A Comparative Survey of Leading Models in CARLA Simulator

B08502041 Yun-Fang Li[1] and B09902054 Chang-Yen Li[2]

[1,2]National Taiwan University

August 31, 2025

Abstract

This report does not have any double assignment or any completed work before this semester. This project is a comparative **survey** of leading autonomous driving models in the CARLA simulator, focusing on ReasonNet, InterFuser, TransFuser++, and the TCP framework. It evaluates their performance in addressing key challenges such as handling rare events, sensor fusion, training biases, and combining control-based and trajectory-based approaches. ReasonNet utilizes temporal and global reasoning for improved prediction and decision-making in urban scenarios. InterFuser enhances sensor data fusion with convolutional neural networks and transformer models. TransFuser++ simplifies architecture and mitigates biases, achieving high performance with fewer inputs. The TCP framework integrates both trajectory and control-based methods to optimize control signals and reduce infractions. The primary contribution of this work is that we summarize the methodology of each model, discuss their limitations, and suggest potential explanations for their performance.

1 Introduction

The development of autonomous driving has experienced significant progress in the last decade, propelled by breakthroughs in artificial intelligence, machine learning, and sensor technology. Advancements in the pursuit of creating completely self-driving vehicles that can safely maneuver through intricate surroundings have resulted in notable achievements. Nevertheless, despite these technological breakthroughs, there are still some significant obstacles that hinder the mainstream acceptance and implementation of autonomous driving systems. This literature analysis emphasizes the significance of tackling these obstacles in order to make significant progress towards achieving fully autonomous cars.

We prioritize the CARLA simulator due to its practicality in autonomous driving research. CARLA is a open-source urban driving simulator that creates a wide variety of driving situations, ranging from ordinary to extremely intricate, which are essential for testing and verifying autonomous driving algorithms. The high-fidelity simulation environment enables researchers to conduct experiments with different conditions and edge situations that are challenging to encounter in real-world testing. CARLA's evaluation framework has three fundamental metrics: Driving Score, Route Completion, and Infraction Penalty. Driving Score assesses overall driving performance by combining route completion and infraction penalties. Route Completion measures the percentage of the predefined route that the autonomous vehicle successfully completes. Infraction Penalty quantifies the number and severity of traffic infractions committed by the autonomous vehicle.

By employing these measures, researchers may thoroughly assess the efficiency and security of self-driving systems in a regulated, simulated setting, facilitating progress in practical, real-life implementations. Therefore, we have chosen the models that exhibit the highest performance in the CARLA simulator. We survey how these models mitigate the following challenges: handling rare events, sensor fusion, addressing model training biases, and limitations of adopting control-based or trajectory-based approaches alone.

2 Tackling Challenges

2.1 Handling Rare Events

The majority of current techniques largely concentrate on addressing common and regular driving scenarios, while inadequately handling infrequent or unexpected events, such as quick pedestrian movements from obstructed locations or unanticipated maneuvers by other vehicles. The lack of this capability highlights the urgent requirement for a more comprehensive and unified method to improve the accuracy of perception and prediction in real-time, especially in crowded and intricate urban traffic situations.

ReasonNet [1], an innovative driving framework, tackles these difficulties by utilizing both temporal and global scene information. The objective is to offer accurate forecasts of how the environment will change and enhance the resilience of decision-making in different urban driving scenarios.

2.1.1 Brief Explanation of ReasonNet

Input and Output Representations. ReasonNet primarily relies on sensor data from a LiDAR sensor and many RGB cameras as its main inputs. The vehicle's left, front, right, and rear cameras capture images from distinct viewpoints, while the LiDAR sensor generates point cloud data for the creation of 3D models of the environment. ReasonNet generates multiple types of outputs to facilitate safe and efficient driving. These include a path prediction with multiple waypoints for the vehicle to follow, an occupancy map indicating the presence of objects in the vicinity, and traffic rule information such as the status of traffic lights and stop signs. The path prediction guides the vehicle's navigation, while the occupancy map and traffic rule information help in making real-time driving decisions.

Model Architecture. ReasonNet is built upon three fundamental modules: The Perception Module, Temporal Reasoning Module, and Global Reasoning Module.

