2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 1-5, 2023. Detroit, USA

ANEC: Adaptive Neural Ensemble Controller for Mitigating Latency Problems in Vision-based Autonomous Driving

Aws Khalil[1] and Jaerock Kwon[2]

***Abstract*— Humans have latency in their visual perception system between observation and action. Any action we take is based on an earlier observation since, by the time we act, the state has already changed, and we got a new observation. In autonomous driving, this latency is also present, determined by the amount of time the control algorithm needs to process information before acting. This algorithmic perception latency can be reduced by massive computing power via GPUs and FPGAs, which is improbable in automobile platforms. Thus, it is a reasonable assumption that the algorithmic perception latency is inevitable. Many researchers have developed different neural network driving models without consideration of the algorithmic perception latency. This paper studies the latency effect on vision-based neural network autonomous driving in the lane-keeping task and proposes a vision-based novel neural network controller, the Adaptive Neural Ensemble Controller (ANEC) that is inspired by the near/far gaze distribution of human drivers during lane-keeping. ANEC was tested in Gazebo 3D simulation environment with Robot Operating System (ROS) which showed the effectiveness of ANEC in dealing with algorithmic latency. The source code is available at `https://github.com/jrkwon/oscar/tree/devel_anec`.**

Autonomous Vehicle Navigation; Machine Learning for Robot Control; Imitation Learning

I. Introduction

We are always living in the past. According to studies [1], [2], our conscious perception system is always behind the time we are. As a result, the brain must catch up to the present in order for us to live it. When we drive a car, for example, we sense our environment, assess it, make a decision, and then act on that decision. The action we take, however, is based on an earlier observation since by the time we act, the state has already changed and we have a new observation, as illustrated in Fig. 1. If we get an observation of the environment at time t_1, our brain needs time δ to process the observation, so we take action at time $t_1+\delta$. However at time $t_1+\delta$, the vehicle has moved, the environment has changed, and we have a new observation. There have been debates on how the brain produces present conscious awareness using past sensory information. One of the hypotheses is that visual perception is predictive, so sensory information is extrapolated ahead of the perceived event.

In autonomous driving, this latency is also present, determined by the amount of time the control algorithm needs to process information before acting. This algorithmic perception latency δ, shown in Fig. 2, is inevitable [3]. To reduce the latency, we may use massive computing power via GPUs, FPGAs, and multicore CPUs. In training a deep neural network, such a high computing power system can be a choice. Yet, in actual inference, computing resources are limited in automobile platforms. Thus, it is a reasonable assumption that there will be an unavoidable latency in the perception and actuation cycle. Most vision-based lateral control studies [4]–[11], assume no delay between observation O_t and action A_t at time t.

In this paper, we study the latency effect on autonomous driving in the lane-keeping task. Fig. 3 illustrates the latency. When vehicle B receives a visual input O_t, the steering angle is not S_t but $S_{t-\delta}$ because S_t will be available after δ from t. This delayed steering angle $S_{t-\delta}$ makes the vehicle at B be at C which is off track. The steering angle ${}^{p}S_{t-\delta}$ that is predictive inferred steering angle at time $t-\delta$ should be

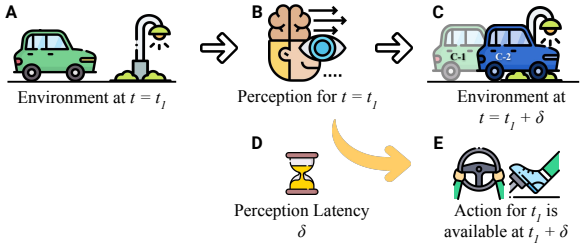

Fig. 1. Latency in the visual perception system of a human driver between observation and action. (A) The system observes its environment at time t_1 (O_{t_1}). (B) A conscious perception process starts at time t_1 and ends at time t_2. (D) δ is the perception latency, which is the time required to complete the perception process, where $\delta = t_2 - t_1$. (C) During the time period δ, the state of the vehicle has changed from C-1 to C-2 and we have a new observation ($O_{t_1+\delta}$ or O_{t_2}). (E) Because of δ, for observation O_{t_1} at t_1, the action A_{t_1} is only available at $t_1+\delta$. (Part of this figure has been designed using resources from Flaticon.com)

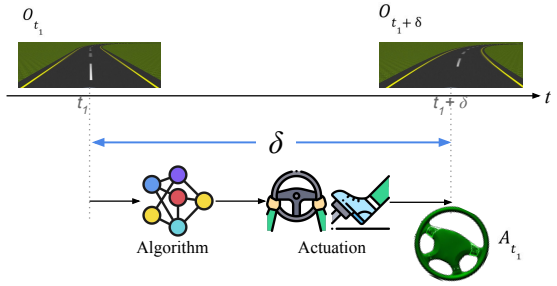

Fig. 2. The definition of the perception latency δ in autonomous driving. For observation O_{t_1} at time t_1, the corresponding action A_{t_1} is applied when observation $O_{t_1+\delta}$ is input at time $t_1+\delta$. (Part of this figure has been designed using resources from Flaticon.com)

Both authors are with the Department of Electrical and Computer Engineering, University of Michigan-Dearborn, Dearborn, MI, USA. awskh@umich.edu [1], jrkwon@umich.edu [2]

2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-9190-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/IROS55552.2023.10342520

