

Cost-based Non-deep Learning for Planning and Decision-making Towards Enhancing Autonomous Driving's Safety and Comfort

Shihao Gan¹, Siyao Wu², Yang Luo³[0000-0002-7478-1159], and
Fan Zhang¹[0000-0001-6228-940X]

¹ School of Robotics, Xi'an Jiaotong-Liverpool University, Suzhou, China

² China Mobile Group Design Institute Co. Ltd., Beijing, China

³ School of Intelligent Manufacturing Ecosystem, Xi'an Jiaotong-Liverpool
University, Suzhou, China
{fan.zhang}@xjtlu.edu.cn

Abstract. In pursuit of enhancing driving safety while controlling and reducing vehicle and testing costs, our research explores advanced decision logic for autonomous driving. This study integrates foundational algorithms and employs a suite of tools, including MATLAB, Simulink, Prescan, and CarSim, to develop a comprehensive virtual simulation testing environment. Within this virtual platform, we create and test various environmental parameters and vehicle states to simulate real-world driving conditions accurately. Our core cost-based decision-making algorithm processes perceptual results to calculate the "cost," which then informs driving decisions and planning. By default, the planning tried not to decelerate, but maintained acceleration changes as smooth as possible, which leading to unsafe scenarios such as driving off-road to go around the obstacles. In contrast, we proposed an explainable algorithm that could effectively balances safety and comfort, ensuring safer driving outcomes. Furthermore, our work could contribute to the field of Explainable Artificial Intelligence in Autonomous Driving by providing a robust framework for testing and optimizing decision logic, ultimately leading to the development of safe and efficient autonomous vehicles.

Keywords: Autonomous Driving · Explainable Artificial Intelligence · Dynamic Planning · Decision-making · Safety-comfort Trade-off

1 Introduction

With the continued advancement of Artificial Intelligence technology, significant achievements have been made in the field of autonomous driving [1], especially in the areas of system perception [2], decision planning [3], and vehicle control [4]. Current technology applications focus on advanced perception systems, deep learning algorithms for complex scene processing, sensor accuracy, and decision-making and control systems that mimic human driving behavior (e.g. mimic human attention and action [5]), which have basically achieved the effect of vehicle navigation and path planning in complex traffic environments by combining

sensor fusion such as advanced cameras, radar, laser radar (LiDAR), and deep learning algorithms in software technology [6]. Currently, artificial intelligence empowers vehicles to understand their surroundings, including recognizing other vehicles, pedestrians, traffic signs and road conditions, and to make corresponding driving decisions based on real-time changes, increasing the proportion of intelligence in the active supervision of self-driving vehicles.

Society of Automotive Engineers International (SAE) is the world’s leading authority in mobility standards development. It defines 6 levels of driving automation ranging from 0 to 5 [7], namely No driving automation (L0), Driver assistance (L1), Partial driving automation (L2), Conditional driving automation (L3), High driving automation (L4), and Full driving automation (L5). Currently, the autonomous driving industry is mostly at L2 and L3.

L2 systems, such as Tesla’s Autopilot [8], can control the vehicle’s steering and speed decisions in specific situations, but still require the driver to be responsible for the car and be ready to take over control at all times [9]. L3, on the other hand, are able to take full control of driving tasks under certain conditions, but still require human intervention and control when encountering complex and changing scenarios. Some companies are experimenting with L4 technologies that enable fully automated driving in specific areas and conditions, but are generally limited to closed or controlled environments [7].

There are a number of factors that prevent autonomous driving technology from being fully automated at this time, The four main challenges to current development are as follows:

System reliability testing: The system safety of autonomous driving is at the core, and is a major human concern, and involves how to maintain a high level of reliability in unpredictable road conditions and complex traffic environments. In the latest self-driving models, testers and suppliers are constantly figuring out how to try to attempt multi-type sensor fusion, advanced computer vision technologies, and improve vehicle performance in road test experiments such as automated lane tracking, assisted parking, or collision detection, in order to adapt as much as possible to more real-world demands and possibilities [10]. However, when faced with practical challenges, road tests, which are the final arbiter of safety performance, are often scheduled at the end of the design cycle and pose the same life-safety risks to researchers [11].

Ethical decision-making: The ethical challenges include the establishment of regulations applicable to different stage classes of self-driving vehicles and the apportionment of liability for accidents. Attempts to explain or define the perception, reasoning and decision-making processes present in the vehicle’s behavior can also increase the difficulty and challenge for the acceptance of the population and the establishment of laws and regulations [10].

Learning Models and Technical Constraints: The technical limitations are mainly in terms of the interpretability of deep learning models. The limited

dataset bias, black-box characteristics, nonlinear complexity, and opacity of the learning process in deep learning models can directly or indirectly increase the difficulty of interpretability.

Regulatory and legal issues/level of consumer trust: Increasing public awareness of the safety and effectiveness of self-driving vehicles is key to improving their acceptance. Currently, most driving tasks are actively supervised or assisted by self-driving vehicles, and the pressure on drivers has declined as a result of being assisted by Artificial Intelligent [12], but due to the lack of systematic transparency and interpretability in traditional drive control and neural network models, humans often fail to understand some of the behaviors accomplished by self-driving vehicles while in operation. Therefore, it is particularly important to create laws and regulations that clearly and explicitly explain the behavior of machines when running neural drive models, improve the corresponding laws and regulations in the field of autonomous driving, and try to obtain and improve general social acceptance.

