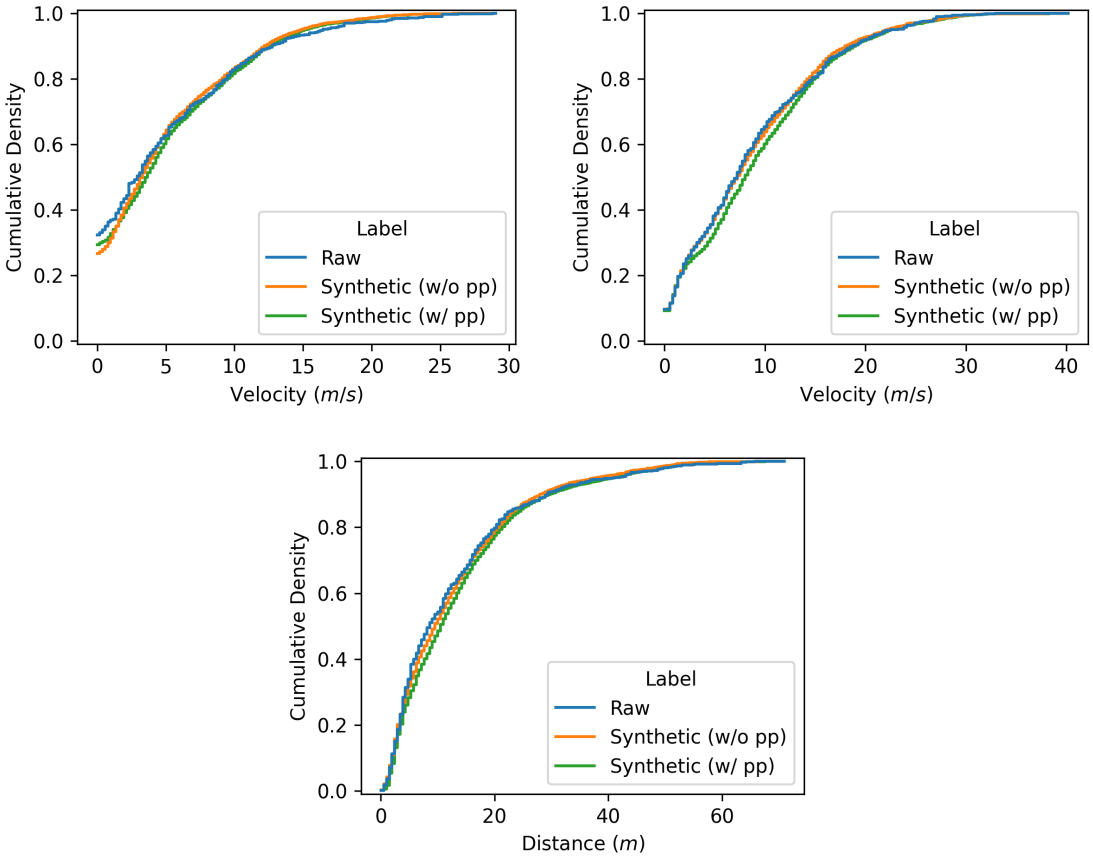


Figure A.14: Cumulative distribution of the kinematics, including LV Velocity, FV Velocity, Distance between LV and FV, on $t = -3s$ before the collision happened

A.5.2 Minimum acceleration

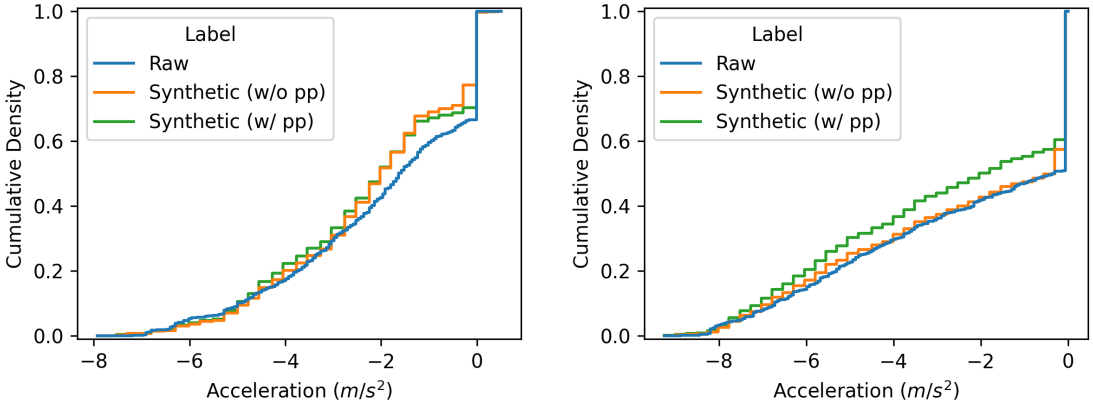


Figure A.15: Cumulative distribution of the minimum acceleration of LV and FV

Table A.4: P-value of KS-test for the cumulative distribution of the minimum acceleration of LV and FV

	Before Post-processing	After Post-processing
LV a_{min}	< 0.001	< 0.001
FV a_{min}	0.028	< 0.001

