Supplementary Materials: Towards Driving-Oriented Metric for Lane Detection Models

Takami Sato
University of California, Irvine
takamis@uci.edu

Qi Alfred Chen
University of California, Irvine
alfchen@uci.edu

A. Detailed Settings of Lane Detection Metrics

Table 1 shows the required parameters and input to calculate each metric for lane detection. As shown, only E2E-LD requires multiple frames for computation. For E2E-LD and PSLD, we adopt the kinematic bicycle model [7] to simulate the vehicle motion. The only parameter in the kinematic bicycle model is the wheelbase. We use WheelBase=2.65 meters, which is the wheelbase of Toyota RAV4 2017.

Table 1. Parameter and input of metrics for Lane Detection

	Parameter	Input	Per-frame
Accuracy	α	X_0	✓
F1 score	α, β	X_0	✓
E2E-LD	T_E, WheelBase	$X_0, ..., X_{T_E}, C$	
PSLD	T_p, WheelBase	X_0, C	✓

B. Detailed Attack Implementation

We use the official implementation of the DRP attack [12]. We also use parameters that are reported to have the best balance between effectiveness and secrecy: the learning rate is 10^{-2}, the regularization parameter λ is 10^{-3}, and the perturbable area ratio (PAR) is 50%. We run 200 iterations to generate the patch in all experiments.

C. Details of Comma2k19-LD dataset

Fig. 1 shows the first frame of all 100 scenarios and its lane line annotations. For annotation, human annotators mark the lane line points and check if the linear interpolation results of the markings align with the lane line information. To convert the annotations to the TuSimple dataset format, we sample points every 10 pixels in the y-axis from the interpolated results.

D. Adaptation to TuSimple Challenge Camera Frames Geometory

In the evaluation of the comma2k19-LD dataset, we use the same pretrained models trained on the TuSimple Challenge training dataset. To deal with the differences in the datasets, we convert the camera frames in the comma2k19-LD dataset to have similar geometry as the camera frames in the TuSimple challenge dataset. Fig. 2 illustrates the overview of the conversion. We remove the surrounding area and use only the central part of the Comma2k19-LD camera frame to have the same sky-ground area ratio and the same lane occupation ratio in the image width with the ones in the TuSimple dataset.

E. Evaluation of the Domain Shift Effect

In this study, we use lane detection models pretrained on the TuSimple dataset and evaluate them on the Comma2k19-LD dataset. Although both datasets are similar driver's view images, there can be some domain shifts between them. To understand the impact, we trained the 4 models with another 100 scenarios extracted from the Comma2k19 dataset. We run 10 epochs with the data on top of the models pretrained on the TuSimple dataset. For the lane line labels, we use OpenPilot's lane detection results in the dataset. We conduct the same evaluation in §4.2 and §4.3 of the main paper. As shown in Table 2 and Table 3, the observations are consistent: SCNN outperforms in the conventional metrics; PolyLaneNet is the most robust in attack scenarios. The Pearson correlation coefficients show almost the same results as the ones in §4.2 and §4.3 that the conventional metrics have strong negative correlations in the benign scenarios and the correlations in the attack scenarios are not statistically significant.

However, the E2E-LD in the attack scenarios are generally higher than the results in §4.2 and §4.3 of the main paper while the E2E-LD in the benign scenarios is generally lower. This indicates that this additional fine-tuning improves the performance in the benign scenarios, but it harms the robustness against adversarial attacks.

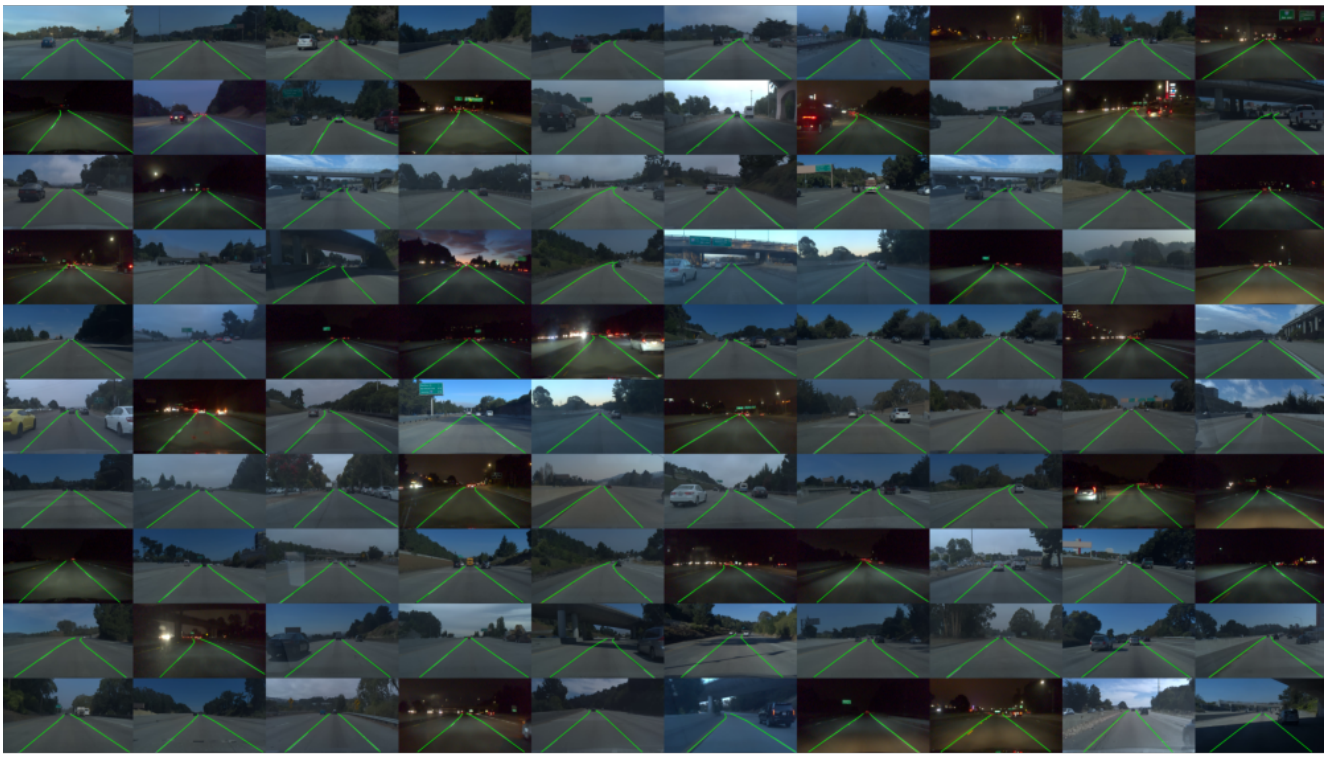


Figure 1. The first frame of all 100 scenarios and its lane line annotations (green line)

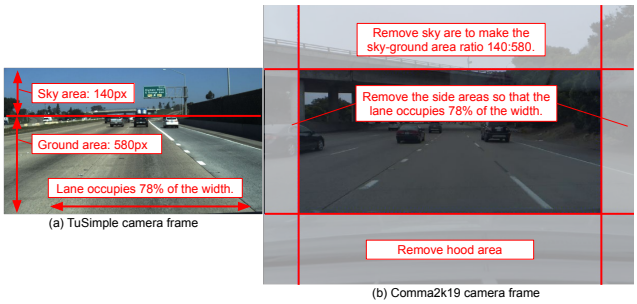
Table 2. Evaluation results of the E2E-LD and the conventional metrics, accuracy and F1 in the benign and attack scenarios. For each metric, the corresponding Pearson correlation coefficient with E2E-LD in the bottom rows. The **bold** and underlined letters indicate the highest and lowest performance or correlation, respectively.

		Benign			Attack		
			Original Parameters ($\alpha = 20, \beta = 0.85$)			Original Parameters ($\alpha = 20, \beta = 0.85$)	
		E2E-LD [m]	Accuracy	F1	E2E-LD [m]	Accuracy	F1
Metric	SCNN [8]	0.20	**0.93**	0.84	**0.52**	**0.67**	0.30
	UltraFast [9]	0.18	0.92	0.84	0.62	0.49	0.16
	PolyLaneNet [14]	**0.13**	**0.93**	**0.86**	0.54	0.62	**0.33**
	LaneATT [13]	0.14	**0.93**	0.85	0.71	0.51	0.12
Corr.	SCNN [8]	-	-0.65***	-0.51***	-	-0.09^{ns}	-0.04^{ns}
	UltraFast [9]	-	-0.63***	-0.60***	-	$\mathbf{0.14}^{ns}$	$\mathbf{0.07}^{ns}$
	PolyLaneNet [14]	-	**-0.32****	-0.62***	-	$\mathbf{0.14}^{ns}$	0.04^{ns}
	LaneATT [13]	-	-0.57***	**-0.26****	-	-0.02^{ns}	-0.06^{ns}

^{ns} Not Significant ($p > 0.05$),* $p \leq 0.05$,** $p \leq 0.01$,*** $p \leq 0.001$

