ChenSafe: Fault Injection and Corner Cases with CARLA Simulation for Autonomous Driving Vehicle

[1,*]*Hsiao-Ning Chen* (陳孝寧), [2,*]*Augustine Tsai* (蔡岳廷), [2]*Kuo-Hua Wu* (吳國華), [3]*Ting-Chi Chang* (張婷淇),[1]*Wei-Cheng You* (游惟丞),[1]*Zhi-Hong He* (何志宏), [1]*Chiou-Shann Fuh* (傅楸善),

[1]Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan,
[2]Institute for Information Industry, Taipei, Taiwan
[3]Graduate Institute of Biomedical Electronics and Bioinformatics, National Taiwan University
[*]E-mail: b03502032@ntu.edu.tw atsai@iii.rog.tw Khwu@iii.org.tw amy222024@gmail.com
wansars11@gmail.com fgghhk640@gmail.com fuh@csie.ntu.edu.tw

ABSTRACT

This paper presents a study on fault injection and corner cases in autonomous driving vehicles using CARLA simulation to improve safety. This research is motivated by the need to evaluate the robustness and safety of self-driving systems under various challenging scenarios. Building upon the work of "AVFI: Fault injection for autonomous vehicles", we extend the approach to incorporate CARLA simulation, a powerful open-source simulator for autonomous driving research [1]. The paper outlines the methodology, discusses the results obtained, and highlights the significance of this research in advancing the development of safe and reliable autonomous vehicles.

Keywords: *Fault Injection, Autonomous Driving Vehicle, Corner Cases, CARLA Simulation.*

1. INTRODUCTION

The development and validation of autonomous driving systems are crucial for ensuring their safety and reliability. However, testing these systems under dangerous or rare corner cases in real-world scenarios can be challenging and even impractical. To address this issue, simulation platforms such as CARLA have emerged as valuable tools for studying and evaluating autonomous driving behaviors.

This paper explores the use of CARLA, an open-source simulator specifically designed for autonomous driving research, to simulate dangerous corner cases in autonomous driving. By leveraging the flexibility and extensive digital assets provided by CARLA, the study aims to evaluate the failure model and mode of autonomous driving systems.

As autonomous driving becomes more widespread, ensuring tolerance and resilience in these systems is crucial for public acceptance and adoption. The study draws inspiration from the work on fault injection in autonomous vehicles and aims to contribute to the field of autonomous driving research by utilizing CARLA simulation to study dangerous corner cases, evaluating different autonomous driving methods, incorporating fault injection techniques, and proposing enhanced verification and validation approaches. The results of this study have the potential to enhance the safety, reliability, and public acceptance of autonomous driving systems in real-world scenarios.

2. RELATED WORK

2.1 Adversarial Attack and Fault Injection

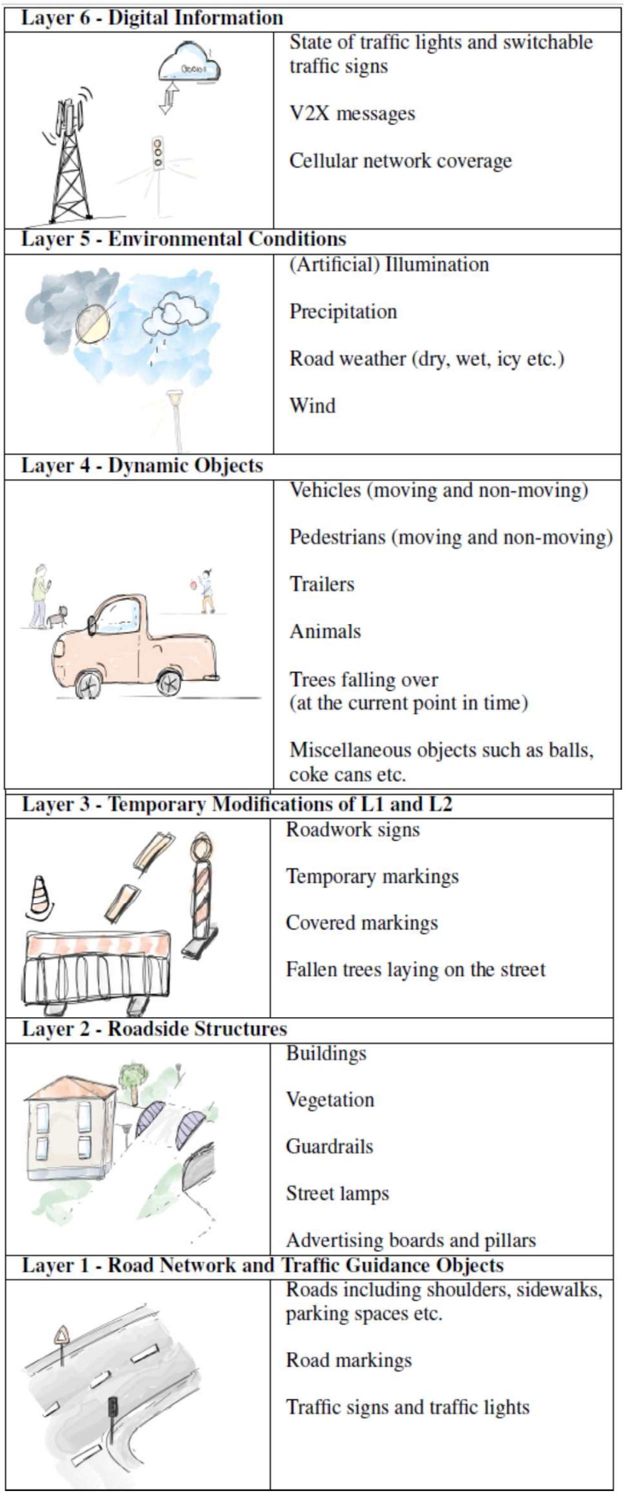
Wu et. al. [2] proposed an end-to-end autonomous driving fault injections based on applying perturbations on front looking camera. Perturbations are generated in two different approaches: image-specific for each frame and image-agnostic as a universal attack for all frames.