The Perception Module processes and combines sensor information to build BEV features. The image data use a two-dimensional backbone, whereas the LiDAR data employs a three-dimensional backbone. The BEV decoder combines these features to generate a comprehensive BEV map. This module also predicts the locations and features of navigation-related waypoints and traffic signs.

The Temporal Reasoning Module maintains a memory bank to store historical features for the purpose of managing temporal information. The method utilizes an attention-based strategy to include past features and present frame data. This approach facilitates the monitoring of objects that experience periodic occlusion and enables more precise forecasts of their future movements.

The Global Reasoning Module tracks interactions and linkages between objects and the environment in order to detect adverse events such as occlusions. In order to ensure consistency between the expected waypoints and occupancy maps, a method for modeling object-environment interaction is employed. This method includes an occupancy decoder that generates occupancy maps, as well as a consistency loss to maintain coherence. This module enhances overall perception performance and vehicle safety by studying imperceptible environments and anticipating potential dangers.

2.1.2 Limitations

Although ReasonNet greatly enhances the ability of autonomous vehicles to navigate in urban environments, it is important to investigate its inherent limits in more depth. An important issue is the framework's reliance on sensor data of excellent quality. When sensor inputs are affected by unfavorable weather circumstances like fog or heavy rain, the performance of ReasonNet may decline, emphasizing the need of having strong sensory processing and data integration capabilities.

ReasonNet has a constraint in terms of computational load due to the integration of complicated temporal and global reasoning modules. Implementing a data-intensive framework in real-time could be difficult in situations when quick decision-making is necessary. This factor could restrict the implementation of ReasonNet in less powerful hardware environments that are commonly found in commercially accessible vehicles.

2.2 Sensor Fusion

Autonomous driving systems must provide a thorough comprehension of the environment and prioritize safety, which is made challenging by the requirement to analyze data from multiple types of sensors. Conventional methods of combining sensors, such as matching geometric features and basic concatenation, are unable to accurately represent the intricate relationships between diverse sensor modalities. These constraints lead to weaknesses in crucial traffic scenarios, such as the abrupt presence of people or vehicles violating traffic signals.

InterFuser [2] tackles these difficulties by utilizing sophisticated methods to efficiently combine data from various sensors. Previous methodologies had challenges in terms of scalability and interpretability, frequently resulting in compromised safety and dependability in autonomous driving systems. InterFuser utilizes a combination of convolutional neural networks and a transformer model to effectively analyze and merge sensor input. This advanced architecture significantly improves the system's capability to manage various intricate driving situations.

2.2.1 Brief Explanation of InterFuser

Input and Output Representations. InterFuser employs a set of four sensors to acquire a thorough comprehension of the scene: three RGB cameras (located on the left, front, and right sides) and one LiDAR sensor. These sensors offer additional perspectives of the surroundings. The RGB cameras record photos from various angles, while a supplementary focusing-view image is generated by extracting the central portion of the front RGB image to specifically highlight distant traffic lights. The LiDAR point cloud data is transformed into a 3-bin histogram using a 3-dimensional Bird's Eye View (BEV) grid. This transformation produces a two-channel LiDAR BEV projection picture input.

InterFuser produces two distinct categories of outputs: safety-insensitive and safety-sensitive. The safety-insensitive outputs consist of a path prediction of 10 waypoints for the ego vehicle to adhere to. The safety-critical outputs consist of an item density map and information on traffic rules. The object density map offers comprehensive data on potential objects in the surrounding area, while the traffic rule information encompasses the current state of traffic lights, the presence of stop signs, and if the vehicle is located at a junction.

Model Architecture. The architecture of InterFuser is specifically engineered to effectively manage the intricate spatial and modal intricacies of sensors used in autonomous driving. The system is composed of three primary elements: a convolutional neural network (CNN) as the foundation, a transformer encoder, and a transformer decoder. The updated ResNet architecture is utilized to process each sensor input and extract feature maps. The feature maps capture essential visual and spatial details from the pictures and LiDAR data, making them ready for subsequent processing in the transformer phases.

The feature maps that have been recovered are then inputted into a transformer encoder. This encoder utilizes a sequence of self-attention techniques to effectively combine information from various sensors and perspectives. This stage captures intricate interdependencies and correlations among different elements, hence augmenting the model's capacity to comprehend the scene in a contextual manner.