With the challenges above, the important advances in automated driving technology, achieving full automation and widespread application, up to full automation (L5) and widespread use in everyday traffic, still requires overcoming these technical, legal, and social acceptance challenges and attempting to make breakthroughs in key areas [7]. An important step forward in this field is the development of Explainable Artificial Intelligence (XAI), which aims to solve the opaque nature of very sophisticated machine learning models, notably in the field of deep learning [13]. In order to guarantee that decisions made by AI are not only accurate but also trustworthy and transparent, XAI works towards the goal of making these processes as interpretable as possible to humans [14]. The heart of XAI resides in the fact that it bridges the gap between the advanced capabilities of AI systems and the human need for understanding and trust [15]. Significant strides of XAI were made in the field of autonomous driving, led to significant advancements [16]. The decision-making process of Convolutional Neural Networks (CNNs) was the subject of an investigation that led to the development of an innovative XAI method in the field of autonomous driving [17]. The primary objective is to examine and extract output feature produced by neurons in the hidden layers [18]. Experiments have been conducted using attention mechanisms that concentrate on visible elements causally linked to the driver's actions, and natural language processing to articulate the vehicle's maneuvers [19]. Moreover, recent research in computational Neuroscience and Cognitive Robotics developed robotic models that mimic human driving behavior (e.g. how human select attention and action [5,20]). Collectively, they significantly augment both the interpretability and reliability of AI decision-making in autonomous vehicles, with a keen focus on crucial elements like pedestrians and traffic signals [21].

The main purpose of the current work is to try to build an autonomous driving model configurator and simulation tester by using the software combination that has been gradually matured, to provide a safer and less costly solution for

the future testing of autonomous driving vehicles based on the multiple data sets and diverse data collection and adjustments therein. At the same time, the experiment also highlights and adopts two different vehicle operation decision-making algorithms (quadratic planning and dynamic planning) for path planning and speed planning, respectively, under the effect of the software combination, through the collection and sampling of certain specific data, in order to achieve the effect of testing the simulated environment. The data collection and analysis is also used to try to improve the safety and comfort level of the self-driving vehicle by using the data for rational optimisation and later improvement.

2 Materials and Methods

2.1 Materials

The software used in this project includes CarSim, Prescan, MATLAB and Simulink, the combination of these applications creates a sophisticated simulation platform that is crucial for the development and testing of autonomous driving systems (Please refer to Table 1 in Appendix A for a detailed comparison of the software). The following is a brief introduction to the software and the function and role of each application used in this work:

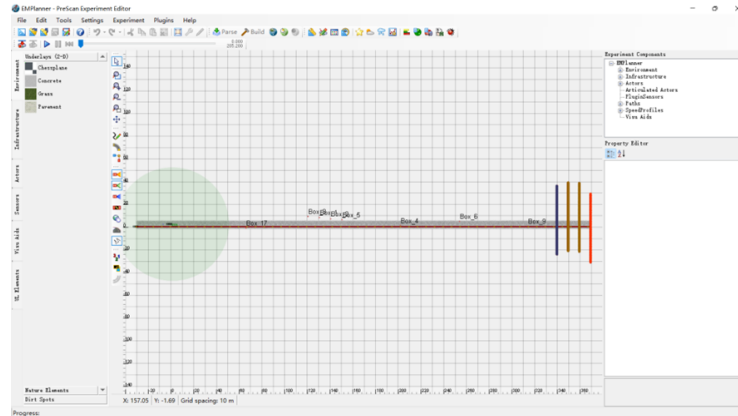


Fig. 1. A screenshot of PreScan[22]. Completion of the environmental model in PreScan (including obstacles, pedestrians, and vehicles). In the environment setting interface of prescan, you can add various obstacles such as houses, vehicles, pedestrians, trees, etc. to assist the test, and arrange the location according to the experimental requirements, so as to achieve the purpose of testing the vehicle algorithm and unexpected conditions.

Prescan is primarily utilized for generating complex traffic scenarios and simulating sensors[22] (Fig.1). The software creates intricate virtual environments that encompass diverse road configurations, traffic scenarios, and pedestrian behaviors. Prescan also emulates the data that would be acquired by vehicle sensors

in these situations, including LiDAR, radar, and camera feeds. The simulations are essential for generating authentic scenarios that an autonomous vehicle could potentially face (Fig.2).

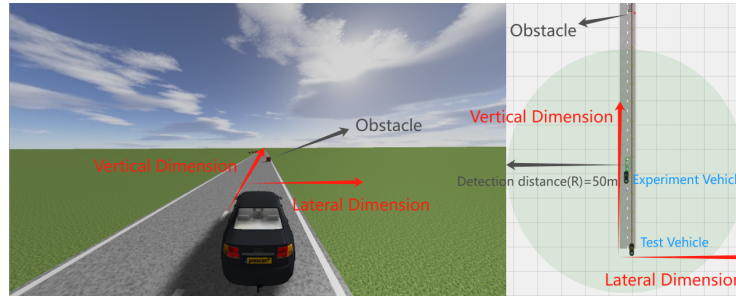


Fig. 2. Coordinate System Selection and Vehicle Initial Path Setting. In the following introduction, the coordinate system will be unified, and the direction of vehicle travel will be defined as the vertical dimension, and the direction perpendicular to the vehicle travel will be defined as the lateral dimension. The test vehicle first performs the path point setup and the experiment vehicle is responsible for the final practical data collection through the path points which provided by test vehicle, followed the trajectory from beginning and ending of the test vehicle; Detection distance is kind of the specific area which could be flexible and changeable to detect the obstacles by using Prescan, which is already set up at 50 meter by Lidar.