N. Piazzesi et al. introduced an example of attacking self-driving agent by the Adversarial Robustness Toolbox (ART, [3]) and PyTorch Fault Injection (PyTorchFI, [4]), two different approaches to inject perturbations into ML-based self-driving agent, this work presents an extensive experimental campaign that investigates the impact of injecting adversarial attacks and software faults into a self-driving agent operating within a driving simulator [5].

ART is a Python library initially created by IBM and, more recently, donated to the Linux Foundation. This comprehensive toolbox equips users with the necessary tools to generate adversarial attacks and develop robust defenses against them. ART enables straightforward invocation by providing the classifier and essential parameters, including loss and optimization functions, as well as image input size. Supporting well-known Machine Learning libraries such as PyTorch and TensorFlow, ART operates under the

MIT open-source license, facilitating its adoption and integration into diverse projects.

PyTorchFI, designed for the PyTorch deep learning platform, serves as a powerful runtime perturbation tool for Deep Neural Networks (DNNs). With its user-friendly API and extensible interface, PyTorchFI empowers users to apply perturbations on DNN weights or neurons during runtime.



Layer 6 - Digital Information	
	State of traffic lights and switchable traffic signs V2X messages Cellular network coverage
Layer 5 - Environmental Conditions	
	(Artificial) Illumination Precipitation Road weather (dry, wet, icy etc.) Wind
Layer 4 - Dynamic Objects	
	Vehicles (moving and non-moving) Pedestrians (moving and non-moving) Trailers Animals Trees falling over (at the current point in time) Miscellaneous objects such as balls, coke cans etc.
Layer 3 - Temporary Modifications of L1 and L2	
	Roadwork signs Temporary markings Covered markings Fallen trees laying on the street
Layer 2 - Roadside Structures	
	Buildings Vegetation Guardrails Street lamps Advertising boards and pillars
Layer 1 - Road Network and Traffic Guidance Objects	
	Roads including shoulders, sidewalks, parking spaces etc. Road markings Traffic signs and traffic lights

Fig. 1. Pegasus 6 layer framework [6].

Moreover, the paper advocates the importance of extensive testing campaigns to thoroughly evaluate the resilience and robustness of self-driving agents. This involves subjecting the agents to diverse scenarios that represent real-world challenges and potential vulnerabilities.

2.2 PEGASUS 6-layer model

In addition, Scholtes et al. [6], described a PEGASUS 6-layer model for driving environment, namely, road network, roadside structure, temporary modification, dynamic objects, environment conditions, and digital information layers, which makes significant contributions to the field of analyzing self-driving cases.

2.3 Corner Cases

Corner cases (CC) in automated driving refer to critical and infrequent scenarios that are vital for training, verifying, and enhancing the performance of machine learning (ML) models within autonomous vehicle systems. These rare and critical situations are often underrepresented in datasets but are crucial for reducing failures and unexpected behaviors during inference.

D. Bogdoll et al. introduced a novel approach for corner case description (CCD) that centers around the PEGASUS 6-layer model as a reference [7]. They comprehensively classified corner cases (CC) into three distinct layers: the sensor layer, content layer, and temporal layer. Furthermore, the researchers explored the application scenarios of CCD, providing valuable insights into effectively capturing and addressing critical situations in the context of autonomous driving systems.

3. METHODOLOGY

In this investigation, we adopt the PEGASUS model, with a specific emphasis on layer 4, particularly pertaining to pedestrian street crossing. The primary objective of this study is to assess the likelihood of collisions along a predetermined route, conducted over multiple iterations.