The transformer decoder utilizes the incorporated characteristics from the encoder to provide predictions and interpretable outputs. The system utilizes many inquiries that pertain to different facets of driving, including waypoints, traffic density, and adherence to traffic regulations. The implementation of a multiheaded attention method enables the system to selectively concentrate on pertinent attributes for each individual driving decision, hence augmenting the precision and safety of the results.

The transformer decoder is accompanied by three concurrent prediction modules for forecasting the waypoints, object density map, and traffic rule information. The waypoints are forecasted using a single-layer GRU model with autoregressive capabilities. The density map of the object is anticipated by passing the matching embeddings from the transformer decoder through a multi-layer perceptron (MLP) for transformation. The traffic rule information is estimated using a solitary linear constraints.

Safety Controller Leveraging Intermediate Interpretable Features. The safety controller is a crucial component of InterFuser, utilizing the intermediate interpretable features produced by the transformer decoder. It utilizes these characteristics to improve driving selections by guaranteeing that all activities are limited inside secure operational limitations.

The safety controller utilizes a two-threshold criterion to identify objects and applies dynamic propagation with a moving average to anticipate trajectories. This method guarantees the reliable identification and anticipation of moving entities and infrequent occurrences. The controller modifies the vehicle's path and velocity by utilizing the object density map and traffic rule information. It promotes safe driving by consistently keeping a safe distance from adjacent objects and strictly following traffic restrictions.

The objective of solving a linear programming optimization issue is to find the optimal velocity that maximizes driving efficiency while maintaining safety. This entails analyzing the trajectories of nearby objects and making appropriate adjustments to the vehicle's movements. InterFuser enhances the safety and dependability of autonomous driving systems by integrating sophisticated approaches, establishing itself as a prominent solution in the industry.

2.2.2 Limitations

Although InterFuser achieves good Driving Score, it does have several limits and areas that can be improved upon in the future. The system persists in encountering traffic violations, with instances of failure stemming from problems such as imprecise anticipation of the direction of other vehicles, miscalculating the speed of bicycles, or the inability to detect pedestrians who have fallen.

The current solution employs a two-threshold criterion to detect objects and use dynamic propagation with a moving average to predict trajectories. Although these strategies are adequate for present standards, they may not be optimal for more intricate and diverse driving conditions.

Furthermore, the utilization of numerous sensors and intricate fusion mechanisms poses difficulties in relation to processing resources and the ability to scale in real-time, particularly when deployed in locations with limited resources. It is imperative to tackle these difficulties in order to ensure the effectiveness of practical implementations.

2.3 Addressing Training Biases

Recent advancements in end-to-end driving systems have shown remarkable improvements, but they also introduce certain biases that can impact real-world performance. Two significant biases are lateral recovery bias and waypoint ambiguity bias, both of which stem from the models' interaction with target points (TPs) and waypoints, respectively. Understanding these biases is crucial for developing more robust and reliable driving systems.

Lateral recovery bias refers to the tendency of these systems to correct compounding steering errors over time by steering towards predefined TPs along a route. These TPs provide geometric cues that the model uses for navigation. When a vehicle deviates from its intended path, the model steers back towards the nearest TP, typically recentering the vehicle within the lane. However, reliance on TPs as corrective signals can lead to issues when the TPs are too far apart, causing the model to make large, abrupt steering adjustments that can result in catastrophic errors.

Waypoint ambiguity bias arises from using waypoints as output representations. This creates ambiguity because the model must commit to a single point estimate, despite the multimodal nature of future vehicle velocities. While this allows for smooth interpolation between paths, it often fails to capture the full range of possible future states, leading to inaccuracies.

Transfuser++ (TF++) [3] is developed to mitigate the impact of the two biases. TF++ is significantly simpler than the previous state-of-the-art, Interfuser, yet it outperforms it by a large margin on the Longest6 benchmark. Utilizing approximately four times less data, one camera instead of four, and eliminating the need for complex heuristics to extract throttle and brake commands for the path output, TF++ achieves superior performance.