After the completion of the environmental model in Prescan ver.8.5.0, the sensor data it produces is transferred to MATLAB R2022b [23]. In the MATLAB environment, the data is employed to create and enhance decision-making algorithms that are crucial for the operation of autonomous vehicles. These algorithms encompass intricate procedures such as object detection, path planning, and decision-making in diverse traffic scenarios. MATLAB’s robust computational tools enable the testing and optimization of these algorithms, guaranteeing their effectiveness and reliability.

Once the decision-making algorithms are created in MATLAB, they are incorporated into CarSim ver.2019.1.[24], see in Fig.3. CarSim is responsible for accurately simulating the physical dynamics of the vehicle. The system utilizes the decision-making algorithms of MATLAB to simulate the physical response of a virtual vehicle to the decisions made. This encompasses the vehicle’s acceleration, deceleration, and maneuvering in reaction to the simulated environment. CarSim’s accurate representation of vehicle dynamics guarantees that the simulated reactions are highly realistic.

Simulink is crucial in integrating these tools (details can be found in *Methods* section). The platform facilitates instantaneous interaction among Prescan, MATLAB, and CarSim. Simulink facilitates the exchange of data between these software tools, guaranteeing that the choices made by the algorithms in MATLAB [25], which rely on Prescan’s sensor simulations, are faithfully represented

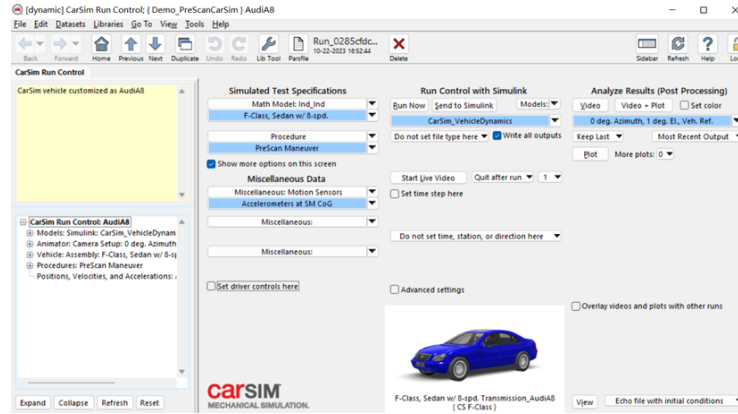


Fig. 3. A screenshot of CarSim[24]. Physical modeling setup for vehicles (The choice here is the more compatible Audi A8). In terms of the versatile function from Carsim, which consist of road settings, friction settings, vehicle performance settings, gradient settings, etc.

in CarSim’s vehicle dynamics[24]. This integration enables the seamless and instantaneous simulation of autonomous driving, wherein modifications in the surroundings (such as a pedestrian entering the road) promptly impact the vehicle’s actions.

The combination of Prescan, CarSim, MATLAB & Simulink, Python and TensorFlow altogether results in a comprehensive and interactive simulation platform with intricate details and dynamic capabilities. It facilitates comprehensive testing and development of autonomous driving technologies, serving as a crucial tool for engineers to evaluate and improve the safety and effectiveness of autonomous vehicles in various situations.

2.2 Methods

Master Functions for Autonomous Driving Decision Making. In the main function framework of autonomous driving decision making, we chose the optimization technique of dynamic programming for path planning, which facilitates the decomposition of a large problem into small interrelated problems, and was dedicated to logically solving complex decision problems with overlapping sub-problems and optimal sub-structures. In the decision-making problem of autonomous driving, dynamic planning of paths mainly solves the optimal path problem from the starting point to the end point, which usually involves the global consideration of the whole route. It evaluates various possible path alternatives, considering obstacles, route length, possible traffic conditions, etc., to find the path with the lowest cost (Fig.4). The implementation process was conducted in five steps, namely 1) Initialization, 2) Cost Calculation, 3) Recycling Process, 4) Path Backtracking, and 5) Output Path.

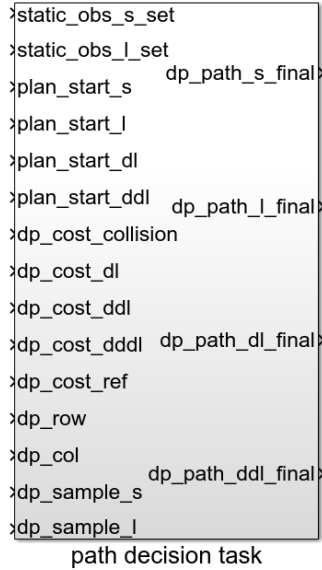


Fig. 4. The block of Path decision main task from Simulink

First, the minimum cost from the starting point to each node and the minimum cost from the starting point to the previous node of the optimal path were recorded using the matrices *"node_cost"* and *"pre_node_index"*, respectively. A node in path planning is a point (in a dynamically planned grid) where the vehicle may be subjected to a travel strategy to reach a particular state due to a combination of node costs such as time, distance and path smoothing. The initial settings of the current node and the previous node were recorded separately in order to be able to track and obtain the entire path history when the optimal path was finally determined. In this way, we combined the planned path information with the speed information and used linear fitting to provide the model with a smoother and more adaptive, real-time update of the path information and speed of the dual-modal smoother and more natural change curve (as shown in Fig.5).

Second, the initial cost calculation includes the possible costs from the first node at the starting point to all subsequent nodes. It directly affected the entire path planning strategy and path generation. Given the fifteen input arguments as shown in Fig.4 (left side), three types of cost calculations were performed in the functions *"CalcNeighbourCost"* and *"CalcStartCost"*, namely Smoothness Cost, Reference Trajectory Cost, and Crash Cost.