3.1 CARLA Simulator

We conducted the experiments using the CARLA simulator, an open-source platform designed for autonomous driving research. CARLA provides a high-fidelity and realistic environment that allows us to study the interactions between autonomous vehicles and pedestrians under various scenarios [8].

3.2 Traffic Manager in CARLA

The Traffic Manager agent is a crucial component responsible for regulating and simulating dynamic traffic scenarios within the CARLA environment. This intelligent agent efficiently controls the flow of vehicles, pedestrians, and other entities, ensuring a realistic and vibrant traffic system. By generating diverse traffic conditions, the Traffic Manager enables comprehensive testing and evaluation of autonomous driving algorithms under various challenging situations.

3.3 Route Planner

The Route Planner serves as a strategic agent responsible for charting optimal paths and planning efficient routes for autonomous vehicles. By analyzing the simulation environment and considering factors such as traffic density and road conditions, the Route Planner generates safe and effective routes for the Autopilot agent to follow. This critical component enables intelligent route planning, making autonomous driving in CARLA more reliable and adaptive to real-world scenarios.

3.4 Autopilot Agent

The Autopilot agent is the core of the autonomous driving system in CARLA. Implemented with advanced control algorithms, this agent allows vehicles to navigate through the simulated world autonomously. By interpreting sensor data and making informed decisions, the Autopilot agent effectively steers the vehicle along predetermined routes, adhering to traffic rules and responding to dynamic obstacles. Its integration with the Traffic Manager and Route Planner facilitates the seamless and efficient execution of autonomous driving tasks.

3.5 Example Codes

For this experiment, we utilized the example codes "generate_traffic.py" and "automatic_control.py" that come bundled with the CARLA simulator. These scripts allowed us to generate dynamic traffic and control the autonomous vehicle's behavior throughout the simulation.

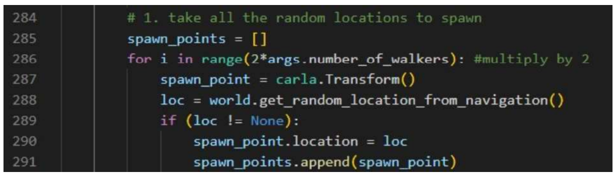
4. EXPERIMENT

4.1 Traffic Generation

The "generate_traffic.py" script was used to create dynamic traffic scenarios, including various vehicles and pedestrians, in the CARLA simulation. The traffic density was set to a level that reflects real-world urban traffic conditions.

When the script tries to spawn an actor (car or walker) at an occupied location, the default action is to skip the spawn, which might cause the inconsistency of the testing conditions, such as less pedestrian than expected on the road.

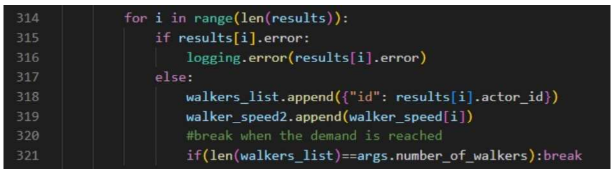
To avoid the potential risk, the script is modified in the spawn actor section. First, the number of spawn points generated is multiplied by *2* to make sure it will exceed the demand amount of actor.



```
        # 1. take all the random locations to spawn
        spawn_points = []
        for i in range(2*args.number_of_walkers): #multiply by 2
            spawn_point = carla.Transform()
            loc = world.get_random_location_from_navigation()
            if (loc != None):
                spawn_point.location = loc
                spawn_points.append(spawn_point)
```

Fig. 2. Multiply spawn point generation by 2.

Secondly, add an if function to check the spawned amount after each actor is added into the world, break the spawning loop once the demand number is reached.



```
        for i in range(len(results)):
            if results[i].error:
                logging.error(results[i].error)
            else:
                walkers_list.append({"id": results[i].actor_id})
                walker_speed2.append(walker_speed[i])
                #break when the demand is reached
                if(len(walkers_list)==args.number_of_walkers):break
```

Fig. 3. *if* function to check the spawned number.

4.2 Automatic Control

In every simulation, the "automatic_control.py" script was employed to control the autonomous vehicle's behavior along the predefined route. The vehicle followed a specified speed profile, obeyed traffic rules, and responded to dynamic changes in the environment.

To decrease the randomness of the simulation parameters, several modifications were made to the sample code, it is achieved by assigning the default value of the car model to “Mercedes Coupe 2020”, and the vehicle target speed to 50 km/h in the script.

4.3 Route Selection

To ensure consistency in the experiment, we designed a fixed given route that would be followed by the autonomous vehicle during each loop of the simulation. The selected routes for all 4 maps are illustrated in Figures 4 to 12, including urban and suburban areas with intersections, crosswalks, and pedestrian-heavy zones.