2.3.1 Brief Explanation of TransFuser++

Input and Output Representations. The input representation for TransFuser++ includes RGB images, LiDAR bird's eye view (BEV) data, and GNSS coordinates of target points (TPs). These inputs are processed through a multi-modal fusion architecture. The model output consists of path predictions and target speeds, which guide the vehicle's steering and speed.

Model Architecture. TransFuser++ integrates multiple sensor modalities to create a comprehensive environmental representation essential for autonomous navigation. The architecture includes components for image and bird's-eye view (BEV) segmentation, depth estimation, bounding box prediction, and target speed classification.

The model processes inputs from an RGB camera and a LiDAR sensor. The RGB camera captures high-resolution images, while the LiDAR sensor provides 360° coverage of the surroundings. These inputs are processed through parallel branches, which allow the model to capture both detailed visual information and spatially extensive geometric data:

Image Branch: Uses a convolutional neural network backbone to extract features from the RGB images.
BEV Branch: Processes voxelized LiDAR data through convolutional layers to produce BEV features.

For addressing lateral recovery bias, TransFuser++ replaces global average pooling (GAP) with a transformer decoder to retain spatial information in the BEV features. This design choice mitigates the shortcut learning of steering directly towards TPs, which leads to catastrophic errors like cutting turns. The transformer decoder utilizes cross-attention mechanisms, maintaining the spatial context and effectively integrating features from different sensor modalities. This approach ensures the model makes nuanced decisions based on comprehensive environmental understanding rather than overly relying on TPs. To address the longitudinal averaging bias, TransFuser++ disentangles the path prediction from target speed prediction.

- Path Prediction: Using a recurrent neural network (GRU) decoder to predict the vehicle's future path as a series of waypoints spaced by fixed distances rather than fixed time intervals. This method isolates the geometric path from temporal predictions, reducing ambiguity.
- Target Speed Prediction: Implementing a multi-layer perceptron (MLP) classifier to predict target speeds. This classifier incorporates prediction uncertainties, allowing the model to consider multiple possible future velocities. By weighting the target speeds according to their predicted confidence, the model can slow down under uncertainty, thus reducing collisions.

Limitations

Although Transfuser++ deals with common biases of other models, it still has a few noteworthy limitations.

One of the primary limitations of TransFuser++ is its focus on low-speed urban driving scenarios. The model's validation primarily occurs within the CARLA simulator, which involves relatively low speeds (under 35 km/h). This constrained environment does not account for the complexities and challenges associated with high-speed driving or highway conditions, leaving its performance in these scenarios untested and unverified. As a result, the applicability of TransFuser++ in real-world settings involving varied speed requirements remains uncertain.

A significant aspect of TransFuser++'s functionality is its dependence on target points (TPs) for navigation and error recovery. This reliance introduces two main issues. First, the model assumes precise localization and mapping to provide accurate TPs, which CARLA ensures. However, in real-world environments, such precision may not always be guaranteed, potentially leading to navigation errors. Second, TransFuser++ tends to exhibit shortcut learning by heavily depending on nearby TPs to correct its path, especially when it deviates from the lane. This behavior can result in significant steering errors, such as cutting turns too sharply when TPs are far from the vehicle's current position, leading to unsafe driving behaviors.

The evaluation scenarios for TransFuser++ do not include static obstacles within lanes, such as parked cars or debris, which are common in urban environments. Consequently, the model's ability to navigate around these obstacles has not been thoroughly tested, limiting its effectiveness in real-world urban driving scenarios where such obstacles are frequent.

TransFuser++ uses waypoints as an output representation, which is inherently ambiguous due to the entanglement of future vehicle velocities with the path. This ambiguity can impact the model's ability to predict accurate future positions under varying speed conditions, potentially compromising navigation and control precision. Although this continuous nature helps in reducing collisions by interpolating to slow down, the representation is less interpretable and does not explicitly expose uncertainties.

While TransFuser++ incorporates improvements such as transformer-based pooling and data augmentation, these techniques come with their limitations. The shift from global average pooling (GAP) to transformer decoders preserves spatial information but requires significant computational resources, which may not be efficient for real-time applications. Moreover, the shift and rotation augmentations used to simulate lateral recovery scenarios may not comprehensively cover the wide range of potential real-world deviations and errors, limiting the model's robustness to unseen disturbances.