The smoothness cost *cost_smooth* is given by Eq.1:

$$cost_smooth = w_{smooth_v} \sum_{i=1}^n v_i^2 + w_{smooth_a} \sum_{i=1}^n a_i^2 + w_{smooth_a'} \sum_{i=1}^n a_i'^2 \quad (1)$$

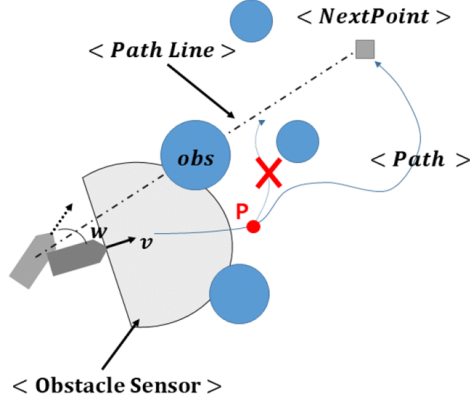


Fig. 5. Process Schematic. Under a specific or previously set path, the vehicle will consider multiple obstacles within its detection range and plan a path that considers all of them.

where v_i , a_i , and a'_i are velocity, acceleration, and jerk (rate of change of acceleration) at path point i , along the direction of the road (defined as vertical dimension, see Fig.2), respectively. w_{smooth_v} , w_{smooth_a} , and $w_{smooth_a'}$ are the corresponding smoothness cost weights.

The reference trajectory cost $cost_ref$ was given by Eq.2:

$$cost_ref = w_{ref} \sum_{i=1}^n ref_i^2 \quad (2)$$

where ref_i denotes the deviation of the reference trajectory from path point i , and w_{ref} is the weight of the reference trajectory cost.

The collision cost $cost_collision$ was given by Eq.3:

$$cost_collision = \sum_{j=1}^m CalcObsCost(w_{collision}, d_j^2) \quad (3)$$

Here, d_j^2 is the square of the distance between path point j and the obstacle, and $w_{collision}$ is the weight of the collision cost.

Third, for each decision step, the system calculated the cost from all nodes in the previous step to each current node, and considered all possible paths coming from the nodes in the previous steps and calculates the total cost to reach the current node. Whenever a lower cost path was found, the generation value in "node_cost" was updated again. The source node of this path was then recorded in "pre_node_index".

Next, after traversing all the nodes in each step that meet the requirements, it found the path that had the smallest cumulative cost from the starting point

to the end point (cumulative optimal), and used the index of the previous node recorded in the "pre_node_index" to backtrack, such that the optimal path was obtained after traversal.

Last, based on the backtracking results, the sequence of the nodes was obtained and displayed in the *s-l* coordinate system. The dynamic obstacles and the vehicle's speed planning were added on the original basis, thus it is necessary to output the vertical position *l* and longitudinal position *s* of each node in the subsequent steps, also including the dynamics of velocity *v* and acceleration *a* for two-speed planning and control, which completed the path information and facilitated the execution of the vehicle control system.

Velocity dynamic planning The implementation process includes four steps (Fig.6), namely 1) Set sampling point, 2) Initialize the cost matrix and speed store, 3) Fill the dynamic programming matrix, and 4) Results backtracking.

First, non-uniform sampling and uniform time sampling were used as the basis, through the definition of "*s_list*" and "*s_list*", non-uniform spatial sampling concentrates the computational resources at the beginning of the planning path, while uniform time sampling was used to simplify the time dimension problem.

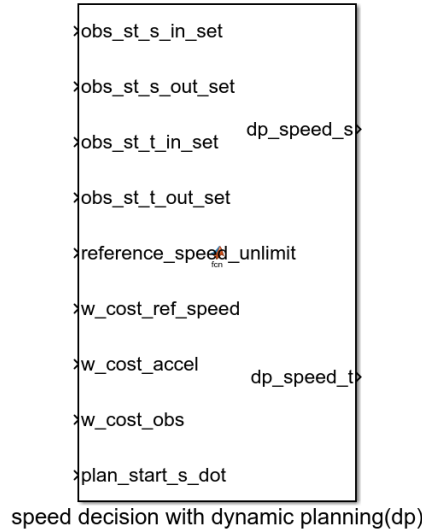


Fig. 6. Speed decision with Dynamic planning

Second, the "*dp_st_cost*" was initialized to infinity to indicate that there was no feasible path cost initially. "*dp_st_s_dot*" was used to store the speed information of each node. The costing and speed information were stored for each traversal as a key for later updates.

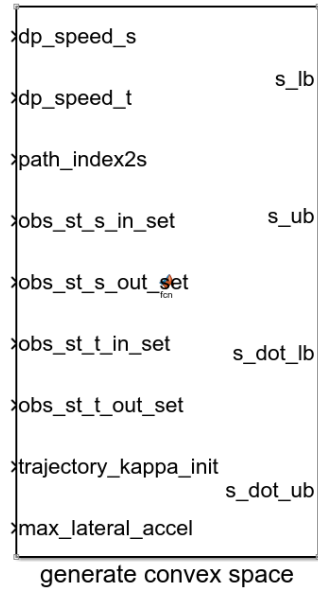


Fig. 7. Generate convex space

Third, each node calculates the cost and velocity from the previous node to all possible spatial points using the "*CalcDpCost*" function, which contained cost information such as obstacles, acceleration, and deviation from the reference velocity, and traverses each time and spatial point using a double loop. At each node, the cost transferred from the previous time step was considered and the lowest cost and corresponding velocity at the current node were updated.

Last, we traversed the edges of the dynamic planning matrix (all spatial points at the last time step and all-time steps at the last spatial point) to find the point with the lowest cost. Starting from the point with the lowest cost, the optimal speed selection was constructed step by step using the former node information "*dp_st_node*" back to the starting point.

Velocity quadratic planning preparation (velocity and acceleration)

Since the types of obstacles, we consider the simulation phase as dynamic and static. Thus it requires the vehicle to follow appropriate speed and acceleration boundaries during travelling, especially when encountering dynamic obstacles it would be more restrictive. The obstacles were mapped and localized to the vehicle to ensure that the vehicle's speed was planned to be within the safety and technical constraints (as shown in Fig.7).

The implementation process includes two steps (Fig.7), namely 1) Vehicle edge dynamics modelling, 2) Dynamically adjust speed limits.

First, a vehicle-side dynamics model in autonomous driving decisions describes the maximum speed. With such model, a vehicle can safely travel on a

given curvature path. The parameters are specified as in Eq.4:

$$\nu^2 = \frac{a_{vertical}}{k} \tag{4}$$

- ν : vehicle velocity
- $a_{vertical}$: the maximum vertical acceleration of the vehicle, i.e., the maximum vertical acceleration due to centripetal force when the vehicle is travelling in a curve
- k : curvature, a measure indicating the degree of curvature of the path, $k = \frac{1}{R}$, where R is the radius of the curve.

The important use of this model is to ensure that the vehicle does not skid or lose control due to excessive speed when travelling in curves, thus ensuring safe driving. The model is used to calculate the maximum safe speed for each curvature, as in Eq.5:

$$cost_smooth = \sqrt{\frac{max_vertical_accel}{cur_kappa}} \tag{5}$$

where $max_vertical_accel$ is the maximum vertical acceleration, and " cur_kappa " refers to the $k(kappa)$, where the smoothness cost function combines the curvature and the maximum vertical acceleration to ensure the safety of the vehicle at any curvature, with the help of the calculation in Eq.5, which helps to determine the maximum safe speed of the vehicle at different curves.

Next, dynamically adjusting speed boundaries in an autopilot system mainly considers the effects of dynamic obstacles:


a. Obstacle localization: the mean position " obs_s " and time " obs_t " of an obstacle is calculated from the start and end spatial positions of the obstacle " $obs_st_s_in_set$ ", " $obs_st_s_out_set$ " as well as the time window " $obs_st_t_in_set$ ", " $obs_st_t_out_set$ ".

b. Speed limit adjustment: If the obstacle is in front of the vehicle, reduce the speed to the upper boundary to avoid a collision, depending on the speed and location of the obstacle; If an obstacle is behind the vehicle, raise the speed lower boundary to help the vehicle pass the obstacle by accelerating.

Finally, this dynamic adjustment allows the vehicle to respond flexibly to unexpected situations, by defining clear speed and acceleration bounds and establishing the necessary convex optimization space for subsequent quadratic planning algorithms. Driving adaptability, safety and flexibility are enhanced in a simulated environment, a key step in solving optimal control commands.

Quadratic Velocity Programming The implementation process includes four steps (Fig.8), namely 1) Construction of continuity constraint, 2) Application of boundary constraints, 3) Construction of cost function, and 4) Quadratic programming solution, and 5) Result processing and output.

```

>plan_start_s_dot
>plan_start_s_dot2    qp_s_init>
>dp_speed_s
>dp_speed_t
>s_lb                qp_s_dot_init>
>s_ub
>s_dot_lb           
>s_dot_ub          qp_s_dot2_init>
>w_cost_s_dot2
>w_cost_v_ref
>w_cost_jerk    relative_time_init>
>speed_reference

```

speed planning with quadratic programming (qp)

Fig. 8. Speed planning with Quadratic planning

First, the equation constraints in quadratic programming ensure continuous changes in velocity and acceleration from one point in time to the next. This is constructed through the matrix A_{eq} , which contains a linear relationship for the velocity and acceleration at the current and next time points.

Secondly, the lb and ub arrays define specific ranges of values for velocity and acceleration that are determined based on static analyses (e.g., road conditions, vehicle performance) and dynamic inputs (e.g., traffic conditions ahead, obstacles).

Next, by defining a cost matrix H and a vector f , the QP algorithm minimizes a cost function that contains the speed, acceleration, and its variation (Jerk). This optimization is directly related to the driving comfort and energy efficiency of the vehicle.

Furthermore, using the *quadprog* function, the optimal velocity and acceleration configurations are solved for based on the given H -matrix (cost term), f -vector (linear term), A -matrix (inequality constraints), A_{eq} -matrix (equation constraints), and upper and lower bounds.

Finally, the outputs include secondary planning and optimized speed, acceleration, position, and relative time. These results provide the vehicle control system with accurate velocity and acceleration commands for speed control in real-world driving.

3 Results

After data collection in the test environment, the output results focus on the speed and acceleration information of the self-driving car in the vertical direction and in the lateral line of defence to measure the safety factor and comfort level of the car.

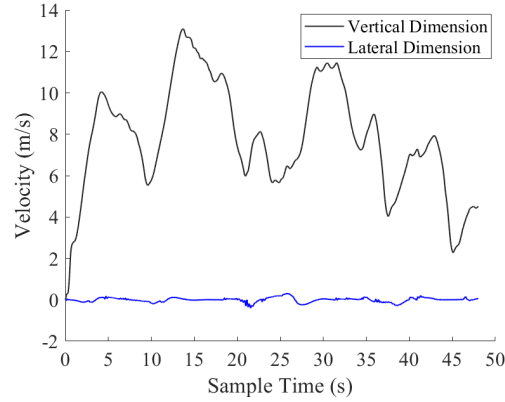


Fig. 9. Vehicle Profile. The solid black line represents the real-time change in speed of the vehicle in the direction of travel (i.e., Vertical Dimension); the blue dotted line represents the real-time change in speed of the vehicle in the direction of steering (i.e., Lateral Dimension).

Throughout the experiment, the dynamic monitoring of speed can be used to provide the current simulation test environment with some vehicle interpretable decisions and actions, such as when to decelerate, when to accelerate, etc., and the relevance and accuracy of the decisions can also be inferred from the curves below (shown in Fig.9), especially on both dynamic and static obstacles.

In the results panel, the speed information in both directions was an important condition indicator, but since we should not only pursue safety but also focus on comfort in the field of autonomous driving, the dynamic display of acceleration in both directions were analysed and presented as in Fig.10. According to the acceleration curve, we could clearly observe that the model has quite a lot of irregular jitters in both directions, indicating that there were still some loopholes and deficiencies in the model's ability to handle some obstacle avoidance and detection of the surrounding environment. Especially for the model in this experiment, the range used for detecting the surrounding environment was fixed as a circled area with a radius of 50m. It was inevitable to deal with several different types of obstacles at the same time and combine the vehicle control behaviors under each different decision during operation.

The absolutely acceleration change of the vehicle was then calculated based on the dynamic change of acceleration in both directions (Fig.10). In order to

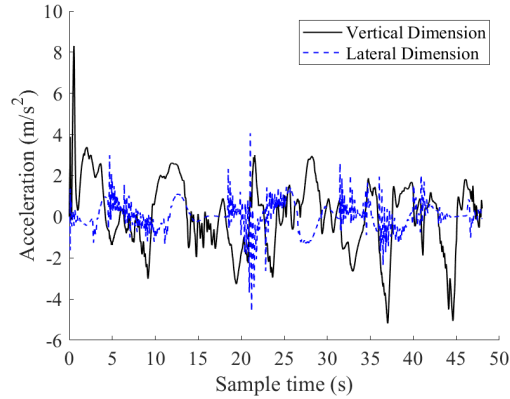


Fig. 10. Vehicle acceleration profile. The solid black line represents the real-time change in acceleration of the vehicle in the traveling direction (i.e., Vertical Dimension); the blue dotted line represents the real-time change in acceleration of the vehicle in the steering direction (i.e., Lateral Dimension).

investigate the improvement of acceleration on the comfort experience of the self-driving vehicle, and to investigate the interpretability of the decisions in the model, we set a threshold value to monitor the stages at which the acceleration of the vehicle changes significantly. In the current work, an acceleration of 3 m/s^2 was chosen as the threshold, considered to be discomfort for passengers [26]. We counted the number of times this threshold was exceeded in the testing, as a judgement of the model and the simulated environment performance in comfort. In the example shown in Fig.11, we observed four times that the acceleration exceeded 3 m/s^2 , and only one of the four times it was well above the threshold, reaching around 5 m/s^2 .

In the previous model, the prediction of unexpected situations on the road was insufficient, and it could only avoid obstacles with almost no deceleration, which was still a safety hazard and lacks the verification of interpretable vehicle behavior information even in the simulation environment. In the above-mentioned results, it could be clearly seen that although the acceleration of the vehicle exceeds the threshold four times in the test environment of nearly one minute. In other works, it made up for the improvement of the safety performance of the vehicle by trading-off minimal discomfort. We also observed the following performances of the model in our testing: 1) it treated the static obstacles on the non-travelling path as the dynamic obstacles with the speed of 0, simulating the occurrence of pedestrians on the side of the road or corner emergencies, 2) it dealt with the problem of the speed decreasing of the dynamic obstacles getting closer, 3) it braked and de-accelerated significantly as the the obstacle were getting too close, i.e. the "cost" increased largely in those occasions, 4) it could make decision of slowing down to avoid dynamic obsta-

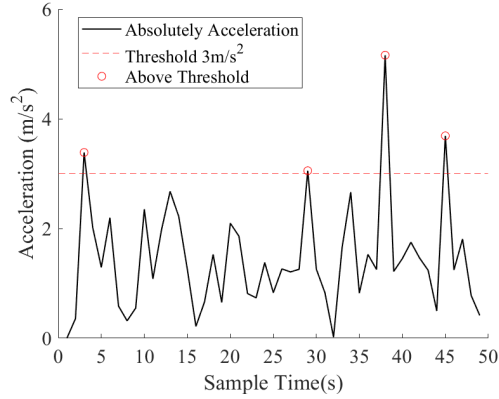


Fig. 11. Absolutely Acceleration of vehicle profile with the threshold ($a = 3m/s^2$). The black plot represents the acceleration profile, the red dashed line represents the threshold $3m/s^2$, and the red circle points out the occasions that the acceleration exceeded the threshold.

cle travelling in horizontal dimension (e.g. pedestrian), and then accelerated to overtake safely.

4 Discussion

The aim of the current work was to build a decision-making model for autonomous driving that provides visual scenario simulations and dynamic displays in terms of safety and explanatory aspects [27]. We demonstrated that only dynamic display data such as speed and acceleration were used for the evaluation criteria of the previous model, which was obviously insufficient for judging the performance of a decision-making model. At the level of automatic driving, the performance of the vehicle includes the passage of different road sections, the passage efficiency, and the collision situation, the performance of the model for a variety of different complex road conditions and emergencies, etc.

We developed a complete framework to test autonomous driving decision-making and planning algorithms in a realistic simulation, though the evaluation of the performance was not fully quantitative. Specifically, our model processes information from perception results, and performs explainable decision-making and planning. We believe our data-driven approach could facilitate more objective assessments and guide further improvements. It could be easily integrated to perception or control models of autonomous vehicles for enhancing the safety and explain-ability of a complete autonomous model, which could contribute to the future development of autonomous driving technology towards L4 or L5 [28].

Possible future work includes the following directions. First, *Vision and Multi-sensor Fusion*, combining data from cameras, LiDAR, radar, and other

sensors, improving the vehicle’s perceptual ability to detect and respond to various obstacles and conditions in real-time [29]. Recent advances in sensor fusion algorithms have shown promising results in improving the robustness and accuracy of autonomous vehicle perception systems. By integrating these advancements, future systems can achieve high levels of situational awareness and reliability. Second, *Handling Real-Time Large-Scale Data and Complex Environments*, optimizing data processing pipelines and employing more sophisticated machine learning techniques to ensure robust performance in diverse and dynamic scenarios [30]. The challenge of processing vast amounts of sensor data in real-time requires innovative solution, such as edge computing and distributed processing frameworks, which can help manage the computational load and enhance system responsiveness. Third, *Higher Computing Resources and Accurate Environmental Data*, exploring the use of more powerful computing platforms and cloud-based solutions to meet computational demands, improving the fidelity of simulations and enhance the reliability of the autonomous driving system in real-world applications [31]. Leveraging high-performance computing (HPC) and cloud infrastructure can significantly accelerate the development and testing cycles of autonomous driving algorithms, enabling more comprehensive validation and iteration processes. And last but not the least, *Quantitative Evaluation of Autonomous Driving Performance*, by systematically measuring various aspects of performance, such as response times, obstacle avoidance success rates, and passenger comfort levels, gaining a clearer understanding of system strengths and weaknesses [32]. Implementing standardized metrics and benchmarks for autonomous driving performance evaluation can provide valuable insights and facilitate comparison across different systems and approaches.

Recent studies have emphasized the importance of multi-sensor fusion and advanced data processing techniques in achieving reliable autonomous driving systems. For instance, according to a survey by Zhang et al., the integration of various sensor modalities is critical for enhancing the perception capabilities of autonomous vehicles and addressing the challenges posed by dynamic environments [33]. Additionally, advances in edge computing and 5G have demonstrated potential in managing the computational demands of real-time data processing for autonomous systems [34,35].

Acknowledgements This project is supported by XJTU Research Development Fund RDF-23-01-072 to FZ.

Open practices statement Datasets generated and analyzed during the current study, including the process and state of running in the final test environment, could be found available in the OSF repository https://osf.io/b7r5d/?view_only=06a67f4d795a4e8e92bb0f03c8780fbe upon publication. None of the experiment reported in the present study was preregistered.

References

1. Yurtsever, E., Lambert, J., Carballo, A., Takeda, K.: A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access* **8**, 58443–58469 (2020). <https://doi.org/10.1109/ACCESS.2020.2983149>
2. Von Foerster, H.: Perception of the future and the future of perception. *Instructional Science* **1**, 31–43 (1972)
3. Schwarting, W., Alonso-Mora, J., Rus, D.: Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems* **1**, 187–210 (2018)
4. Kuutti, S., Bowden, R., Jin, Y., Barber, P., Fallah, S.: A survey of deep learning applications to autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems* **22**(2), 712–733 (2020)
5. Makwana, M., Zhang, F., Heinke, D., and Song, J.H.: Continuous action with a neurobiologically inspired computational approach reveals the dynamics of selection history. *PLoS Computational Biology* **19**(7), e1011283 (2023).
6. Meyer, M., Kuschik, G.: Deep learning based 3D object detection for automotive radar and camera. In: *Proc. of the 16th European Radar Conference (EuRAD)*, pp. 133–136 (2019)
7. Fussell, L. M., Daily, J., Haines, H., Roseberry, S., White, J. J.: The Society of Automotive Engineers Clean Snowmobile Challenge 2001-Summary and Results. SAE Technical Paper (2001)
8. Dikmen, M., Burns, C. M.: Autonomous driving in the real world: Experiences with Tesla autopilot and summon. In: *Proc. of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pp. 225–228 (2016)
9. Zhang, Z., Fan, Y., Jiang, W., Han, G., Yang, C., Cong, J.: AutoPilot: A platform-based ESL synthesis system. In: *High-Level Synthesis: From Algorithm to Digital Circuit*, pp. 99–112. Springer (2008)
10. Zablocki, E., Ben-Younes, H., Pérez, P., Cord, M.: Explainability of deep vision-based autonomous driving systems: Review and challenges. *arXiv:2101.05307 [cs.CV]* (2022). <https://arxiv.org/abs/2101.05307>
11. Myers, A.: How AI Is Making Autonomous Vehicles Safer. <https://hai.stanford.edu/news/how-ai-making-autonomous-vehicles-safer#:~:text=,Andrew%20Myers>. Last accessed 7 Mar 2022
12. Mankodiya, H., Jadav, D., Gupta, R., Tanwar, S., Hong, W.-C., Sharma, R.: OD-XAI: Explainable AI-Based Semantic Object Detection for Autonomous Vehicles. *Applied Sciences* **12**(11), 5310 (2022). <https://doi.org/10.3390/app12115310>
13. A. Adadi and M. Berrada, "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)," in *IEEE Access*, vol. 6, pp. 52138–52160, 2018, doi: 10.1109/ACCESS.2018.2870052.
14. Q. V. Liao and K. R. Varshney, "Human-Centered Explainable AI (XAI): From Algorithms to User Experiences," 2022. [Online]. Available: arXiv:2110.10790 [cs.AI].
15. K. Ćyras, A. Rago, E. Albin, P. Baroni, and F. Toni, "Argumentative XAI: A Survey," 2021. [Online]. Available: arXiv:2105.11266 [cs.AI].
16. H. S. Kim and I. Joe, "An XAI method for convolutional neural networks in self-driving cars," *PLoS One*, vol. 17, no. 8, e0267282, Aug. 2022. DOI: 10.1371/journal.pone.0267282. PMID: 35972916; PMCID: PMC9380950.
17. H.-S. Kim and I. Joe, "An XAI method for convolutional neural networks in self-driving cars," *PLoS One*, vol. 17, no. 8, e0267282, 2022.