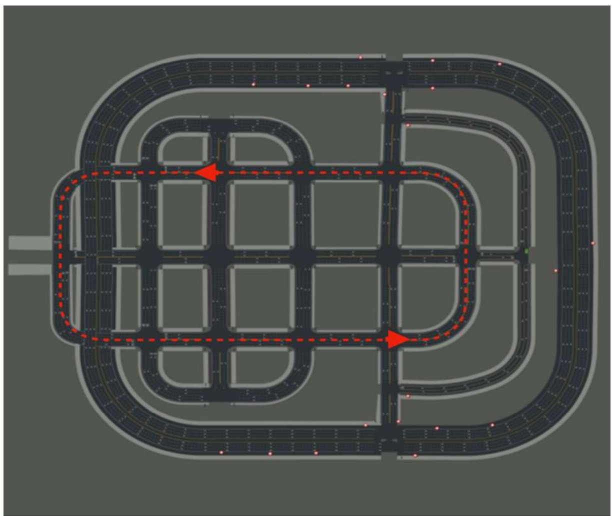


Fig. 4. Overview of selected Route in Town05 Map.

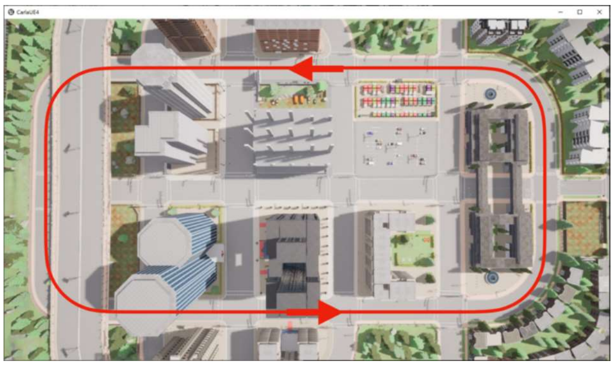


Fig. 5. Real scene of selected Route in Town05 Map.

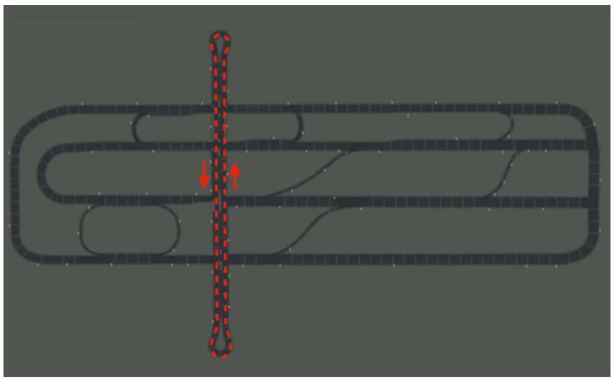


Fig. 6. Overview of selected Route in Town06 Map.

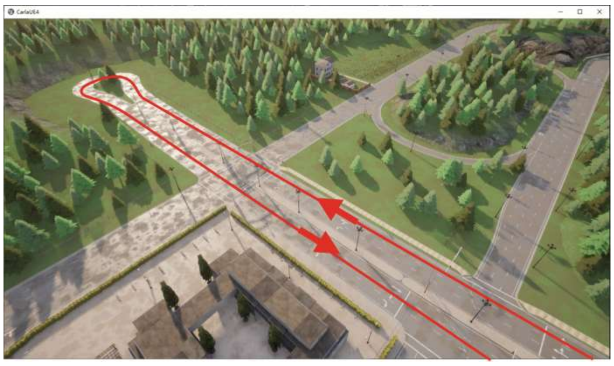


Fig. 7. Real scene of selected Route in Town06 Map (South).

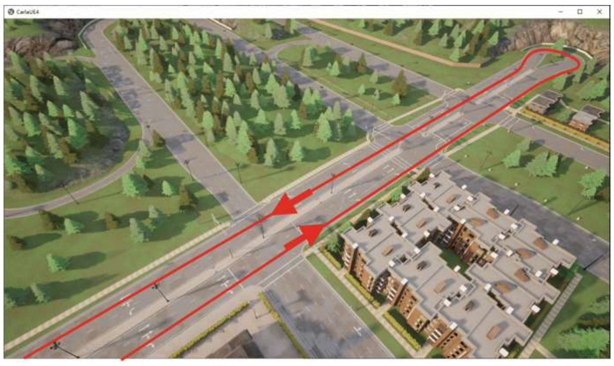


Fig. 8. Real scene of selected Route in Town06 Map (North).

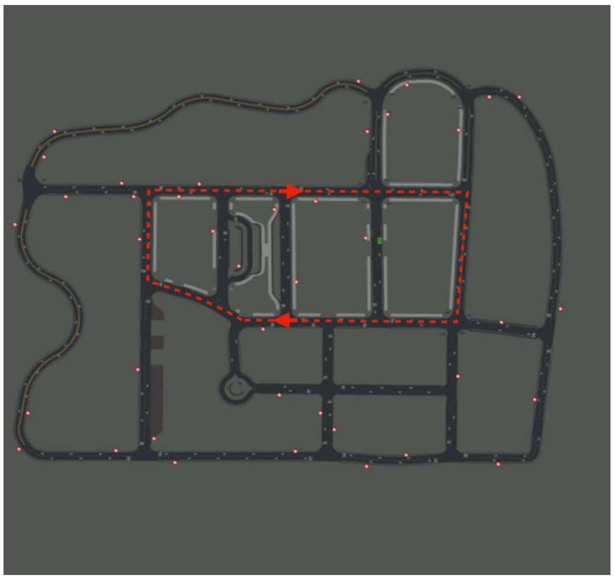


Fig. 9. Overview of selected Route in Town07 Map.

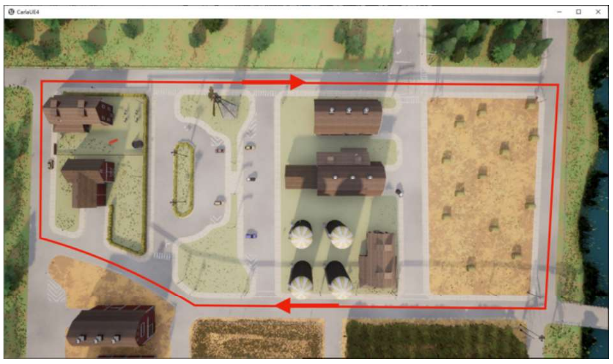


Fig. 10. Real scene of selected Route in Town07 Map.

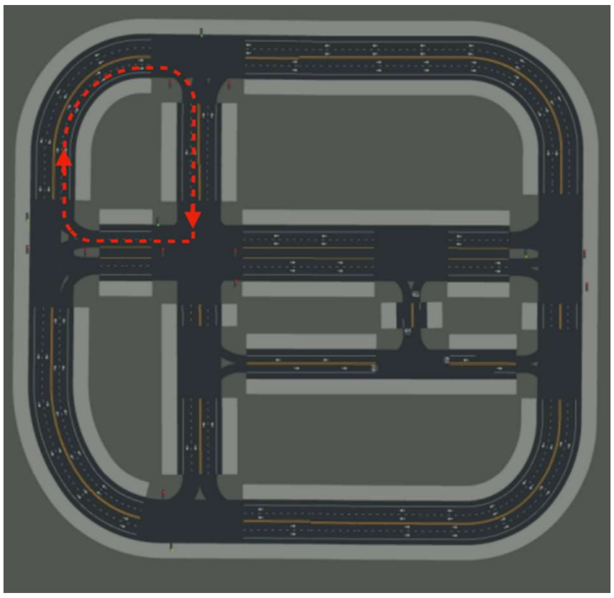


Fig. 11. Overview of selected Route in Town10 Map.

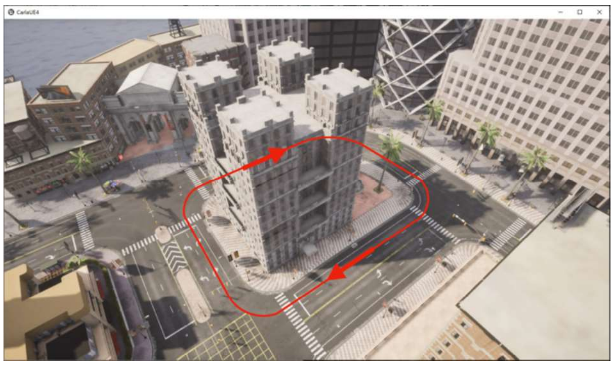


Fig. 12. Real scene of selected Route in Town10 Map.

4.4 Repetition

The experiment was repeated for ten loops to obtain a statistically significant number of collision events and calculate the probability of collisions with pedestrians during the specified route.

4.5 Collision Recording

Throughout each loop, the CARLA simulator continuously monitored the interactions between the autonomous vehicle and pedestrians. If a collision occurred, the simulator recorded the incident and the corresponding data.

The recorded collision data, along with relevant simulation parameters, were collected for each loop. The data included the time of the collision, vehicle, pedestrian ID, and other relevant information, Figures 13 and 14 demonstrate a collision between a vehicle and a pedestrian.

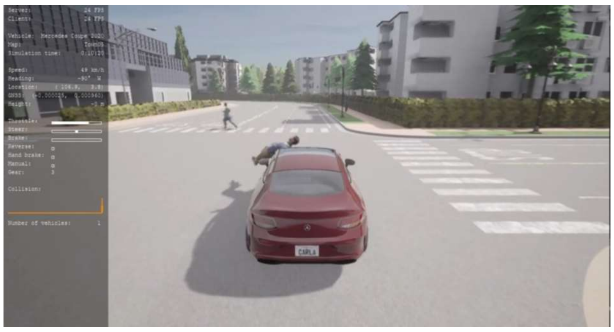


Fig. 13. First frame of collision occurrence.

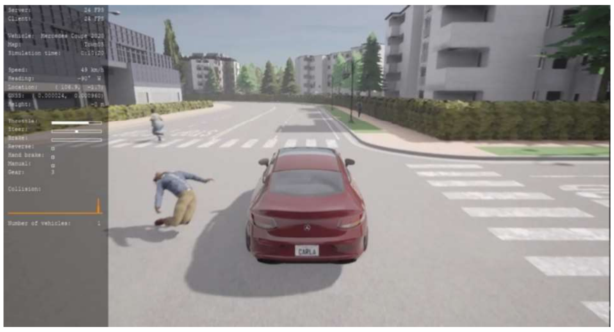


Fig. 14. Second frame of collision occurrence.

4.6 Control Variables

Most of the control variables are assigned in the scripts “generate_traffic.py” and “automatic_control.py”, only the map and weather are set in the configuration of simulation server.

Table 1. Control Variables.

Variable Name	Values
Probability of Pedestrian Crossing Road (C.R.%)	30%, 50%
Number of Pedestrians (NoP)	50, 100, 150
Weather	ClearNoon, MidRainyNoon
Vehicle Speed	50 km/h
Map	Town05, Town06, Town07, Town10

5. RESULTS

Table 2. Simulation Results.

NoP	50		100		150		150	
Weather	ClearNoon						MidRainy Noon	
C.R.%	30	50	30	50	30	50	30	50
Town05	2	2	4	5	8	10	9	10
Town06	5	4	4	4	7	10	8	9
Town07	3	5	6	10	8	11	7	11
Town10	2	5	4	8	10	12	9	10
Total	12	16	18	27	33	43	33	40

Table 2 shows the number of collisions recorded during the simulation, by comparing the number of collisions on each map with fixed C.R.% and NoP, Figures 15 and 16 show that map differences have no significant effect on the collision probability.

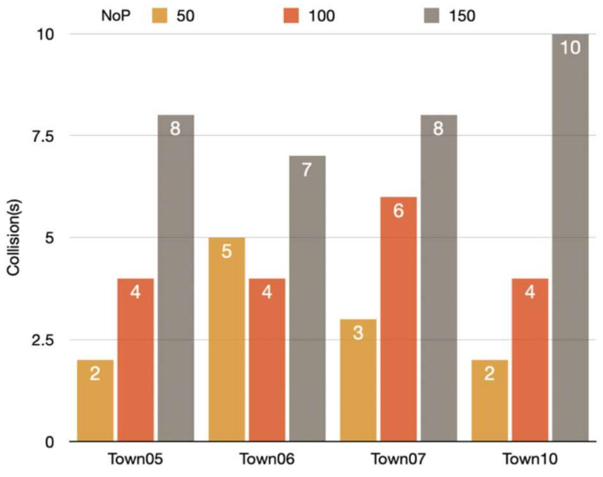


Fig. 15. Bar chart with 30% C.R.%.

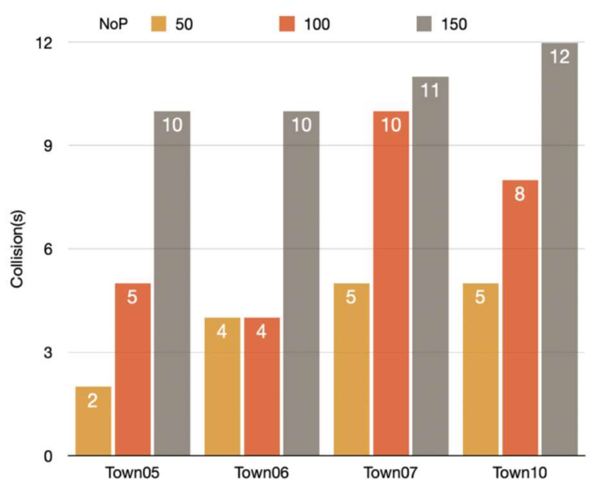


Fig. 16. Bar chart with 50% C.R.%.

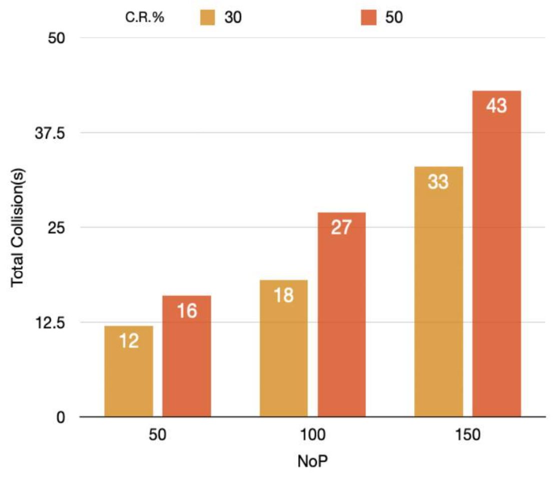


Fig. 17. Bar chart of total collisions with different NoP and crossing rate.

In Figure 17, a clear and discernible trend emerges as the NoP and C.R.% increase. Notably, the analysis reveals that changes in NoP have a slightly more pronounced effect on collision probability compared to alterations in the probability of pedestrian road crossings.

As the NoP rises, the collision rate exhibits a notable upward trend, indicating that higher pedestrian densities contribute to an elevated likelihood of collisions between autonomous vehicles and pedestrians. This finding underscores the significance of accounting for pedestrian traffic density in developing robust autonomous driving systems that can safely navigate through dense urban environments with increased pedestrian activity.

Moreover, the observed trend highlights the complexity of interactions between autonomous vehicles and pedestrians in highly populated areas. It suggests that the collision probability is influenced not only by the frequency of pedestrians crossing the road but also by the sheer volume of pedestrians present. The interplay of these factors can lead to intricate and dynamic scenarios, necessitating advanced collision avoidance strategies and attentive pedestrian detection mechanisms to mitigate safety risks effectively.

Lastly, an observation arises from our experiments conducted under the MidRainyNoon weather condition in the CARLA simulation world. Surprisingly, we found no significant difference in collision probability when compared to the ClearNoon weather, which serves as the primary scenario in our study, as demonstrated in Figure 18.

This unexpected result (maybe caused by route planner ignoring weather condition data) warrants further examination and analysis, it is essential to conduct detailed investigations to validate the robustness of these findings in real-world driving scenarios. Physical testing in various rain intensities and visibility conditions will provide additional insights into the system's performance and safety measures during adverse weather conditions.

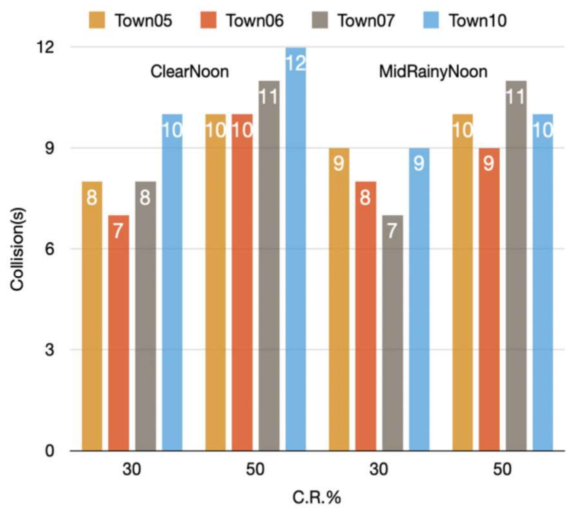


Fig. 18. Bar chart under different weather conditions with NoP=150.

6. CONCLUSION

We investigated corner cases, mainly on pedestrian-heavy environments of autonomous driving with a focus on collision probability and safety assessment in complex traffic scenarios.

The analysis of collision probabilities concerning the number of pedestrians and the probability of pedestrian road crossings revealed a significant relationship between these variables and collision outcomes. The observed trend of an increased collision probability with higher NoP emphasizes the importance of accounting for pedestrian density in designing robust autonomous driving systems. This finding highlights the necessity of developing advanced pedestrian detection algorithms and effective collision avoidance strategies to ensure safe interactions with pedestrians in densely populated areas.

Overall, the experiment contributes to the understanding of collision probability and safety considerations in autonomous driving. The insights gained from this research have implications for the development of enhanced autonomous driving algorithms, adaptive navigation strategies, and weather-aware systems. By addressing the challenges posed by pedestrian traffic and adverse weather conditions, we move closer to the vision of safe and reliable autonomous vehicles in our future transportation landscape.

As we continue to advance autonomous driving technologies, it is essential to emphasize real-world validation and the integration of human-machine interaction to instill trust and transparency in these systems. Furthermore, future research should explore the robustness and generalizability of the findings to ensure safe and efficient autonomous driving experiences across diverse and dynamic traffic scenarios.

In conclusion, this thesis serves as a steppingstone towards safer and more efficient autonomous driving systems, and it contributes to the broader effort of making autonomous vehicles an integral part of our transportation ecosystem. With ongoing advancements and continuous research, we aim to propel the autonomous driving revolution, making transportation safer, sustainable, and accessible for all.

7. FUTURE WORK

By introducing future research directions, we demonstrate an understanding of the broader challenges and opportunities in the field of autonomous driving, contributing to the ongoing efforts to enhance safety, reliability, and public acceptance of autonomous vehicles.

7.1 Improved Collision Avoidance Strategies

Investigate and develop more advanced collision avoidance strategies that leverage real-time sensor data and predictive analytics to proactively mitigate potential collisions with pedestrians and other road users. Implement and evaluate these strategies in diverse scenarios to enhance the overall safety of autonomous driving systems.