2.4 Control-Based and Trajectory-Based Approaches

End-to-end autonomous driving methods, which directly map raw sensor data to a planned trajectory or low-level control actions, show the virtue of simplicity. The output prediction of the model for end-to-end autonomous driving generally falls into two forms: trajectory/waypoints and direct control actions.

For methods that plan trajectory, additional controllers needed as a subsequent step to convert the planned trajectory into control signals. One potential supremacy of trajectory-based prediction is that it actually considers a relatively longer time horizon into the future. However, turning the trajectory into control actions so that the vehicle could follow the planned trajectory is not trivial. The industry usually adopts sophisticated control algorithms to achieve reliable trajectory-following performance. Simple PID controllers may perform worse in situations such as taking a big turn or starting at the red light due to the inertial problem.

On the other hand, for control-based methods, the control signals are directly optimized. Nevertheless, their focus on the current step may cause deferred reactions to avoid potential collisions with other moving agents. The independence between the control predictions of different steps also makes the actions of the vehicle more unstable or discontinuous.

How to combine the strengths of both trajectory-based and control-based methods remains an open and active area of research in autonomous driving. One promising approach involves a hybrid framework that leverages the advantages of both techniques to achieve better overall performance. Wu et al. (2022) proposed TCP [4] framework, packing these two branches into a unified framework.

2.4.1 Brief Explanation of TCP framework

Input and Output Representations. TCP framework takes as input the sensor signal from a monocular camera, the vehicle speed, and high-level navigation information that includes a discrete navigation command and target coordinates from the global planner. The model outputs control signals, which consist of longitudinal controls—throttle and brake—and a lateral control signal, steer.

Model Architecture. The whole architecture of TCP framework is comprised of an input encoding stage and two parallel subsequent branches. The input image goes through a CNN based image encoder, to generate a feature map $\mathbf{F}$. In the meantime, an MLP based encoder takes the speed and the navigation information and outputs measurement feature $\mathbf{j}_{\mathrm{m}}$. The encoded features are shared by both the trajectory-based and control-based prediction branches. The predictions from these two branches are then merged using a situation-based fusion strategy to produce the final control signal.

The trajectory planning branch first generates a planned trajectory comprised of waypoints at K steps for the agent to follow, and then the trajectory is processed by low-level controllers to get the control actions. The image feature map $\mathbf{F}$ is average pooled and concatenated with the measurement feature $\mathbf{j}_{\mathrm{m}}$. This combined feature is then fed into a GRU, which auto-regressively generates future waypoints to form the planned trajectory. With the planned trajectory, the magnitudes of the vectors between consecutive waypoints represent the desired speed and are sent to PID controllers to get throttle, brake, and steer actions.

The multi-step control prediction branch in TCP framework addresses the limitations of prior control-based models by introducing a novel approach. Unlike previous methods, which rely solely on the current input and IID assumption, TCP predicts multiple future actions to accommodate sequential decision-making in closed-loop tests, where historical actions affect future states. To overcome the challenge of predicting future actions with only current sensor inputs, a temporal module leveraging GRU is employed. This module captures dynamic environmental information, such as object motion and traffic changes, by auto-regressively feeding predicted actions into GRU.

Trajectory-Guided Attention. To enhance spatial consistency and incorporate crucial static details like curbs and lanes, the control branch is guided by the trajectory branch, ensuring attention is focused on relevant regions of the input image at each future time step. TCP achieves this by learning an attention map that extracts significant information from the encoded feature map $\mathbf{F}$. At each time step t, this attention map is calculated using the hidden states of the GRU from both the control and trajectory branches through an MLP. The map aggregates the feature map $\mathbf{F}$, combining it with the control branch's hidden states to create an informative representation, which contains both static and dynamic environmental information. This representation is then fed into a policy head, shared across all time steps, to predict the corresponding control action.

Finally, to further leverage the advantages of both trajectory and control branch predictions, a situation-based fusion strategy was adopted. When the ego vehicle is turning, the control branch is more advantageous; therefore, weights of α and $1 - \alpha$ are assigned to the control branch and trajectory predictions, respectively. Otherwise, the weights are reversed, with $1 - \alpha$ assigned to the trajectory prediction and α to the control branch prediction.