A.5.3 Delta-V

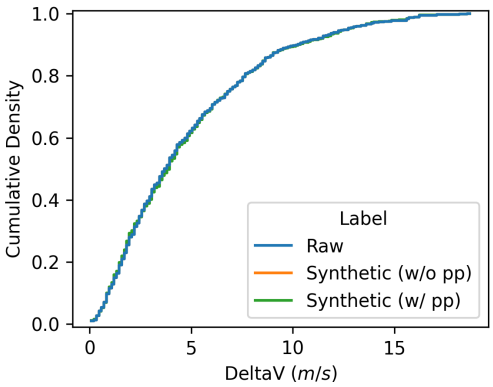


Figure A.16: Cumulative distribution of Delta-V of both Raw and Synthetic dataset

Table A.5: P-value of KS-test for the cumulative distribution of the Delta-V

	Before Post-processing	After Post-processing
Delta-V	1.000	0.999

A.5.4 Velocity Difference

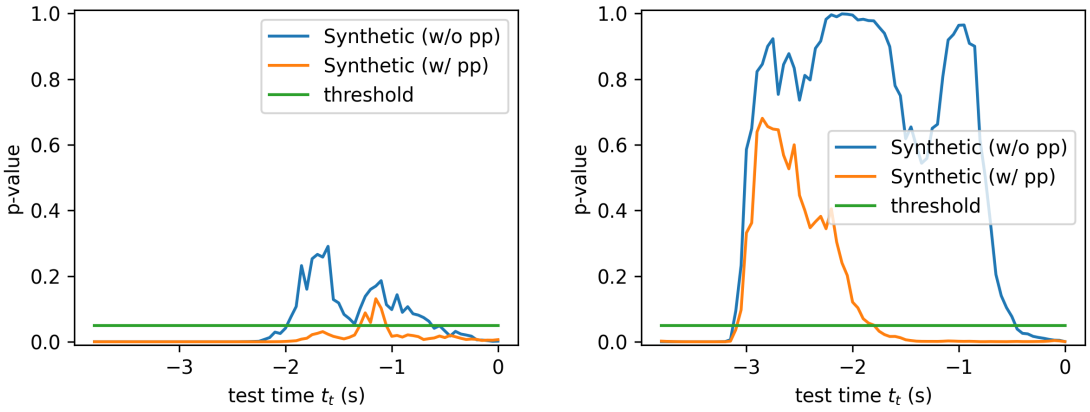


Figure A.17: KS-Test for the velocity difference between $1s$ in $5s$ before crash happened

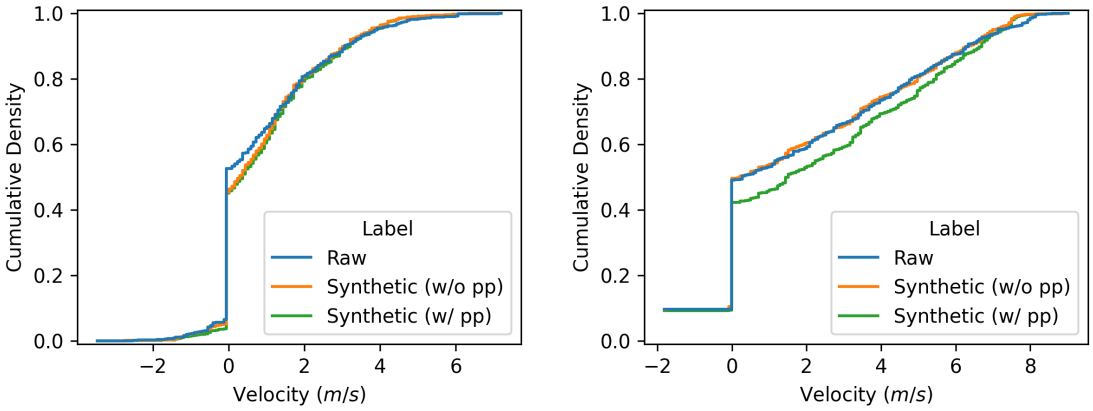


Figure A.18: Cumulative distribution of the velocity difference between $t = 0s$ and $t = -1s$

A.5.5 No-Retrun-time

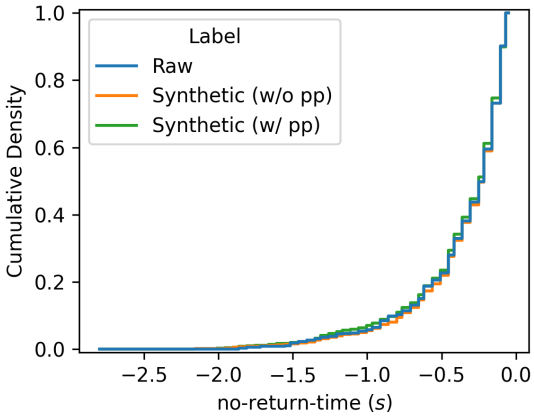


Figure A.19: Cumulative distribution of no-return-time of both Raw and Synthetic dataset

Table A.6: P-value of KS-test for the cumulative distribution of the no-return-time

	Before Post-processing	After Post-processing
no-return-time	0.998	0.999

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY