Tuning the feedback controller gains is a simple way to improve autonomous driving performance

Wenyu Liang, Pablo R. Baldivieso and Ross Drummond
Dept. of Automatic Control and Systems Engineering
University of Sheffield
Sheffield, UK
{w.liang,p.baldivieso, ross.drummond}@sheffield.ac.uk

Donghwan Shin
Dept. of Computer Science
University of Sheffield
Sheffield, UK
d.shin@sheffield.ac.uk

Abstract—Typical autonomous driving systems are a combination of machine learning algorithms (often involving neural networks) and classical feedback controllers. Whilst significant progress has been made in recent years on the neural network side of these systems, only limited progress has been made on the feedback controller side. Often, the feedback control gains are simply passed from paper to paper with little re-tuning taking place, even though the changes to the neural networks can alter the vehicle's closed loop dynamics. The aim of this paper is to highlight the limitations of this approach; it is shown that re-tuning the feedback controller can be a simple way to improve autonomous driving performance. To demonstrate this, the PID gains of the longitudinal controller in the TCP autonomous vehicle algorithm are tuned. This causes the driving score in CARLA to increase from 73.21 to 77.38, with the results averaged over 16 driving scenarios. Moreover, it was observed that the performance benefits were most apparent during challenging driving scenarios, such as during rain or night time, as the tuned controller led to a more assertive driving style. These results demonstrate the value of developing both the neural network and feedback control policies of autonomous driving systems *simultaneously*, as this can be a simple and methodical way to improve autonomous driving system performance and robustness.

Index Terms—Autonomous driving systems, feedback control, neural networks

I. INTRODUCTION

Training an Autonomous Driving System (ADS) requires solving a complex control problem so the autonomous vehicle can efficiently navigate through uncertain and changing environments. As a control problem, the development of an ADS is complicated by issues such as: *i)* the sensors (such as image data from cameras) are only able to infer local and incomplete state information, *ii)* the lack of clear definitions for the control objectives and constraints to capture the diverse range of driving behaviours seen in practice, and *iii)* the ADS has to react to agents moving dynamically and, often, aggressively. These issues represent a challenge that traditional control policies, such as model predictive control (MPC) [12], struggle to handle. Instead, machine learning algorithms built from neural networks (NN) dominate the landscape of ADS. These data-driven NN algorithms are able to flexibly handle both the

wealth of data generated by the vehicle sensors as well as the vagueness of the control problem definition. They can now also be trained using readily-available and simple software which do not require mathematical models, with this simplicity being another key reason fuelling the growth of NNs. Spurned on by the growth of NN control algorithms, ADSs have improved significantly in recent years, and there now exists an array of open source software tools and challenges to evaluate different algorithms in benchmark scenarios. One notable example is the open source CARLA driving simulator [6] that hosts the CARLA Autonomous Driving Leaderboard [20]. CARLA is used in this paper to evaluate the ADS algorithms.

Currently, the algorithms dominating the CARLA Leaderboard are those based on NNs that translate the vehicle's image/lidar data into control actions for steering, throttle and braking. Examples include: Trajectory-guided control prediction (TCP) [19], ReasonNet [15], Interfuser [14], and Pylot [10], with CARLA also being used as a resource for software testing ADSs [11]. However, while such NN-dominated ADS control algorithms have performed well in experiments, their limitations affect how well they perform in the field. Most significant are the issues around robustness and explainability, which are, coincidentally, two issues which classical control algorithms can, arguably, deal with, at least in principle. However, the rate of progress of NN-based ADS algorithms has meant that, often, there is not enough time/data to follow the classical control design pipeline of developing a mathematical model, tuning the controller, and then verifying its robustness *e.g.* using Lyapunov functions. Instead, different NN architectures are proposed and trained using the latest techniques where only limited stability analysis/modelling/testing is conducted. As such, to keep pace with the latest NN developments, often the control gains used to manipulate the vehicle towards the reference trajectory generated by the NN are simply passed from paper to paper without re-tuning. An example of this is the Proportional-Integral-Derivative (PID) control gains of [19] which were carried over from [4], as the tuning from [4] was deemed satisfactory. From a control perspective, this practice of carrying over PID gains is problematic. Changes to the NN in the ADS can impact the vehicle's closed-loop dynamics, and this change should be accounted for in the controller gains.

The aim of this paper is to demonstrate the value of tuning the feedback controller gains in ADS which use NNs to generate the vehicle's reference trajectories. Whilst rapid advances have been for the NNs of ADS, often little attention is paid to the re-tuning of the controller gains. It is shown here that re-tuning these gains can be a simple way to improve ADS performance. And, since the tuning can be done using the methods and machinery of control theory, the resulting improvements can be implemented at scale and in a methodical way, potentially with robustness guarantees as well [7]. In this way, the results of this paper are aimed at opening a dialogue between the control theory and machine learning methods behind ADSs. Specifically, our work on re-tuning ADS controllers have led to the following results:

1) Re-tuning the PID gains in the TCP [19] autonomous driving systems algorithm can improve performance.
2) The results are evaluated in CARLA, with the driving score of 73.21 for TCP increased to 77.38 by simply tuning the gains of its longitudinal PID controller.
3) The performance improvements of the tuned controller were most apparent in low-light and wet scenarios.

These results are preliminary and further improvements are expected through the application of more advanced tuning methods, such as automatic tuning [1] or designing the controller to avoid overshooting [17]. Given the promising results obtained, we expect that employing more advanced control methods for ADS will lead to further improvements in the driving score. In particular, there is great promise in the application of MPC for this problem, to take advantage of the future way points generated by NNs planning algorithms such as TCP. Combining those future way points with the robustness and constraint satisfaction feature of MPC should deliver further improvements in the driving score.

In terms of comparison to the literature, there is a wealth of results and methods in the area of ADS, as reviewed in papers such as [3], [20] In this work, the focus is on improving the driving score of the TCP ADS described in [19] for CARLA [6]. The essence of our result lies in showing how tuning the feedback controller gains, here the focus is on the the longitudinal PID controller's gains, in an ADS which uses NNs to generate reference trajectories can improve performance. As mentioned above, many of the emerging NN-based algorithms do not re-tune the feedback controller gains during development, an issue which could deteriorate performance and/or increase training time. Our aim is to highlight this limitation and recommend that a dialogue should be opened between machine learners and control engineers to remedy it, as doing so can bring quick and simple gains in the driving score. We focus on the TCP algorithm for ADS in this work, but we are conscious that the idea of the need for controller re-tuning should be applicable to a broad class of ADS algorithms. Moreover, algorithms for vehicle control have also been implemented from the pure control theory perspective, for example the eco-driving results of [16], the robust MPC approach developed in [18] or the stochastic MPC approach of

[13]. The difference with these more control-focused results, such as [18], is that whilst they are often equipped with performance and optimality guarantees, a feature lacking for many NN algorithms, e.g. [10] and [19], these algorithms often require full state/road information which may not always be available

II. CARLA DRIVING SIMULATOR

The results of this paper are generated with the CARLA driving simulator, with version CARLA 0.9.10 run on Linux. CARLA is as an open-source driving simulator based upon Unreal Engine 4 able to create a diverse range of driving scenarios for training and validating ADS. These scenarios include different weather settings, city infrastructure, traffic density, and other such dangerous driving scenarios that are impractical to replicate within the real physical world. Environments in CARLA consist of 3D representations of both static, *i.e.,* buildings, vegetation, infrastructure, or dynamic elements, *i.e.,* moving traffic or pedestrians. Additionally, CARLA offers an array of sensors for collecting data, such as RGB cameras, LiDAR, radar, GPS, IMU, speedometers, and pseudo-sensors that provide semantic segmentation and ground-truth depth information [6].

CARLA also hosts the Autonomous Driving Challenge Leaderboard to assess the driving capabilities of autonomous agents within real traffic scenarios. Throughout a series of predetermined routes, the ego vehicle encounters a diverse array of traffic situations, including lane merging, lane changing, interactions at crossroads and roundabouts, responses to traffic lights and signs, and interactions with bicycles and pedestrians.

A. Driving score

Given an autonomous vehicle, the performance of its ADS can be evaluated in CARLA using the Driving Score of the CARLA Driving Challenge. To generate the score, the ADS undergoes evaluation across 16 route scenarios. Each route scenario is repeated three times, and the results for these three routes are then averaged. When an infraction occurs during a run, the penalty coefficients of Table I are applied. At the culmination of the route, regardless of whether the ego vehicle has reached its destination or not, the total penalty score is determined by multiplying all the penalty coefficients together. A penalty score of 1 is registered if no infractions occurs.

TABLE I: Penalty coefficient for the infractions.

Infraction	Coefficient
Collision with pedestrian	0.5
Collision with pedestrian	0.6
Collision with pedestrian	0.65
Running a red light	0.7
Running a stop sign	0.8
Off-road driving	1-Percentage of the completed route

Furthermore, if the agent's progress is halted, then the simulation has to be suspended as detailed in Table II. In such instances, the current score at the time of the halting is recorded and the evaluation proceeds to the next scenario.

TABLE II: Shut down events

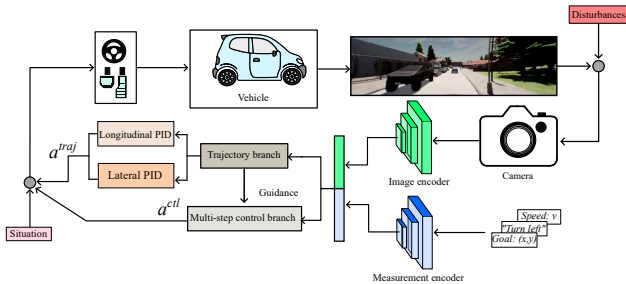| Shut down event	Explanation
Route deviation	More than 30m deviation from the planned route
Agent blocked	Agent no action for 180s
Simulation timeout	No client-server communication after 60s
Route timeout	Simulation route takes too long to finish

Fig. 1: Schematic of the TCP algorithm from [19].

The ADS's **Driving score** is the route completion score multiplied by the total penalty score, where

1) **Total penalty score**: Obtained by multiplying all the penalty coefficients together.
2) **Route completion** is the percentage of the route's length that an agent has completed. A successful arrival at the destination corresponds to a perfect score of 100.

After completing all the routes, an overall infraction score is derived by normalizing against the infraction count incurred per kilometre travelled. This is achieved by subtracting the average of all penalties from 1. Similarly, the global route completion score is the mean for all routes, and the global driving score is the global infraction score multiplied by the global route completion score.

III. Autonomous Driving Systems: TCP

In this work, the TCP algorithm from [19] is used as the benchmark ADS. The structure of TCP is shown in Figure 1 and it splits into two different branches, a **Trajectory branch** and a **multi-step control prediction branch**. This combination contrasts with most end-to-end driving systems, e.g. [2], which typically either exclusively map raw sensor data to trajectories *or* low-level control actions.

A. Trajectory planning branch

Within TCP's trajectory planning branch, images captured by the camera are processed by a CNN-based image encoder to generate a feature map. Simultaneously, measurement data, comprising information such as location, speed, and navigation is processed by a multi-layer NN measurement encoder to generate measurement features. These features are then amalgamated via average pooling, forming a trajectory feature that enters a Gated Recurrent Unit (GRU) [5] to predict four waypoints for the vehicle to follow.

TABLE III: Values for the longitudinal PID control gains of both the original and tuned versions of TCP [19]. Here, k_p is the proportional gain, k_i is the integral gain and k_d is the derivative gain.

	k_p	k_i	k_d	Max throttle	Brake speed
TCP	5	0.5	1	0.75	0.4
TCP-tuned	11	0.1	1	0.8	0.45

These waypoints form a reference trajectory for the autonomous vehicle. PID controllers for both the longitudinal and lateral direction are then used to regulate the autonomous vehicle and follow this reference. The longitudinal controller is used to generate throttle and brake control actions whilst the lateral controller is used to implement steering action. In this work, the focus will be on tuning the longitudinal controller, with the gains of the lateral controller kept the same as in [19]. The values of the longitudinal PID gains are stated in Table III with k_p being the proportional gain, k_d the derivative gain and k_i the integral gain.

B. Multi-step control branch

In the multi-step control prediction branch of TCP, the goal is to directly forecast control actions for the ensuing four time steps. Within this branch, an embedded temporal module is used to implicitly account for the dynamic interactions and alterations within the environment and the agent. This module aims to predict how objects move within the environment and relative to the ego vehicle, and so anticipate dynamic changes.

C. Integrating the branches

The control actions from both the trajectory branch and the multi-step control branch of TCP [19] are then combined together using a situation-based fusion strategy. Specifically, a situation is determined by calculating the steering actions over the previous seconds. If more than half of these steering actions surpass 0.1, then the vehicle is assumed to be turning and a "*control specialized*" situation is regarded as being active. Conversely, if this criterion is not met, then the situation is deemed "*trajectory specialized*." A weight, $\alpha \in [0, 0.5]$, is used to combine the two control actions from the trajectory branch, $a^{\text{traj}}[k]$, and the multi-step control prediction branch, $a^{\text{ctl}}[k]$, to generate the final control action. When the vehicle is in a control specialised state, the combined action, $a[k]$, is

$$a[k] = \alpha \times a^{\text{ctl}}[k] + (1 - \alpha) \times a^{\text{traj}}[k]$$

and, when it is trajectory specialised, then the two control actions are combined according to

$$a[k] = \alpha \times a^{\text{traj}}[k] + (1 - \alpha) \times a^{\text{ctl}}[k].$$

In this paper, the weight was set to $\alpha = 1/2$.

D. Tuning the Feedback control algorithm

The values of the longitudinal PID gains from TCP [19] and the tuned values considered in this paper are shown in Table III—the tuned gains are referred to as TCP-tuned.
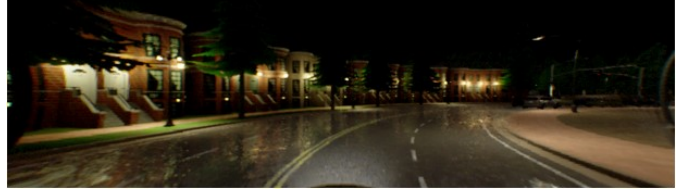
(a) *Clear Noon* in *Town02*.

(b) *Cloudy Sunset* in *Town02*.

(c) *Soft Rain Dawn* in *Town05*.

(d) *Hard Rain Night* in *Town05*.

Fig. 2: Images from the drivers perspective of the vehicle in four different weather scenarios.
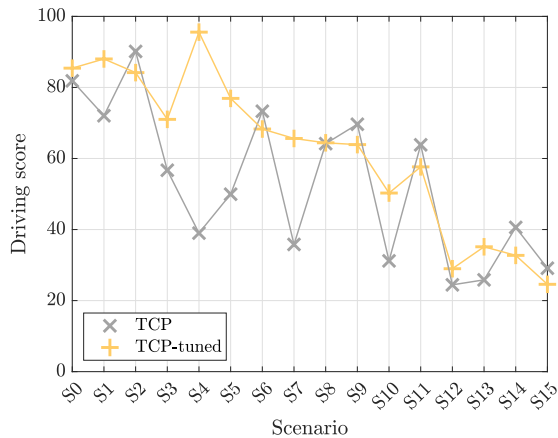

Fig. 3: Comparison between the original driving scores for the original TCP algorithm [19] and that with the tuned PID gains for the longitudinal controller, referred to as TCP-tuned.

No other additions were made to TCP beside these changes to the longitudinal PID gains. Simple trial-and-error tuning was applied to tune the gains, with it being observed that a reduction in the integral gain k_i and an increase in the proportional gain k_p led to a more reactive and dynamic driving resposne. It is acknowledged that this is rudimentary tuning compared to, for instance, the methods detailed in [1]; further improvements over the results of Table IV are expected using more effective tuning algorithms, and even more advanced control algorithms such as MPC. However, the results of this paper demonstrate that even rudimentary tuning of the PID gains can improve the results, as will be seen in the following section.

IV. RESULTS

This section compares the CARLA driving score results of the original TCP algorithm of [19] against the version with the tuned PID gains. Again, this tuned TCP algorithm is referred to as TCP-tuned in the results. The score is based upon the

TABLE IV: Penalty coefficient of the infractions for both TCP and the version with the tuned PID gains used in this work.

	Driving score	Route completion	Infraction penalty
TCP	73.21	85.63	0.855
TCP-tuned	77.38	89.36	0.866

analysis of 4 different routes, 2 from *Town02* and 2 from *Town05*. *Town02* features T-junctions in a small town setting, while *Town05* represents a larger city with multiple traffic lanes. Both towns are provided in CARLA by default. Each route is assessed under four distinct weather conditions (*Clear Noon*, *Cloudy Sunset*, *Soft Rain Dawn*, *Hard Rain Night*). Figure 2 provides a visual representation of different weather conditions and town scenarios as viewed by the driver.

The combination of the four different routes with the four different weather events resulted in a total of 16 different route scenarios being tested. These sixteen scenarios are labelled $S0-S15$ and the results comparing the original TCP algorithm with the tuned version is shown in Figure 3. In each scenario, the traffic participants were randomly generated.

A. Driving Score

Table IV compares the driving score for the original TCP and that with the tuned longitudinal PID gains from Table III. The results show that the tuned version of TCP can exhibit superior performance when evaluated across all sixteen driving scenarios, with the PID tuning increasing the Driving Score from 73.21 to 77.38. It was noted that that TCP suffered particularly in the challenging driving scenarios of *Hard Rain Night*, with the reduced visibility and the ground reflections impacting the camera sensors and, correspondingly, TCP's ability to maintain a precise lane position for the vehicle. It was observed that the more assertive driving of the tuned TCP algorithm (with a higher proportional PID gain and a lower integral one) was able to alleviate some of the hesitancy of
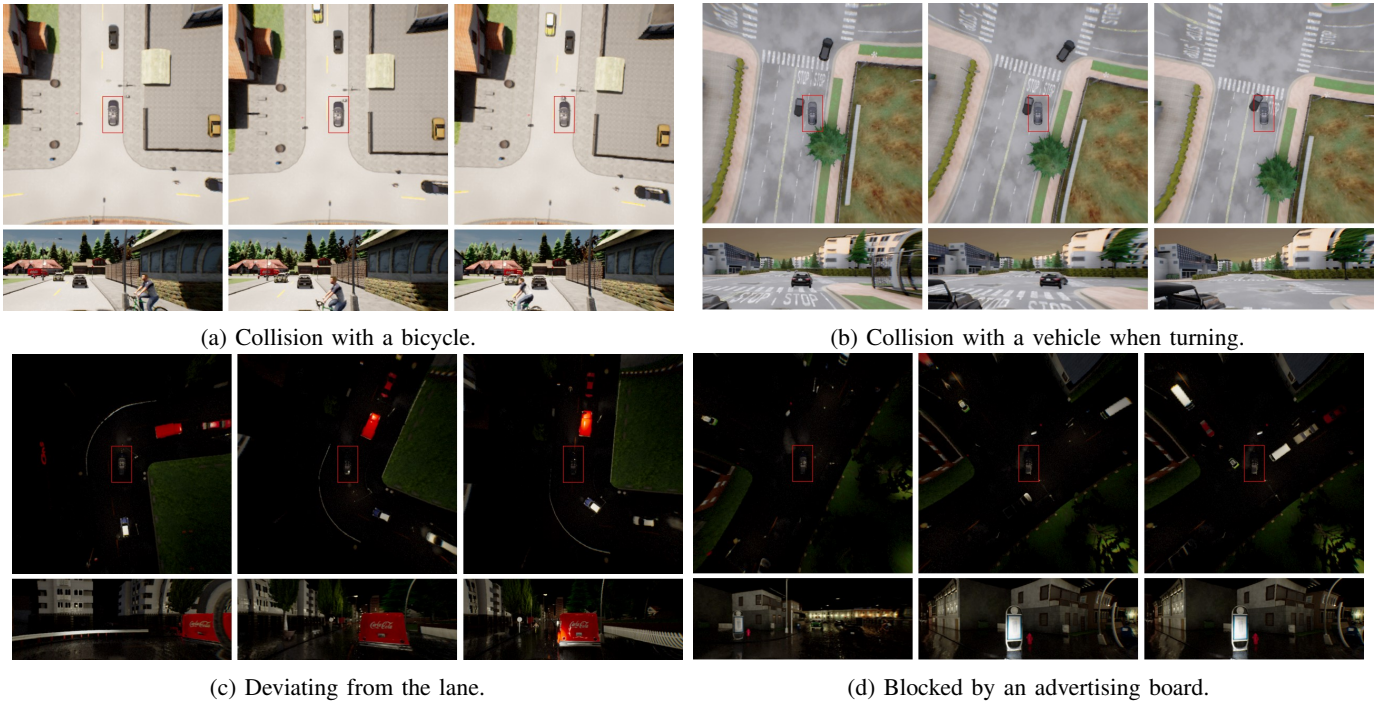
(a) Collision with a bicycle.

(b) Collision with a vehicle when turning.

(c) Deviating from the lane.

(d) Blocked by an advertising board.

Fig. 4: Four examples of crashes recorded with the TCP [19] algorithm in CARLA (ego vehicle highlighted by red box).

(a) Collision avoided with vehicle at intersection.

(b) Collision avoided with bicycle.

Fig. 5: Instances of the TCP algorithm with tuned PID gains avoiding crash events. In both cases, the ego vehicle is highlighted by a red bounding box. The bicycle in b) is highlighted by a blue bounding box.

the driver in these challenging driving scenarios, and hence improve the score. To reduce the risk of this assertive driving style leading to more crashes, the brake ratio of TCP [19] was increased to force the vehicle to initiate braking earlier.

The line chart of Figure 3 compares the driving scores for both the original version of TCP [19] against the one with the tuned PID gains. The figure demonstrates that the tuned controller can overcome some of the challenging scenarios, such as avoiding the collision with the bicycle in Scenario 4 and passing the blocking point in Scenario 7, resulting in a higher driving score in these scenarios.

B. Inspection of crash incidents

Tuning the feedback PID gains in the TCP not only increased the overall driving score but also meant that certain crash situations could be avoided. Figures 4 shows several

scenarios of the autonomous vehicle crashing in CARLA when controlled by the untuned TCP algorithm. These examples include the vehicle colliding with a bicycle which unexpectedly crossed the road, crashing into a vehicle when turning around a corner, leaving its lane when turning and stopping for some unknown reason when encountering an advertisement board. In particular, it was observed that the vehicle stopped at the advertisement board on all three repetitions of the route, which was on a rainy night. Videos showing simulations with TCP can be found in [8] Specifically, the files `Original_S7_1.mp4`, `Original_S7_2.mp4` and `Original_S7_3.mp4` show the TCP-driven vehicle being blocked twice at an advertising board and exceeding the allotted time once.

Often these collision incidents could be avoided with the tuned version of TCP. For instance, the collision observed

in Scenario 10 of the vehicle crashing when turning round a corner, as shown in Figure 4b, was avoided after tuning the PID gains, as illustrated in Figure 5a. In that instance, the ego vehicle maintained some distance from the adjacent vehicle and the stop line, allowing it to navigate through without colliding. The ego vehicle with the tuned gains also demonstrated an ability to evade bicycles crossing the road unexpectedly, even in challenging conditions like heavy rain at night. An example of this is shown in Figure 5b where, despite the bicycle being barely discernible to human eyes, the ego vehicle halted in a timely manner, avoiding a collision. Videos of the vehicle driven by TCP with the tuned PID gains can be found in [9]. In this case, the collision in Scenario 10 from TCP shown in Figure 4b is avoided, as shown in Figure 5a. The video of the vehicle evading the bicycle, as shown in Figure 5b, can be found in the file `TCP3_S10_2.mp4`.

CONCLUSIONS

It was shown that simply re-tuning the feedback control gains embedded within neural-network based autonomous driving systems can improve performance, especially for challenging driving scenarios such as when it is raining and during night time. To demonstrate this result, the PID gains for the longitudinal control of the TCP algorithm were tuned and simulations were conducted in `CARLA` to evaluate performance. After tuning the PID gains, the driving score over 16 scenarios was increased from 73.21 to 77.38– demonstrating that tuning the control gains can be a simple way to improve driving performance. The aim of this work was to highlight that while there has been rapid advancement in neural network design for autonomous driving systems, there is value in spending time and effort in tuning the feedback control gains as well. We hope that this result can encourage a dialogue between machine learners and control engineers to create more robust and explainable driving systems which retain the flexibility and power of the underlying machine learning algorithms.

We are conscious that the presented results are rudimentary (manual tuning of a PID controller is studied) and we see significant opportunity to develop these results further. Specifically, we see potential in applying more advanced tuning algorithms and control policies (in particular MPC since the future way points generated by TCP can be embedded within the MPC cost). Moreover, we believe that these results would benefit from the certificates of neural network robustness. However, whilst guaranteeing closed-loop stability of the system is an appealing objective from a control theory point of view, it was observed that most of the crashes encountered in the `CARLA` simulations could be attributed to poor trajectories being generated (i.e. pushing the vehicle towards a crash/leaving a lane) rather than some underlying instability of the closed-loop feedback system. Still, by applying the deep mathematical toolbox of control theory, we expect significant benefits in terms of robustness and explainability could be achieved.

REFERENCES

[1] K. J. Åström and T. Hägglund, *Advanced PID control*. ISA-The Instrumentation, Systems and Automation Society, 2006.

[2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[3] S. Chen and H. Chen, "MPC-based path tracking with PID speed control for autonomous vehicles," *IOP Conference Series: Materials Science and Engineering*, vol. 892, p. 012034, 7 2020.

[4] K. Chitta, A. Prakash, and A. Geiger, "Neat: Neural attention fields for end-to-end autonomous driving," in *Procs. of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 793–15 803.

[5] K. Cho, B. V. Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," *Procs. of the Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, 2014.

[6] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator." PMLR, 10 2017, pp. 1–16.

[7] R. Drummond, S. Duncan, M. Turner, P. Pauli, and F. Allgower, "Bounding the difference between model predictive control and neural networks," in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 817–829.

[8] R. Drummond, W. Liang, P. R. B. Monasterios, and D. Shin, "CARLA simulation videos with the TCP autonomous driving system," 11 2023. [Online]. Available: https://figshare.com/articles/media/CARLA_simulations_of_autonomous_driving_system_using_TCP_https_github_com_OpenDriveLab_TCP/24619512

[9] ——, "CARLA simulation videos with the tuned TCP autonomous driving system," 11 2023. [Online]. Available: https://figshare.com/articles/media/TCP-tuned/24619599

[10] I. Gog, S. Kalra, P. Schafhalter, M. A. Wright, J. E. Gonzalez, and I. Stoica, "Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous vehicles," in *Procs. of the International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8806–8813.

[11] F. U. Haq, D. Shin, and L. Briand, "Efficient online testing for DNN-enabled systems using surrogate-assisted and many-objective optimization," in *Procs. of the International Conference on Software Engineering*, 2022, pp. 811–822.

[12] B. Kouvaritakis and M. Cannon, "Model predictive control," *Switzerland: Springer International Publishing*, vol. 38, 2016.

[13] S. H. Nair, H. Lee, E. Joa, Y. Wang, H. E. Tseng, and F. Borrelli, "Predictive control for autonomous driving with uncertain, multi-modal predictions," *arXiv preprint arXiv:2310.20561*, 2023.

[14] 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*. PMLR, 2023, pp. 726–737.

[15] 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 *Procs. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 723–13 733.

[16] C. Sun, J. Guanetti, F. Borrelli, and S. J. Moura, "Optimal eco-driving control of connected and autonomous vehicles through signalized intersections," *IEEE Internet of Things Journal*, vol. 7, pp. 3759–3773, 5 2020.

[17] H. Taghavian, R. Drummond, and M. Johansson, "Pole-placement for non-overshooting reference tracking," in *Procs. of the Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 414–421.

[18] B. A. H. Vicente, P. A. Trodden, and S. R. Anderson, "Fast tube model predictive control for driverless cars using linear data-driven models," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 3, pp. 1395–1410, 2022.

[19] 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.

[20] Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu, and A. M. Lopez, "Multimodal end-to-end autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 537–547, 1 2022.