2.4.2 Limitations

The TCP framework is an innovative approach that combines the strengths of both trajectory-based and control-based methods to achieve a more effective driving model. However, despite its promising results, the framework has several notable limitations that may impede its broader applicability and optimal performance.

Multi-Sensor Integration. One significant limitation of the TCP framework is its reliance on GRU to handle temporal dependencies. While GRUs are effective for such scenario, their applicability becomes limited in scenarios involving multiple sensor inputs. The current GRU implementation in TCP is primarily designed to handle monocular camera input, which constrains its ability to integrate and process multi-sensor data effectively. Autonomous vehicles often rely on a variety of sensors, such as LiDAR and radar, in addition to cameras, to obtain a comprehensive understanding of the environment. The TCP framework's inability to seamlessly incorporate these diverse sensor inputs limits its potential for achieving better performance in complex driving scenarios.

Trajectory-Guided Attention Generalization. Another limitation of the TCP framework is its approach to trajectory-guided attention. While adopting various trajectory-based methods could significantly enhance trajectory signals, the current design of TCP's attention mechanism is tightly integrated with its own trajectory prediction model. This makes it challenging to incorporate alternative trajectory-based approaches that might perform better. Consequently, this lack of generalization restricts the framework's adaptability and its potential for optimization by integrating more advanced or complementary trajectory prediction techniques.

Situation-Base Fusion. The experimental results in the paper underscore a significant limitation: the use of a fixed constant hyperparameter α in situation-based fusion. This parameter controls the weighting between the control branch and trajectory branch predictions across different driving scenarios. However, a fixed α can lead to suboptimal results because the ideal weighting may change with different driving contexts. For example, various road conditions and traffic densities might necessitate different α values for optimal performance. Adopting a more dynamic and adaptable approach, where α is adjusted in real-time based on contextual cues, might improve the framework's responsiveness and effectiveness in diverse driving situations. Thus, the current rigidity of this parameter setting limits the potential for optimal performance across varying scenarios.

3 Comparison of the Four Methods

3.1 Performance

In the CARLA simulator, the performance of autonomous driving models is evaluated using three crucial metrics: Driving Score, Route Completion, and Infraction Penalty [5].

Driving score is the main assessment indicator, combining route completion and penalties for violations. It is the multiplication of the penalty factor for the violation and the percentage of completion of the route.

Route completion measures the majority of allocated routes successfully completed by the automatic agent. It is expressed in percentage of the total route.

Infraction Penalty uses geometric sequence to punish agents for various driving violations (such as crashes or traffic violations), thereby reducing overall driving scores.

The table below shows the performance of each model [6].

	Driving Score	Route Completion	Infraction Penalty
ReasonNet	79.95	89.89	0.89
InterFuser	76.18	88.23	0.84
TCP	75.14	85.63	0.87
TF++	66.32	78.57	0.84

Table 1: Performance metrics of four different models in the CARLA simulator

3.2 Comparison

3.2.1 Safety Performance

H. Shao et al. (2023) introduced the Drive in Occlusion Simulation (DOS) benchmark to better evaluate the performance and security of ReasonNet. The DOS benchmark provides a comprehensive assessment of autonomous vehicles using four challenging types: parking, sudden braking, left-wing and red light breaches, with 25 different situations for each type.

The benchmark is designed to test ReasonNet's time-effective reasoning capabilities for intermittent closed objects, as well as its ability to conduct global reasoning for continuous blocked object with interactive clues. This comprehensive assessment ensures that ReasonNet's predictive and perceptive capabilities are thoroughly tested.

Test results on the DOS benchmark showed that ReasonNet scored the highest points in infraction penalty on CARLA leaderboard. Studies have confirmed the key role of temporary and global reasoning modules, which have been deleted and which have seen a decrease in performance.

3.2.2 InterFuser v.s TransFuser++

TF++ and InterFuser enhance TransFuser, although they employ distinct approaches to achieve this improvement. TF++ surpasses TransFuser by prioritizing the mitigation of training biases, hence facilitating the recovery from deviations and ensuring consistent speed. However, it fails to particularly tackle the issues of completing routes or improving the overall driving score.

InterFuser surpasses TransFuser because to its all-encompassing sensor fusion methodology and its strong focus on safety and interpret ability. Consequently, this leads to enhanced comprehension of the environment and more prudent choices when driving.

