Encoding Consistency: Optimizing Self-Driving Reliability With Real-Time Speed Data

William Fowler
Tufts University
Medford, MA, USA
william.fowler@tufts.edu

Kate Keahey
Argonne National Laboratory
Lemont, IL, USA
keahey@mcs.anl.gov

Alicia Esquivel Morel
University of Missouri
Columbia, MO, USA
ace6qv@mail.missouri.edu

Abstract

Self-driving cars can revolutionize transportation systems, offering the potential to significantly enhance efficiency while also addressing the critical issue of human fatalities on roadways. Hence, there is a need to investigate methods to enhance self-driving technologies through end-to-end learning techniques. In this paper, we investigate methodologies that integrate Convolutional Neural Networks (CNNs) to enhance self-driving consistency through real-time velocity and steering estimation. We extend an end-to-end state-of-the-art learning solution with real-time speed data as additional model input to refine reliability. Specifically, our work integrates an optical encoder sensor system to record car speed during training data collection, ensuring the throttle can be regulated during model inference. An end-to-end experimental testbed is deployed on the Chameleon cloud using CHI@Edge infrastructure to manage a 1:18 scaled car, equipped with a Raspberry Pi as its onboard computer. Finally, we provide guidance that facilitates reproducibility and highlight the challenges and limitations of supporting such experiments.

CCS Concepts: • **Computer systems organization** → **Robotic autonomy**.

Keywords: Autonomous driving, optimization, machine learning

ACM Reference Format:
William Fowler, Kate Keahey, and Alicia Esquivel Morel. 2024. Encoding Consistency: Optimizing Self-Driving Reliability With Real-Time Speed Data. In *Workshop on Flexible Resource and Application Management on the Edge (FRAME '24), June 3–7, 2024, Pisa, Italy*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3659994.3660308

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of the United States government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only. Request permissions from owner/author(s).

FRAME '24, June 3–7, 2024, Pisa, Italy

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0641-7/24/06
https://doi.org/10.1145/3659994.3660308

1 Introduction

Self-driving cars can potentially improve transportation efficiency and reduce human fatalities – provided they have access to significant processing power and large amounts of data [11]. A current limitation of end-to-end learning is that models are often trained to only predict steering while maintaining constant velocity. This limitation can be avoided

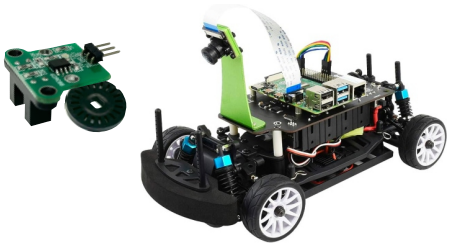


Figure 1. Speed Measuring Sensor Module used as optical encoder sensor and Pi-Racer, embedded with a Raspberry Pi 4 and camera.

by using an optical encoder sensor to observe car speed by tracking wheel revolutions and then controlling speed by regulating throttle and power sent to the motor. However, a speed-predicting end-to-end model has only been tested in simulation [12]. The Donkey Car self-driving project [7] utilizes a CNN to predict steering and throttle to drive a scale model car using human-collected data. Predicting a throttle percentage is problematic, as the power sent to the motor will have a varying effect on the car's speed depending on the road environment and the motor's capabilities. Thus, a key question driving this research is whether using end-to-end learning for current speed prediction will result in fewer errors and increased accuracy on varying surface conditions.

In this paper, we replicated and extended a similar end-to-end autonomous driving CNN solution [1], albeit at a smaller scale, with the goal to enhance self-driving consistency through real-time velocity and steering estimation. We introduce additional model input to refine reliability through an optical encoder sensor, specifically designed to record car speed during training data collection and to regulate throttle during model inference. **Figure 1** illustrates a speed measuring sensor module which we used as optical encoder sensor embedded with a Raspberry Pi 4 and camera within the PI-Racer [3] car kit. When configuring a small-scale car with

multiple sensors, optical wheel encoders were shown to significantly increase the accuracy of position estimation in autonomous driving compared to an inertial measurement unit [2]. Moreover, by leveraging shared resources on the Chameleon Cloud testbed [6], we were able to efficiently conduct the project without the need of expensive in-house hardware. Our work also guarantees practical reproducibility of our experimental pipeline. Finally, we packaged the code for the experiment into an artifact publicly accessible on Chameleon's Trovi [9] sharing platform. The remainder of this paper is organized as follows: in Section 2, we discuss our approach, and present details in the implementation. In Section 3, we present the evaluation of our work. Section 4 we discuss difficulties encountered during the experiment and outline how our experiment can be reproduced. Lastly, we conclude the paper.

2 Approach

Authors in [1] demonstrated that a CNN trained on 72 hours of real driving data could operate in a real-road setting with 98% precision. Our approach is based on this work and its replication, resulting in a 1:18 scale car equipped with a Raspberry Pi 4 as its onboard computer, collecting data by driving the car around an oval track. Our first goal was to achieve similar results to this work, and then to extend their experiments to achieve fully autonomous driving on multiple surface environments. To accomplish this, we equipped the car with an optical encoder sensor to measure current speed. The optical encoder was attached to the drive shaft of the car. The small size of the car prevented us from directly fitting encoders on the wheels, as mentioned in [2]. **Figure** 2 illustrates how the optical encoder was attached to the car. Specifically, our proposed solution to the problem of autonomous speed control is the **Velocity Model**, which uses the same CNN structure as the default model packaged with Donkey Car (the Linear model), but it is trained to predict the current speed of the car based on the image input, while the default Linear model predicts throttle percentage. Furthermore, to measure performance on different road surfaces, we created two tracks: the default track (see Figure 3a) consists of two lines of orange tape on carpet flooring, and the Waveshare track (see Figure 3b) which is an orange-lined track printed on a mat laid out on the floor [3].

By ensuring that the models are built using the same CNN structure, we control for model complexity. Additionally, the models are trained on the same dataset. We anticipate that the model performance should be independent of battery percentage, and that the performance is improved on different surfaces compared to the control. Furthermore, we aim to learn to be more cautious and, as a result, have fewer errors. **Figure** 4 illustrates our solution architecture and the I/O structure for Velocity Model. Essentially, the optical encoder is used to capture the car's current speed during data collection. Then, we train a CNN to predict steering angle and the car's current speed based on the image captured by the camera. During inferencing, we compare the predicted speed versus the actual speed observed by the encoder and make adjustments to the throttle as necessary to speed up or slow down the car so that it matches the speed predicted by the model.

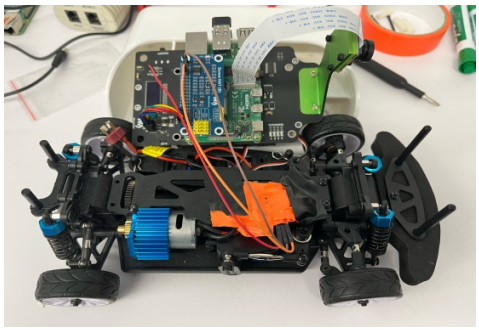


Figure 2. Encoder is attached to internal drive shaft (obscured by orange tape) and connected back to the Raspberry Pi

(a) Default track **(b)** Waveshare track

Figure 3. Two different oval tracks were utilized for the experiments.

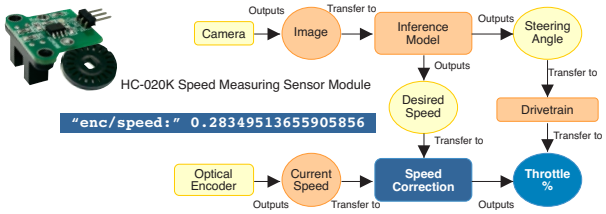


Figure 4. Solution Architecture, I/O structure for Velocity model.

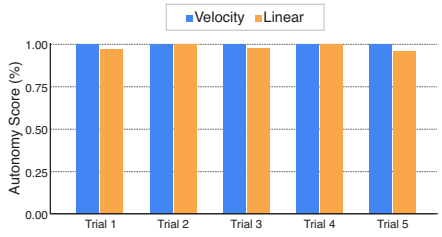
3 Evaluation

The components of the car were purchased as part of the Waveshare PiRacer Pro AI Kit [3]. This kit includes all the required components for our experiment: Raspberry Pi 4, a compatible camera, the Waveshare track, a remote controller for human-controlled driving, as well as the physical parts of the car, all for around $250 USD. We believe this relatively

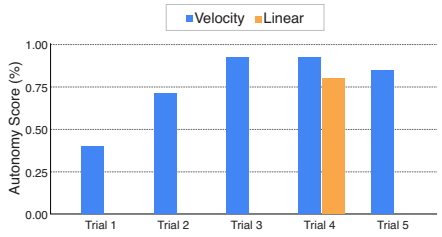
low cost is desirable for an entire small-scale self-driving testbed. We tested the models on both tracks (**Figure** 3). The car was allowed to autonomously drive for 10 minutes at a time for 5 trials. An error is defined as both front wheels of the car leaving the track or the car stopping for more than 5 seconds. Successful laps are defined as full laps around the track completed with zero errors. The accuracy of the models was calculated using an autonomy score:

$$\textbf{Autonomy Score} = 1 - \frac{\text{Number of Errors}}{\text{Number of Laps}} \quad (1)$$

Equation 1 outputs an autonomy score that shows how well a model could complete a full lap without error. The autonomy value was at or near 1.0 for both models on the default track (Figure 5a). On the other hand, in the Waveshare track, Velocity averaged 473% higher autonomy than Linear (Figure 5b). On the default track, the models made little or no errors (Figure 6a). On the Waveshare track, Linear made 7.27× as many errors as Velocity (Figure 6b).



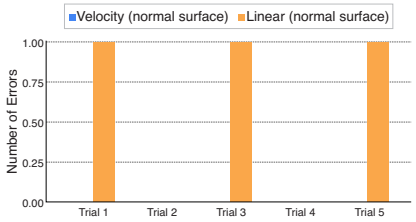
(a) Autonomy Default track



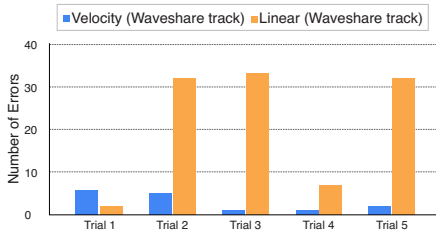
(b) Autonomy Waveshare track

Figure 5. Results show few or no errors on the default track, whereas on the Waveshare track, the Linear model made 7.27× as many errors as Velocity. Autonomy is a percentage score showing for what percentage of a lap the car drove on the road. Trial 4 of the linear model on the Waveshare track shows an outlier where the car was able to drive well for a sustained time, which it was unable to attain in other trials.

Errors were recorded on both the default and Waveshare tracks during a 10-minute period. On the default track, successful laps using the Linear model were 2.11 times quicker than with Velocity (Figure 7a). On the Waveshare track, this ratio was 2.85 (Figure 7b). Over the course of 5 trials, the velocity model successfully completed 1.86 times as many laps on the Waveshare track as the linear model. Considering that

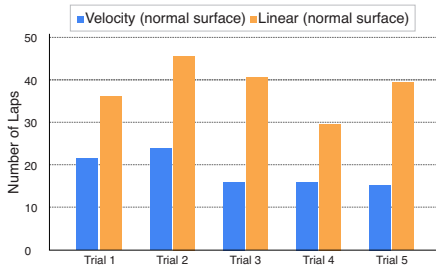


(a) Number of Errors: Default track

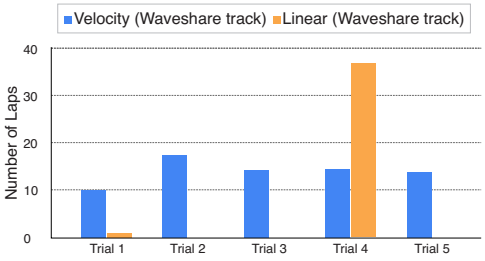


(b) Number of Errors: Waveshare track

Figure 6. Errors made on default and Waveshare tracks respectively in a 10 minutes period.



(a) Default track



(b) Waveshare track

Figure 7. Successful laps completed on default and Waveshare tracks respectively in a 10 minutes period. This plot gives insight into which track allowed for faster laps.

the velocity model also achieved a 473% higher autonomy score than the linear model, we can observe that the velocity model, when operating on track surface conditions different from the environment it was trained on, outperformed the linear model in reliability, despite both models being trained on the same dataset.

4 Experimental Methodology and Reproducibility

In this section, we will explain how our experiment can be reproduced. First, one must purchase and assemble the small-scale car. In the AutoLearn education module [4], a detailed guide for purchasing and assembling the physical components of the car is outlined in the instruction guide [10]. This instruction guide also outlines the setup process for connecting the Raspberry Pi to CHI@Edge. Finally, an artifact containing the code needed to run the experiment can be accessed on Trovi [5], Chameleon's artifact sharing platform. Conducting this experiment was not without difficulty. Working with physical components can be time-consuming, as required parts must be ordered and delivered, and inconsistent, as parts can break or wear down. Making modifications to the vehicle, like we did with the encoder, can also add an additional layer of variability that we would be unable to control in subsequent iterations of this experiment.

Despite these difficulties, our experimental workflow, once the physical components were set up, was streamlined by the use of shared resources on Chameleon, which facilitated the experimentation through the utilization of the shared GPU resources provided by the infrastructure. Since we needed to train multiple self-driving models on large amounts of data, this would have taken excessive amounts of time if we only had access to our laptops or even basic CPUs. By using Chameleon, we could reserve a GPU for a period of time, complete model training, and then load the trained models back onto the Raspberry Pi. Finally, our consideration of reproducibility while conducting the experiment not only made our end result more useful to future researchers, but it also helped us backtrack through our workflow while working on the experiment.

5 Conclusions

In order to ensure safe autonomous driving, neural networks must be able to properly control speed. We presented a comparison of two different methods for controlling the steering and speed of a scale-model car with a CNN. Our proposed solution, which predicts the current speed of the car, resulted in 7.47 times fewer errors than a model of the same structure that predicts throttle percentage. These results demonstrate that a velocity-dependent model is better suited for safety in autonomous driving. Future work will involve examining the trade-offs between model accuracy and car speed, as highlighted in previous studies such as [8]. By exploring these dynamics, existing models can be refined and more robust autonomous driving systems capable of navigating diverse environments with precision can be developed. Additionally, further experimental setups on the Chameleon testbed using CHI@Edge computing infrastructures with complex real-world implementation and the integration of other optical encoder sensor systems for real-time speed data collection can be explored to ensure accurate model predictions.

Acknowledgments

This paper's results were obtained with support from the National Science Foundation through the Chameleon testbed (Award Number 2027170) and the FOUNT project (Award Number 2230077). The opinions, findings, conclusions, or recommendations expressed in this publication are solely those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

[2] Niklas Brusên. 2019. Simultaneous Localization and Mapping of a small-scale vehicle using low-cost IMU, optical wheel encoders and LiDAR.

[3] Waveshare Electronics. [n. d.]. PiRacer Pro AI Kit. https://www.waveshare.com/piracer-pro-ai-kit.htm. Accessed: 2024-4-12.

[4] Alicia Esquivel Morel, William Fowler, Kate Keahey, Kyle Zheng, Michael Sherman, and Richard Anderson. 2023. AutoLearn: Learning in the Edge to Cloud Continuum. In *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis* (, Denver, CO, USA,) *(SC-W '23)*. Association for Computing Machinery, New York, NY, USA, 350–356. https://doi.org/10.1145/3624062.3624101

[5] Keahey Kate, Anderson Jason, Powers Mark, and Cooper Adam. 2023. Three Pillars of Practical Reproducibility. In *rewords23: 3rd Workshop on Reproducible Workflows, Data Management, and Security During eScience'23*.

[6] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S. Gunawi, Cody Hammock, Joe Mambretti, Alexander Barnes, François Halbach, Alex Rocha, and Joe Stubbs. 2020. Lessons Learned from the Chameleon Testbed. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association.

[7] Tawm Kramer. 2016. Donkey Car. https://github.com/autorope/donkeycar.

[8] Xiaopeng Li. 2022. Trade-off between safety, mobility and stability in automated vehicle following control: An analytical method. *Transportation research part B: methodological* 166 (2022), 1–18.

[9] Alicia Esquivel Morel, William Fowler, Kate Keahey, Kyle Zheng, Michael Sherman, and Richard Anderson. 2023. AutoLearn Autonomous Cars. https://www.chameleoncloud.org/experiment/share/8800ebd1-411e-4e94-9b62-6883f09188e7 [Accessed: 2024-3-3].

[10] Alicia Esquivel Morel, William Fowler, Kate Keahey, Kyle Zheng, Michael Sherman, and Richard Anderson. 2023. Chi@Edge Education. https://chi-education.gitbook.io/chi-edge-or-education/ [Accessed: 2024-3-3].

[11] Mark Ryan. 2020. The future of transportation: ethical, legal, social and economic impacts of self-driving vehicles in the year 2025. *Science and engineering ethics* 26, 3 (2020), 1185–1208.

[12] Shakti N Wadekar, Benjamin J Schwartz, Shyam S Kannan, Manuel Mar, Rohan Kumar Manna, Vishnu Chellapandi, Daniel J Gonzalez, and Aly El Gamal. 2021. Towards end-to-end deep learning for autonomous racing: On data collection and a unified architecture for steering and throttle prediction. *arXiv preprint arXiv:2105.01799* (2021).