18. T. Rojat, R. Puget, D. Filliat, J. Del Ser, R. Gelin, and N. Díaz-Rodríguez, "Explainable artificial intelligence (xai) on timeseries data: A survey," arXiv preprint arXiv:2104.00950, 2021.
19. T. Rojat, R. Puget, D. Filliat, J. Del Ser, R. Gelin, and N. Díaz-Rodríguez, "Explainable artificial intelligence (xai) on timeseries data: A survey," arXiv preprint arXiv:2104.00950, 2021.
20. Patil, M., Heinke, D., Zhang, F.: Reveal the Distractions of the Irrelevant Features using a Neurobiologically Plausible Cognitive Robotics Model. In Rita 2023-The 11th International Conference on Robot Intelligence Technology and Applications 2023 Oct 31.
21. P. Gohel, P. Singh, and M. Mohanty, "Explainable AI: current status and future directions," arXiv preprint arXiv:2107.07045, 2021.
22. TASS International. PreScan. Version 2022.1. Helmond, The Netherlands: TASS International, 2022. Available: <https://tass.plm.automation.siemens.com/prescan>
23. The MathWorks, Inc. MATLAB. Version 9.12.0 (R2022a). Natick, Massachusetts: The MathWorks, Inc., 2022. Available: <https://www.mathworks.com/products/matlab.html>
24. Mechanical Simulation Corporation. CarSim. Version 2022.1. Ann Arbor, Michigan: Mechanical Simulation Corporation, 2022. Available: <https://www.carsim.com/products/carsim/index.php>
25. The MathWorks, Inc. Simulink. Version 10.3 (R2022a). Natick, Massachusetts: The MathWorks, Inc., 2022. Available: <https://www.mathworks.com/products/simulink.html>
26. K. N. de Winkel, T. Irmak, R. Happee, and B. Shyrokau, "Standards for passenger comfort in automated vehicles: Acceleration and jerk," *Appl. Ergon.*, vol. 106, p. 103881, Jan. 2023. doi: 10.1016/j.apergo.2022.103881. Epub 2022 Sep 2. PMID: 36058166.
27. VincentWong3. (2021). Automatic-driving-decision-and-planning-for-matlab. Retrieved from <https://github.com/VincentWong3/automatic-driving-decision-and-planning-for-matlab>.
28. E. Lehtonen, J. Wörle, F. Malin, B. Metz, and S. Innamaa, "Travel experience matters: Expected personal mobility impacts after simulated L3/L4 automated driving," *Transportation*, pp. 1-20, 2022.
29. J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, ... and J. Williams, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727-774, 2008.
30. A. Kumar, M. Boehm, and J. Yang, "Data management in machine learning: Challenges, techniques, and systems," in *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 1717-1722, May 2017.
31. J. Fadaie, "The state of modeling, simulation, and data utilization within industry: An autonomous vehicles perspective," arXiv preprint arXiv:1910.06075, 2019.
32. RGBSI, "Sensor Fusion in Autonomous Driving Systems: Part 2," RGBSI Blog, May 21, 2019. [Online]. Available: <https://blog.rgbsi.com/sensor-fusion-autonomous-driving-systems-part-2>. [Accessed: May 23, 2024].
33. J. Zhang and S. Singh, "A survey of advances in multi-sensor data fusion approaches for autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1897-1915, Mar. 2021. doi: 10.1109/TITS.2020.2977552.
34. M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30-39, Jan. 2017. doi: 10.1109/MC.2017.9.

35. D. Li, Y. Lu, and S. Wu, "Research and application on the technology of the SPN intelligent operation and maintenance based on autonomous network," *Telecom Engineering Technics and Standardization*, vol. 37, no. 5, pp. 48-53, May. 2024. doi: 10.3969/j.issn.1008-5599.2024.05.010

A Appendix A

Below is a side-by-side comparison of the basic information of the software platforms we implemented:

Table 1. Comparison of various simulators [31]. Y=supported, N = not supported, U = unknown

Requirements	MATLAB (Simulink)	Carsim	PreScan
Perception: Sensor models supported	Y	Y	Y
Perception: Support for different weather conditions	N	N	Y
Camera Calibration	Y	N	Y
Path planning	Y	Y	Y
Vehicle Control: Support for proper vehicle dynamics	Y	Y	Y
3D Virtual Environment	U	Y	Y
Traffic infrastructure	Y, allow to build lights model	Y	Y
Traffic Scenario simulation: Support of different types of Dynamic objects	Y	N	N
2D/3D Ground Truth	Y	N	N
Interfaces to other software	Y, with Carsim, Prescan, ROS	Y, with MATLAB (Simulink)	Y, with MATLAB (Simulink)
Scalability via a server multiclient architecture	U	U	U
Open source	N	N	N
Well-maintained/Stable	Y	Y	Y
Portability	Y	Y	Y
Flexible API	Y	Y	U