InterFuser demonstrates superior performance compared to TF++ on the CARLA leaderboard. We believe that the main factor behind this is that although TF++ tackles training biases, it does not offer explicit techniques to enhance route completion, resulting in decreased driving scores. InterFuser's extensive scene comprehension and adherence to safety limitations lead to improved driving scores and superior overall performance, establishing it as the leading option for autonomous driving in intricate surroundings.

3.2.3 Trajectory-Based v.s TCP

The models we surveyed are trajectory-based, with the exception of TCP. TCP employs a hybrid approach, as it considers the control branch, where control signals are directly optimized. This approach is designed to mitigate the inertia problem commonly associated with PID controllers, which are adopted by the trajectory-based methods. Consequently, TCP achieves a better Infraction Penalty score. However, ReasonNet, a trajectory-based method, outperforms TCP and achieves state-of-the-art performance. We believe this is because TCP's trajectory branch does not generate signals as robust as ReasonNet's. Additionally, TCP uses fewer sensors than ReasonNet. Incorporating additional sensors, such as Lidar, would require significant changes to TCP's architecture. Moreover, the dataset's limited non-trivial conditions may not fully showcase TCP's strengths.

4 Conclusion

This survey presents an in-depth evaluation of advanced autonomous driving models using the CARLA simulator. The models examined are ReasonNet, InterFuser, TransFuser++, and the TCP framework. Each model addresses different challenges inherent in autonomous driving, such as handling rare events, sensor fusion, training biases, and integrating control-based and trajectory-based approaches.

ReasonNet focuses on handling rare events through temporal and global reasoning modules, significantly enhancing decision-making in complex urban environments. However, it is limited by its high reliance on high-quality sensor data and computational load.

InterFuser uses advanced convolutional neural networks and transformer models to fuse data from multiple sensors, improving safety and interpretability. It addresses the limitations of conventional sensor fusion methods but faces challenges related to real-time scalability.

TransFuser++ simplifies the model architecture while adding a few extra components to mitigate biases in end-to-end driving systems. It separates path prediction from speed prediction to reduce errors associated with lateral recovery and waypoint ambiguity. Despite its improvements, it is primarily validated in low-speed urban scenarios and has limitations regarding static obstacle detection and real-world application.

TCP framework integrates trajectory-based and control-based methods to optimize control signals and improve stability. It uses a hybrid approach to mitigate the inertia problems of PID controllers and the deferred reactions of control-based methods. However, it may face the challenge of limited generalizability when incorporating other trajectory-based methods to improve trajectory signals.

In summary, this report concludes the limitations and performance of different models, and potential explanations for their performance. While these works are not flawless and face numerous challenges in real-world application, they still contribute to the broader understanding and advancement of this research domain.

5 Contribution by Member

- Equal contribution:
  - Recorded presentation video.
  - Contributed to the introduction, comparison, and conclusion sections.
- Yun-Fang Li:
  - Provided summaries of [1], [2], [3].
- Chang-Yen Li:
  - Provided summaries of [3], [4].
  - Edited presentation video.
  - Formatted the report.

References

[1] H. Shao, L. Wang, R. Chen, S. L. Waslander, H. Li, and Y. Liu, "Reasonnet: End-to-end driving with temporal and global reasoning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13723–13733, 2023.

[2] H. Shao, L. Wang, R. Chen, H. Li, and Y. Liu, "Safety-enhanced autonomous driving using interpretable sensor fusion transformer," in *Conference on Robot Learning*, pp. 726–737, PMLR, 2023.

[3] B. Jaeger, K. Chitta, and A. Geiger, "Hidden biases of end-to-end driving models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8240–8249, 2023.

[4] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, and Y. Qiao, "Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline," *Advances in Neural Information Processing Systems*, vol. 35, pp. 6119–6132, 2022.

[5] "Carla autonomous driving leaderboard: Evaluation and metrics." `https://leaderboard.carla.org/#evaluation-and-metrics`, 2024. Accessed: 2024-06-07.

[6] "Carla leaderboard benchmark (autonomous driving)." `https://paperswithcode.com/sota/autonomous-driving-on-carla-leaderboard`, 2024. Accessed: 2024-06-07.