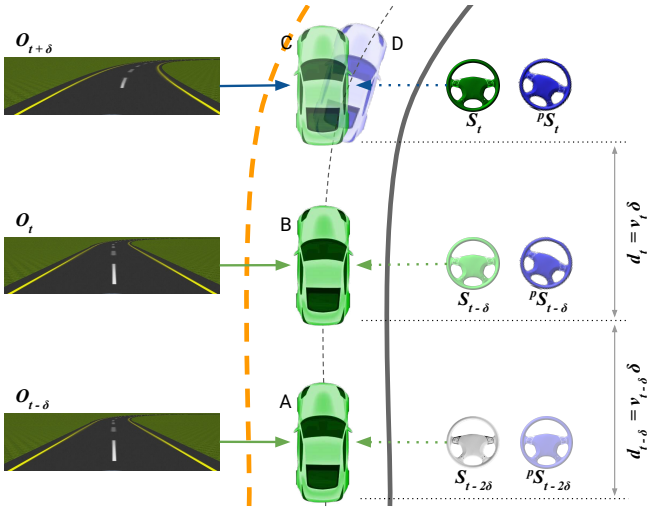

Fig. 3. Algorithmic perception latency in automobile platform. The vehicle at B has visual input O_t. $S_{t-\delta}$ is an inferred steering angle at time $t-\delta$ available at t. If $S_{t-\delta}$ is applied to the vehicle at time t, the vehicle location at time $t+\delta$ will be C that is about to be out off the track. The steering angle ${}^pS_{t-\delta}$ that is predictive inferred steering angle at time $t-\delta$ should be applied to B to locate the vehicle at D.

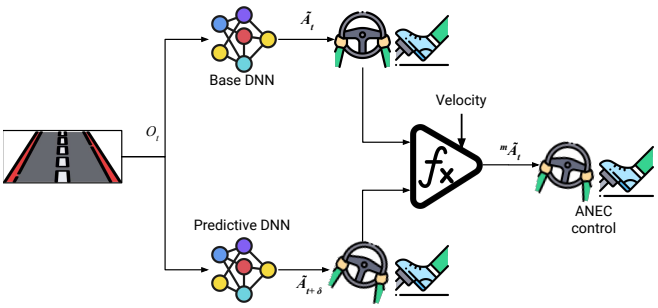

Fig. 4. General overview of the proposed Adaptive Neural Ensemble Controller (ANEC) for self-driving vehicles. The output of two driving models, the Base Model (BM) and the Predictive Model (PM), must be combined for ANEC to function. Like any vision-based neural network model, BM is anticipated to concentrate on the near-point area. On the other hand, PM is anticipated to examine the region beyond the near-point area in order to uncover latent characteristics that would guide future decisions. For the purpose of determining ANEC's ultimate output, each model will be given a dynamic weight based on the current speed of the vehicle. (Part of this figure has been designed using resources from Flaticon.com)

applied to B to locate the vehicle at D, because as Fig. 3 shows, during the latency time δ the vehicle has traveled distance d_t which is defined as $d_t = v_t\ \delta$, where v_t is the vehicle speed at time t and δ is the algorithmic perception latency. The change in the scene, which is the difference between two observations, is determined by the distance d_t traveled between the two observations. A larger effect of the latency δ can be expected in higher velocity and curved sections of the road. The changes in two scenes observed at time t and $t+\delta$ can be more significant when d_t is a higher value. For instance, a model could be able to drive at a speed up to 35 km/h, if it is trained at a maximum speed of 30 km/h, but it would likely not function effectively at higher speeds, such as 40 km/h. When vision-based neural network models, such as PilotNet [7] and CNN-LSTM, were employed for racing [11], neither model was able to finish the course. We believe the reason was that the high speed amplified the effect of the latency.

Perception latency does not prevent human drivers from successful in-time decision-making. According to [12], when they looked at the gaze distributions of human drivers during lane keeping, they discovered that most of the time (assuming there is no lead vehicle), humans hold their gaze at a distance in front of their own vehicle, which aids in anticipating future actions. Inspired by this gaze distribution, we hypothesize that visual input at time t has latent variables not only for the current state at time t but also for future states at time $t+\delta$ and the weight of importance of the future states varies with the current driving speed and curvature of the road.

We propose the Adaptive Neural Ensemble Controller (ANEC) to show that the algorithmic perception latency issue can be addressed by adaptively infusing the prediction action output into the baseline output. A high-level overview of the proposed system is shown in Fig. 4. ANEC depends on combining the output of two driving models, the Base Model (BM) and the Predictive Model (PM). BM is expected to focus on the near-point area like any vision-based neural network model. PM, on the other hand, is expected to cover the far-point area to extract latent variables for future actions. Note that near/far areas are not explicitly selected. BM and PM will figure out which areas are important to infer control output. A dynamic and adaptive weight, dependent on the vehicle speed, is assigned to each model to establish ANEC final output. The higher the speed, the greater the significance of future states, and hence the greater the weight assigned to PM.

II. Related Work

As noted in the preceding section, most vision-based neural network driving models from various research [4]–[11] were developed without taking the algorithmic perception latency δ into account. Instead, latency in autonomous driving has been discussed from different view angles. The majority of the work concentrate on deployment-related computational latency (i.e., hardware) [13]–[16] and network-related latency (i.e., communication) [17]–[22], occasionally both [23], [24]. In our study, we address the algorithmic latency that was covered by just a few research [3], [25]–[27].

Li et al. [3] emphasized how the problem of algorithmic latency should not be overlooked in online vision-based perception. Taking this latency into consideration, they presented a method for measuring the real-time performance of perception systems that quantifies the trade-off between accuracy and latency. Their approach might be a strong solution for well-defined problems but is not ideal for safety-critical systems such as automotive platforms. Mao et al. [25] investigated the algorithmic latency for video object detection, comparing the detection latency of several video object detectors. They proposed a metric to measure the latency, not a solution to deal with the latency. Kocic et al.

[26] attempted to reduce the algorithmic latency in driving by modifying the DNN architecture. This approach may reduce the latency but it is challenging to maintain the original accuracy. Wu et al. [27] highlighted that control-based driving models (image $\rightarrow$ control signal) have algorithmic latency and may fail since they focused on the current time step. To address this issue, they created Trajectory-guided Control Prediction (TCP), a multi-task learning system that combines a control prediction model with a trajectory planning model. Their approach requires extracting the exact trajectory which can be challenging.

In our approach, we accept that the latency is inevitable and try to reduce its effect by adaptively combining predicted future actions with the current action.

III. Method

In this section, we cover the ANEC framework, the architecture of neural networks, the training process, and finally the performance metrics used to evaluate ANEC.

A. Proposed System

The proposed ANEC for vision-based autonomous driving improves driving quality by implicitly scanning the road ahead. A detailed overview of ANEC is shown in Fig. 5. Since BM is like any vision-based neural network model, its policy π_ϕ^{BM} will provide the action $\widetilde{A}_t^{BM}$ as the corresponding action for the observation O_t given the vehicle's state S_t, as

$$\widetilde{A}_t^{BM} = \pi_\phi^{BM}(O_t|S_t). \tag{1}$$

On the other hand, because PM is expected to extract future actions from the observation, it does not only have access to the current state S_t but can also estimate the future state $\hat{S}_{t+\delta}$ of the vehicle. Hence, the PM's policy π_ϕ^{PM} minimizes the latency effect by providing the future action $\widetilde{A}_{t+\delta}^{PM}$ as the corresponding action to the current observation O_t, given the current state of the vehicle S_t and the predicted state $\hat{S}_{t+\delta}$, as

$$\widetilde{A}_{t+\delta}^{PM} = \pi_\phi^{PM}(O_t|S_t, \hat{S}_{t+\delta}). \tag{2}$$

Finally, each action from each model will be given a weight based on the vehicle's current speed as shown in (4) and (5), and the final action provided by ANEC is calculated as

$$\widetilde{A}_t^{ANEC} = w_{BM}\ \widetilde{A}_t^{BM} + w_{PM}\ \widetilde{A}_{t+\delta}^{PM}. \tag{3}$$

The speed of the vehicle has a significant impact on how latency affects driving quality. PM is supposed to scan the road ahead, so the higher the speed, the more we need it to take control. The two models are combined in such a manner that BM can act as the primary controller at low speeds and PM can act as the primary controller at high speeds. We employed an adaptive weight function that can dynamically vary based on the current speed to be able to create ANEC. The hyperbolic tangent function ($tanh$) was selected to adaptively represent the weights given to each model. Based on the current speed, the PM's weight w_{PM} is defined in (4) and the BM's weight can be calculated using (5).

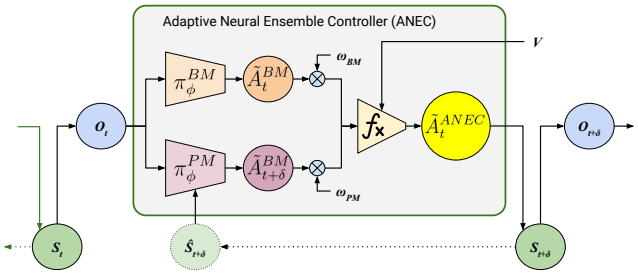

Fig. 5. A thorough description of the proposed Adaptive Neural Ensemble Controller (ANEC) for autonomous vehicles. According to BM's policy π_ϕ^{BM}, given the vehicle's state S_t, the action $\widetilde{A}_t^{BM}$ will be the appropriate response for the observation O_t. In contrast, PM is designed to extract future actions from the observation, which not only grants it access to the current state S_t but also enables it to estimate the future state $\hat{S}_{t+\delta}$ of the vehicle. To minimize latency effects, PM's policy π_ϕ^{PM} offers the future action $\widetilde{A}_{t+\delta}^{PM}$ as the corresponding action for the current observation O_t, taking into account both the current state of the vehicle S_t and the predicted state $\hat{S}_{t+\delta}$. Finally, ANEC incorporates each model's action, dynamically weighted based on the vehicle's current speed v, to calculate the final output $\widetilde{A}_t^{ANEC}$.

$$w_{PM} = (\alpha - \beta V_{max} + \gamma v_{cur}) tanh(\kappa(v_{cur} - V_p)) + 0.5, \tag{4}$$

$$w_{BM} + w_{PM} = 1, \tag{5}$$

where V_{max} is the maximum speed set by the user, v_{cur} is the current speed of the vehicle, and V_p is the pivot speed, which is the maximum speed achieved in the training data.

B. Neural Network Architecture

The two driving models, BM and PM, that we employ in our proposed system ANEC have the same model architecture shown in Fig. 6. This model architecture was adopted from our previous work [10], influenced by PilotNet [6] but no max-pooling layers were employed. Five convolutional layers and five dense layers are present. The first three convoluted layers are composed of 5 × 5 filters with 2 × 2 strides. The final two convoluted layers feature 3 × 3 filters with 1 × 1 stride. The dense layers have 1000, 100, 50, and 10 neurons, respectively. The network has a total of 11,054,019 trainable parameters.

C. Models Training

A training dataset for a vision-based neural network driving model can be defined as $D = \{O_t, A_t\}_{t=1}^N$, where O_t is an observation (i.e., visual input) at time t, A_t is the correlative action (i.e., steering angle), and the total time-steps is N.

As shown in Fig. 7, BM is trained like any vision-based neural network model, with a training dataset $D^{BM} = \{O_t, A_t\}_{t=1}^N$. Each observation collected at a certain time step (e.g., O_t collected at time t) with its corresponding driving data of the same time step (A_t) are fed to the DNN to learn the BM policy (π_ϕ^{BM}) by minimizing the prediction error of A_t when observation O_t is given ($D^{BM} \rightarrow \pi_\phi^{BM}$).

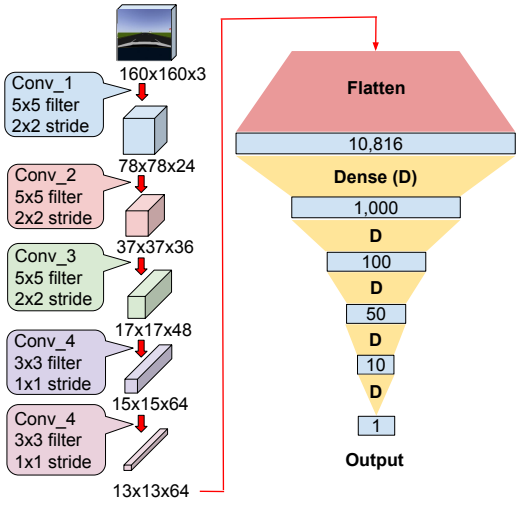

Fig. 6. The network architecture of BIMI-net. This model design is based on our earlier work [10] and was inspired by PilotNet [6]. However, no max-pooling layers were used. There are five Convolutional layers and five Fully-connected (Dense) layers. The model starts with three Convolutional layers with 5×5 filters with 2×2 strides. The fourth and fifth Convolutional layers feature 3×3 filters with 1×1 stride. The final Convolution layer is then flattened into a vector with a length of 10,816. This flat vector is fed to a series of Dense layers with the following number of neurons: 1000, 100, 50, and 10. The last Dense layer will give us the final single output which represents the steering angle.

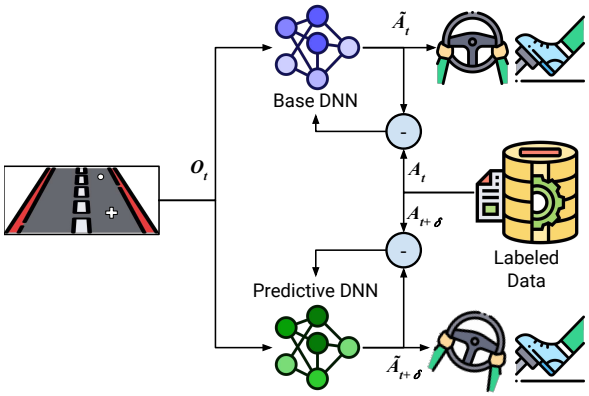

Fig. 7. Model Training. The BM's policy π_{ϕ}^{BM} is learned by the base DNN by providing it an observation collected at a specific time step with its corresponding action data of the same time step $\{O_t, A_t\}_{t=1}^{T}$ as input. On the other hand, for the PM to uncover latent variables for future actions, the PM's policy π_{ϕ}^{PM} is learned by Predictive DNN by feeding it an observation collected at a certain time step with future driving data at a future time step $\{O_t, A_{t+\delta}\}_{t=1}^{T-\delta}$, where δ is the algorithmic perception latency. (Part of this figure has been designed using resources from Flaticon.com)

The training dataset for PM is $D^{PM} = \{O_t, A_{t+\delta}\}_{t=1}^{N-\delta}$ as it is expected to extract latent variables for future actions from a given observation. The DNN will be fed O_t (i.e., the image collected at time t) with $A_{t+\delta}$ (i.e., future driving data at the time step $t+\delta$), where δ is the algorithmic perception latency, to learn the PM policy (π_{ϕ}^{PM}).

D. Performance Metrics in Driving

The performance metrics used to assess the three models' driving abilities were taken from [28] and [29].

a) Trajectory Similarity: Using several methods from [28] at a top speed of 90 km/h, we examine the trajectories of all three models on a series of consequence turns, to show driving quality. These methods include partial curve mapping, Frechet distance, area between two curves, curve length, and dynamic time warping. The driving trajectory of BM at a reference speed will be compared to three driving trajectories of BM, PM, and ANEC at a maximum speed of 90 km/h.

b) Track Completion: We evaluate how well each of the three models performed throughout the whole track at various speeds. We utilize the autonomy metric shown in (6) that was inspired by [29] and count the number of times the car leaves the lane and the track.

$$Autonomy = (1 - \frac{W_{inf}P_w + Y_{inf}P_y}{T}) \times 100\%, \qquad (6)$$

where W_{inf} is the number of white lane infringements, P_w is the penalty for W_{inf} in seconds, Y_{inf} is the number of yellow lane infringements, P_y is the penalty for Y_{inf} in seconds, and T is the total travel time in seconds.

IV. Experimental Setup

A. Simulator

We conducted the experiments on OSCAR simulator [30] that is user-friendly and customizable. The OSCAR is based on ROS (Robotic Operating System) [31] integrated with Gazebo multi-robot 3D simulator [32]. ROS Melodic and Gazebo 9 were used. We updated the OSCAR platform to enable the simultaneous operation of several neural network controllers because it was originally designed to support just one neural network controller.

B. Data Collection

We designed a new three-lane track that has several sharp turns in left and right directions and straight road segments, as shown in Fig. 8. Training datasets were collected by a human driver who drove the track at a maximum speed of 65 km/h.

To train our models, we need high-quality data in pairs (road images, steering angles). We collected data using OSCAR, which records an image and all related control information, including steering angle, throttle position, braking pressure, time, velocity, and position. We utilized the Logitech G920 dual-motor feedback driving force racing wheel with pedals and a gear shifter to collect high-quality training data. The total number of training data samples for each model was 7,167.

C. Parameter Tuning

When training the models, δ, shown in Fig. 7, was set to 25. both models were trained using the Adam optimizer [33] with a batch size of 16 and a learning rate of 0.001. To fuse both models, each model will be assigned with a weight based on (4) and (5). The parameters of the equations are set to $\alpha = 0.5$, $\beta = 0.01$, $\gamma = 0.01$, and $\kappa = 20$. Pivot speed V_p was set to 65 km/h. For the performance metric *Autonomy*, we used 3 for P_w and 5 for P_y and 360 for T.

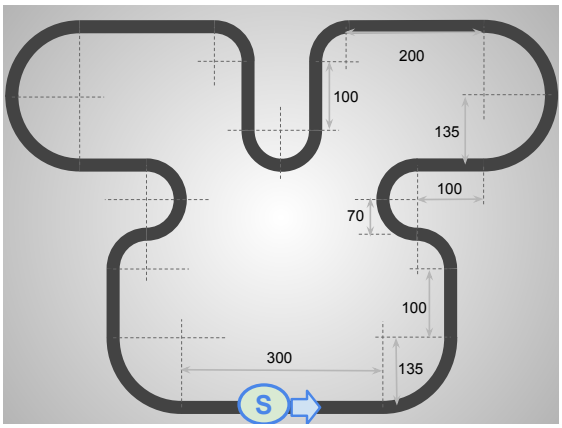

Fig. 8. Driving track. The dimensions of the track are shown, as well as the starting point and the direction of driving.

TABLE I

BM DRIVING PERFORMANCE AT MAXIMUM SPEED OF 65 KM/H AND MAXIMUM SPEED OF 90 KM/H.

Max Speed	**Off Lane**	**Off Track**	**W_{inf}**	**Y_{inf}**	***Autonomy***
65 (ref)	1	0	8	0	93.33 %
90	11	8	13	9	**76.67 %**

D. Machine Learning Framework and Computing Platforms

The OSCAR's neural network modules are designed using Tensorflow and Keras. We used keras 2.2.5 with tensorflow-gpu 1.12.0 on CUDA 9 and cuDNN 7.1.2. All experiments were run on a computer with Intel i7-10700K CPU, 32GB RAM, and an NVIDIA GeForce RTX 2080 8GB GPU. The operating system was Ubuntu 18.04.6.

V. RESULTS

The conducted experiments were designed with the following purposes: 1) To illustrate how the latency effect on driving quality increases when we increase the maximum driving speed. 2) To compare the trajectories of all three models (BM, PM, and ANEC) on a series of consequence turns while driving at a maximum speed of 90 km/h against the trajectory of the reference model while driving at an ideal maximum speed. 3) To compare the driving performance of all three models at different maximum speeds (90, 94, and 97 km/h) to validate that the proposed method can address the algorithmic perception latency issue.

First, to illustrate the latency effect, we had the BM neural network drive a vehicle at a maximum speed of 65 km/h and 90 km/h. The driving performance of BM at a maximum speed of 65 km/h compared to its driving performance at 90 km/h, is shown in Fig. 9 and Table I. The increase in speed amplified the latency effect where the autonomy score significantly decreased from 93.33% to 76.67%.

The reference model will determine how well our proposed driving model is performing. We chose the reference model to be BM driving at a maximum speed of 65 km/h, as this was the maximum speed in the training dataset.

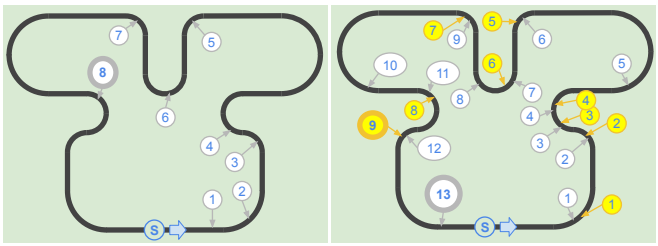

Fig. 9. BM driving performance at a maximum speed of 65 km/h (a) and a maximum speed of 90 km/h (b). A white circle represents a white lane infringement, and a yellow circle represents a yellow lane infringement. The thick border indicates the final value. In (a), the final number of white lane infringements is 8. In (b), the final number of white lane infringements is 13 and the final number of yellow lane infringements is 9.

TABLE II

PERFORMANCE COMPARISON OF ALL THREE MODELS TRAJECTORIES WITH THE REFERENCE BM TRAJECTORY ON A SERIES OF CONSEQUENT TURNS.

	Partial Curve Mapping	Frechet distance	Area between two curves	Curve length	Dynamic Time Warping
BM 65 (ref)	0.000	0.000	0.000	0.000	0.000
BM 90	9.295	10.526	1692.05	3.153	4143.48
PM 90	7.008	5.234	1472.98	1.134	2950.07
ANEC 90	**3.124**	**3.743**	**818.114**	**0.945**	**1313.59**

The trajectories of all driving models at a maximum speed of 90 km/h with the reference model trajectory are illustrated in Fig. 10. It can be said that the trajectory of ANEC is the closest to the reference BM, qualitatively. Table II confirms the visual results and shows how each model performs in contrast to the reference BM model, quantitatively. For all measures shown in the table, the higher the number, the less similar the two trajectories are to one another. The data unambiguously demonstrates that ANEC performed better and was able to tolerate high speed to some extent compared to BM and PM.

TABLE III

PERFORMANCE COMPARISON OF ALL THREE MODELS WITH THE REFERENCE BM USING AUTONOMY METRIC.

Max Speed	**Model**	**Off Lane**	**Off Track**	**W_{inf}**	**Y_{inf}**	***Autonomy* (6)**	**Fig. 11 sub-figure**
65 (ref)	BM	1	0	8	0	***93.33 %***	–
90	BM	11	8	13	9	76.67 %	(a)
90	PM	7	2	12	2	87.22 %	(b)
90	ANEC	**6**	2	8	2	**90.56 %**	(c)
94	BM	10	8	13	8	78.06 %	(d)
94	PM	9	2	11	3	86.67 %	(e)
94	ANEC	**6**	2	10	2	**88.89 %**	(f)
97	BM	11	8	14	8	77.22 %	(g)
97	PM	8	3	11	5	83.88 %	(h)
97	ANEC	**5**	4	9	5	**85.56 %**	(i)

Fig. 11 and Table III compare the performance of all

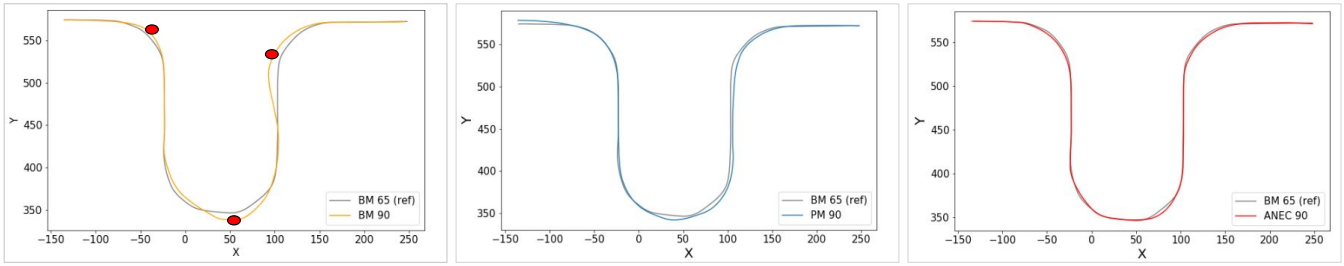

Fig. 10. Visualization of all three models' trajectories with respect to the reference BM trajectory on a series of consequent turns at a maximum speed of 90 km/h. (a) The trajectory of BM, where red circles indicate that the vehicle went off track and required human intervention. (b) The trajectory of PM. (c) The trajectory of ANEC. It can be said that the trajectory of ANEC is the closest to the reference BM, qualitatively.

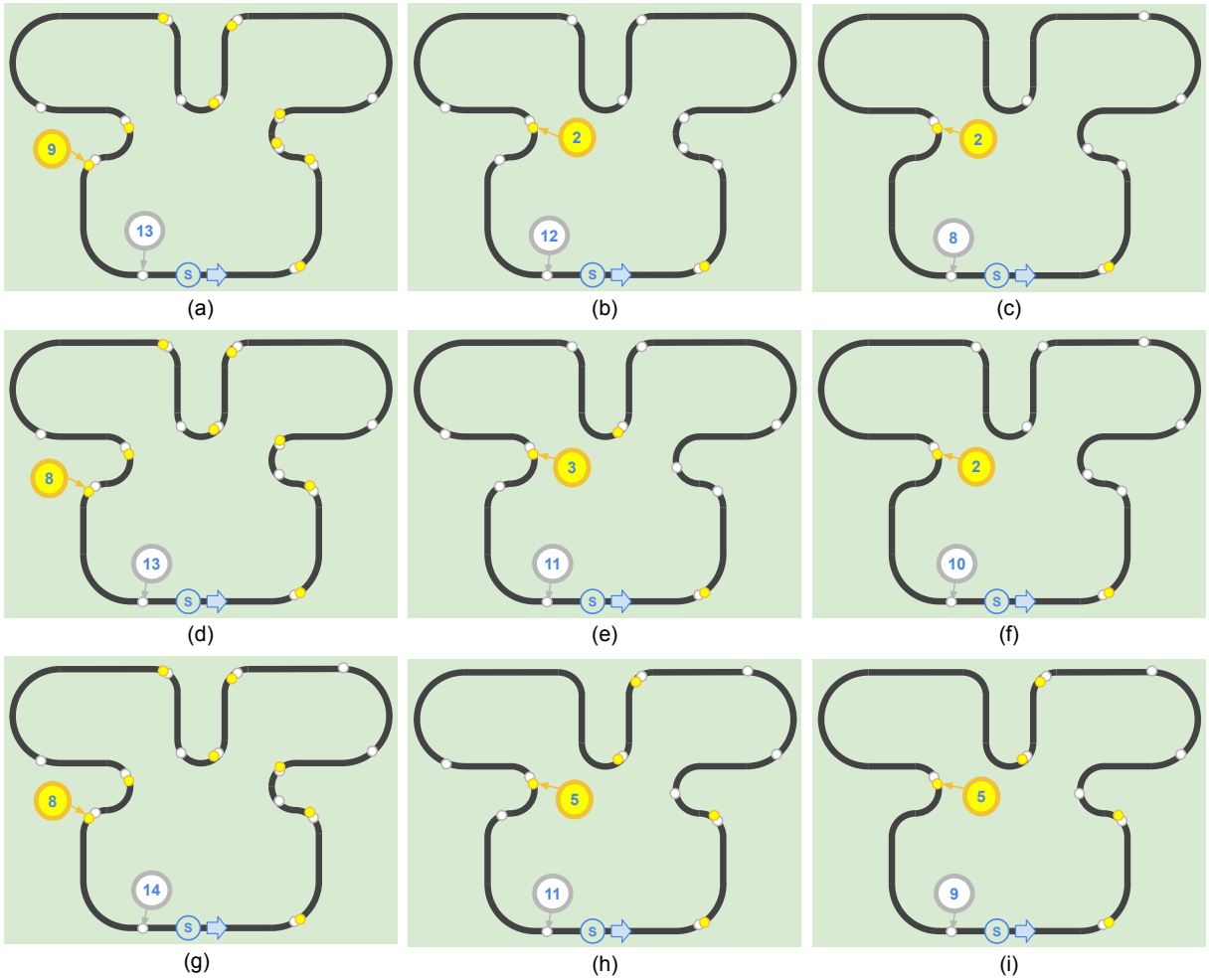

Fig. 11. Performance comparison of all three models with the reference BM using Autonomy metric. A white circle represents a white lane infringement, and a yellow circle represents a yellow lane infringement. The bigger the number of infringements, the higher the penalty, thus a lower *Autonomy* score. ANEC model shows higher Autonomy at 90 km/h, 94 km/h, and 97 km/h compared to BM and PM. (a-c) performance of BM, PM, and ANEC, respectively, at a maximum speed of 90 km/h. (d-f) performance of BM, PM, and ANEC, respectively, at a maximum speed of 94 km/h. (g-i) performance of BM, PM, and ANEC, respectively, at a maximum speed of 97 km/h.

three models using the Autonomy metric in (6). The test was carried out at three distinct top speeds: 90, 94, and 97 km/h. The table demonstrates the ANEC's superiority, as it obtained the greatest Autonomy score at the three different speeds and did not drive off lane as frequently as the other models. When compared to ANEC, the PM performed marginally worse. The BM, on the other hand, performed badly at high speeds.

VI. Conclusion and Future Work

In this study, we introduced a novel adaptive neural lateral controller, ANEC, for autonomous driving to solve

a problem that most studies overlook: algorithmic latency. Our proposed ANEC is inspired by human drivers' near/far gaze distribution during lane-keeping, and it ensembles two driving models: BM and PM, as illustrated in Fig. 5. Note that near/far areas are not explicitly selected. BM and PM will figure out which areas are important to infer control output. While BM is expected to cover the near point of observation, PM is expected to scan the road ahead to extract latent variables for future actions. PM is adaptively combined with BM using the current speed of the vehicle using ANEC. As a result, ANEC can overcome the algorithmic latency issue so that neural controllers can maintain the quality of driving.

Future work will involve expanding our dataset and incorporating additional test tracks to further explore and validate ANEC's capabilities in various new environments and scenarios. In addition, different driving model architectures will be used to demonstrate that the concept of ANEC remains independent of the network architecture. Finally, we aim to optimize the weight adaptation function and explore various alternative approaches to enhance ANEC's overall performance.

REFERENCES

[1] Tessel Blom, Daniel Feuerriegel, Philippa Johnson, Stefan Bode, and Hinze Hogendoorn. Predictions drive neural representations of visual events ahead of incoming sensory information. *Proceedings of the National Academy of Sciences*, 117(13):7510–7515, 2020.

[2] Daniel Feuerriegel, Tessel Blom, and Hinze Hogendoorn. Predictive activation of sensory representations as a source of evidence in perceptual decision-making. *Cortex*, 136:140–146, 2021.

[3] Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In *European Conference on Computer Vision*, pages 473–488. Springer, 2020.

[4] Dean Pomerleau. ALVINN: An Autonomous Land Vehicle In a Neural Network. In *Proceedings of Advances in Neural Information Processing Systems 1*, pages 305–313. Morgan Kaufmann, 1989.

[5] Net-Scale. Autonomous Off-Road Vehicle Control Using End-to-End Learning, July 2004.

[6] Z. Chen and X. Huang. End-to-end learning for lane keeping of self-driving cars. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1856–1860, June 2017.

[7] Mariusz Bojarski, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel, and Urs Muller. Explaining how a deep neural network trained with end-to-end learning steers a car, 2017.

[8] Qing Wang, Long Chen, Bin Tian, Wei Tian, Lingxi Li, and Dongpu Cao. End-to-End Autonomous Driving: An Angle Branched Network Approach. *IEEE Transactions on Vehicular Technology*, 68(12):11599–11610, December 2019. Conference Name: IEEE Transactions on Vehicular Technology.

[9] Tianhao Wu, Ao Luo, Rui Huang, Hong Cheng, and Yang Zhao. End-to-End Driving Model for Steering Control of Autonomous Vehicles with Future Spatiotemporal Features. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 950–955, November 2019. ISSN: 2153-0866.

[10] Jaerock Kwon, Aws Khalil, Donghyun Kim, and Haewoon Nam. Incremental end-to-end learning for lateral control in autonomous driving. *IEEE Access*, 10:33771–33786, 2022.

[11] Trent Weiss and Madhur Behl. Deepracing: Parameterized trajectories for autonomous racing. *arXiv preprint arXiv:2005.05178*, 2020.

[12] Dario D Salvucci. Modeling driver behavior in a cognitive architecture. *Human factors*, 48(2):362–380, 2006.

[13] Karl Berntorp, Tru Hoang, Rien Quirynen, and Stefano Di Cairano. Control architecture design for autonomous vehicles. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pages 404–411. IEEE, 2018.

[14] Hyoeun Lee, Youngjoon Choi, Taeho Han, and Kanghee Kim. Probabilistically guaranteeing end-to-end latencies in autonomous vehicle computing systems. *IEEE Transactions on Computers*, 2022.

[15] Kieran Strobel, Sibo Zhu, Raphael Chang, and Skanda Koppula. Accurate, low-latency visual perception for autonomous racing: Challenges, mechanisms, and practical solutions. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1969–1975. IEEE, 2020.

[16] Benjamin Vedder, Jonny Vinter, and Magnus Jonsson. A low-cost model vehicle testbed with accurate positioning for autonomous driving. *Journal of Robotics*, 2018, 2018.

[17] Xiaohu Ge. Ultra-reliable low-latency communications in autonomous vehicular networks. *IEEE Transactions on Vehicular Technology*, 68(5):5005–5016, 2019.

[18] Oussama El Marai and Tarik Taleb. Smooth and low latency video streaming for autonomous cars during handover. *Ieee Network*, 34(6):302–309, 2020.

[19] Shiva Raj Pokhrel, Neeraj Kumar, and Anwar Walid. Towards ultra reliable low latency multipath tcp for connected autonomous vehicles. *IEEE Transactions on Vehicular Technology*, 70(8):8175–8185, 2021.

[20] Qingyang Zhang, Hong Zhong, Jie Cui, Lingmei Ren, and Weisong Shi. Ac4av: a flexible and dynamic access control framework for connected and autonomous vehicles. *IEEE Internet of Things Journal*, 8(3):1946–1958, 2020.

[21] David J Gorsich, Paramsothy Jayakumar, Michael P Cole, Cory M Crean, Abhinandan Jain, and Tulga Ersal. Evaluating mobility vs. latency in unmanned ground vehicles. *Journal of Terramechanics*, 80:11–19, 2018.

[22] Ibrar Yaqoob, Latif U Khan, SM Ahsan Kazmi, Muhammad Imran, Nadra Guizani, and Choong Seon Hong. Autonomous driving cars in smart cities: Recent advances, requirements, and challenges. *IEEE Network*, 34(1):174–181, 2019.

[23] Shaobing Xu, Huei Peng, and Yifan Tang. Preview path tracking control with delay compensation for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 22(5):2979–2989, 2020.

[24] Viktor Tihanyi, András Rövid, Viktor Remeli, Zsolt Vincze, Mihály Csonthó, Zsombor Pethő, Mátyás Szalai, Balázs Varga, Aws Khalil, and Zsolt Szalay. Towards cooperative perception services for its: Digital twin in the automotive edge cloud. *Energies*, 14(18):5930, 2021.

[25] Huizi Mao, Xiaodong Yang, and William J Dally. A delay metric for video object detection: What average precision fails to tell. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 573–582, 2019.

[26] Jelena Kocić, Nenad Jovičić, and Vujo Drndarević. An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms. *Sensors (Basel, Switzerland)*, 19(9):2064, May 2019.

[27] Penghao Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. *arXiv preprint arXiv:2206.08129*, 2022.

[28] Charles F Jekel, Gerhard Venter, Martin P Venter, Nielen Stander, and Raphael T Haftka. Similarity measures for identifying material parameters from hysteresis loops using inverse analysis. *International Journal of Material Forming*, 12(3):355–378, 2019.

[29] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to End Learning for Self-Driving Cars. *arXiv:1604.07316 [cs]*, April 2016. arXiv: 1604.07316.

[30] Jaerock Kwon. OSCAR (Open-Source robotic Car Architecture for Research and education), February 2021. original-date: 2020-11-29T19:48:57Z.

[31] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[32] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE.

[33] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.