7.2 Implement Different Self-Driving Agent

In the pursuit of advancing autonomous driving technology, a crucial avenue for future research lies in the implementation and evaluation of diverse self-driving agents. By exploring a range of autonomous driving algorithms, including rule-based systems, reinforcement learning, and deep neural networks, such as Autoware [9] or Apollo from Baidu [10], these agents are provided as ROS [11] package, both agents can provide complete set of self-driving modules, including detection, localization, prediction, planning, and control. The responses to the corner cases can be validated in CARLA simulated environment with vehicle dynamics context [12].

7.3 Multi-Agent Interaction

Investigate the interactions between autonomous vehicles and human-driven vehicles, as well as the interactions between multiple autonomous vehicles on the road. Address challenges related to cooperation, coordination, and communication to promote safe and efficient traffic flow.

7.4 Adversarial Robustness Enhancement

Expand the research on adversarial attacks and explore robustness-enhancing techniques. Investigate methods like adversarial training, defensive distillation, and robust optimization to fortify the autonomous driving system against adversarial manipulations and improve its resistance to potential cyber threats.

7.5 Fault Injection

Fault injection involves deliberately introducing errors or faults into the system to assess its robustness under adverse conditions. By systematically simulating and evaluating various fault scenarios, we can identify vulnerabilities in the self-driving agent and develop targeted solutions to mitigate potential failures. Fault injection testing will not only contribute to refining the system's fault tolerance and safety mechanisms but also provide critical insights into the system's behavior and decision-making during unpredictable and challenging situations on the road.

7.6 Ethical Decision-Making

Delve into the ethical implications of autonomous vehicle decision-making, particularly in life-threatening scenarios. Develop ethical frameworks that prioritize human safety while considering the interests of different road users and ensuring transparency in the decision-making process.

7.7 Autoware SiL Validation

Autoware is a software stack includes modules for localization, detection, prediction, planning, and control. CARLA can be used to generate synthetic corner case data for SiL (Software in the Loop) validation.

7.8 Real-World Validation

Conduct extensive real-world validation and testing of the autonomous driving system in a controlled and safe environment. Collaborate with regulatory bodies and stakeholders to obtain permission for on-road testing, ensuring that the system adheres to safety standards and regulations.

8. REFERENCE

[1] S. Jha, S. S. Banerjee, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer, "AVFI: Fault Injection for Autonomous Vehicles," *Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks Workshops*, Luxembourg City, Luxembourg, pp. 1-2, 2018.

[2] H. Wu and W. Ruan, "Adversarial Driving: Attacking End-to-End Autonomous Driving Systems," *Proceedings of IEEE Intelligent Vehicle Symposium*, Nagoya, Japan, pp. 266-273, 2023.

[3] M.I. Nicolae, M. Sinn, M.N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, Ian M. Molloy, B. Edwards, "Adversarial Robustness Toolbox v1.0.0," *arXiv preprint* arXiv:1807.01069v4, 2019.

[4] A. Mahmoud, N. Aggarwal, A. Nobbe, J. R. S. Vicarte, S. V. Adve, C. W. Fletcher, I. Frosio, S. K. S. Hari, "PyTorchFI: A Runtime Perturbation Tool for DNNs," *Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks Workshops*, Valencia, Spain, pp. 25-31, 2020.

[5] N. Piazzesi, M. Hong, and A. Ceccarelli. "Attacks and Faults Injection in Self-Driving Agents on the CARLA Simulator – Experience Report," https://arxiv.org/abs/2202.12991, 2022.

[6] M. E. Scholtes, "6-Layer Model for a Structured Description and Categorization of Urban Traffic and Environment," *IEEE Access*, Vol. 9, pp. 59131-59147, 2021.

[7] D. Bogdoll, J. Breitenstein, F. Heidecker, M. Bieshaar, B. Sick, T. Fingscheidg, and J. M. Zollner, "Description of Corner Cases in Automated Driving: Goals and Challenges," *Proceedings of IEEE/CVF International Conference on Computer Vision Workshops,* arXiv:2109.09607, pp. 1-6, 2021.

[8] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," *Proceedings of Annual Conference on Robot Learning*, arXiv:1711.03938, 2017.

[9] Autoware Foundation, "Autoware" *http://autoware.org, 2023.*

[10] M. Feng and H. Zhang, "Application of Baidu Apollo Open Platform in a Course of Control Simulation Experiments," *Computer Applications in Engineering Education,* https://onlinelibrary.wiley.com/doi/full/10.1002/cae.22492, 2022.

[11] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, Architecture, and Uses in the Wild," *10.48550/arXiv.2211.07752*, 2022.

[12] R. Rajamani, *Vehicle Dynamics and Control*, 2nd. Edition, Springer, Berlin, 2012.