F. Alternative metric design

To improve the conventional metrics, one of the most intuitive approaches is the $\mathcal{L}_1$ or $\mathcal{L}_2$ distance in the bird's eye view because they do not suffer from the problem of the ill parameters discussed in §2.2, and lane detection results from a bird's eye view may be a more adequate to measure of drivability than detection results from a front camera. We actually have considered such metrics before, but we did not finally choose them because, without some form of control simulation, we find it fundamentally nontrivial to accurately predict the combined effects of detection errors at different lane line positions and with different error amounts on the downstream AD driving. This can be concretely shown in Table 4. As shown, both such 3D-L1 and 3D-L2 distance metrics have considerably lower correlation coefficient r with E2E-LD compared to our PSLD. They are indeed better than conventional accuracy and F1 score metrics. However, they are still leading to erroneous judgment on downstream AD performance similar to the accuracy and F1 score: e.g., PolyLaneNet is 2nd-worst based on 3D-L1/L2 distance metrics in the attack scenarios, but in E2E-LD it is the best. With our PSLD, such judgment is



(a) TuSimple camera frame

(b) Comma2k19 camera frame

Figure 2. Overview of adapting the camera frames in Comma2k19-LD dataset to the camera frame in the TuSimple Challenge dataset. We remove the surrounding area and use only the central part of the Comma2k19 camera frame to ensure that the comma2k19-LD camera frames have similar geometry as the TuSimple challenge camera frames.

Table 3. Evaluation results of the E2E-LD and PSLD in the benign and attack scenarios. The format is the same as Table 2.

		Benign		Attack	
		E2E-LD [m]	PSLD [m]	E2E-LD [m]	PSLD [m]
Metric	SCNN [8]	0.20	0.04	**0.52**	0.61
	UltraFast [9]	0.18	**0.02**	0.62	0.66
	PolyLaneNet [14]	**0.13**	**0.02**	0.54	**0.55**
	LaneATT [13]	0.14	0.03	0.71	0.82
Corr.	SCNN [8]	-	**0.93*****	-	0.93***
	UltraFast [9]	-	0.60***	-	**0.99*****
	PolyLaneNet [14]	-	0.65***	-	**0.99*****
	LaneATT [13]	-	0.55***	-	0.78***

^{ns} Not Significant ($p > 0.05$),* $p \leq 0.05$,** $p \leq 0.01$,*** $p \leq 0.001$

strictly consistent with E2E-LD (Table 4). One reason we observe is that the 3D-L1/L2 metrics can be greatly biased by farther points; those points by design have much less impact on the downstream AD control, but suffer from more detection errors (due to the far distance). One thought is to assign smaller weights to farther points, but how to systematically decide such weights without any form of control simulation is fundamentally nontrivial. Additionally, such a weight-based design can still be fundamentally limited in achieving sufficient AD control relating capabilities.

Table 4. Pearson correlation coefficient r with E2E-LD. *3D-L1/L2* denote the L1/L2 distances in 3D space following Reviewer 1's suggestion. **Bold** and underline denote highest and lowest scores.

	Benign			Attack		
	PSLD (*ours*)	3D-L1	3D-L2	PSLD	3D-L1	3D-L2
SCNN	**0.93**	0.71	0.65	**0.96**	0.38	0.34
UltraFast	**0.54**	0.24	0.19	**0.93**	0.24	0.21
PolyLaneNet	**0.49**	0.47	0.44	**0.97**	0.33	0.38
LaneATT	**0.38**	0.23	0.17	**0.95**	0.23	0.23
Average	**0.59**	0.41	0.36	**0.95**	0.29	0.29

G. Details of OpenPilot ALC and its integration with lane detection models

In this section, we explain the details of OpenPilot ALC [2] and the details of its integration with the 4 lane detection models we evaluate in this study. As described in [12], the OpenPilot ALC system consists of 3 steps: lane detection, lateral control, and vehicle actuation.

G.1. Lane detection

The image frame from the front camera is input to the lane detection model in every frame (20Hz). Since the original OpenPilot lane detection model is a recurrent neural network model, the recurrent input from the previous frame is fed to the model with the image. All 4 models used in this study do not have a recurrent structure, i.e., they detect lanes only in the current frame. This is because the TuSimple Challenge has a runtime limit of less than 200 ms for each frame. Another famous dataset, CULane [8], does not provide even continuous frames. In autonomous driving, the recurrent structure is a reasonable choice since past frame information is always available. Hence, the runtime calculation latency imposed in the TuSimple challenge is one of the gaps between the practicality for autonomous driving and the conventional evaluation.

G.2. Lateral control

Based on the detected lane line, the lateral control decides steering angle decisions to follow the lane center (i.e., the desired driving path or waypoints) as much as possible. The original OpenPilot model outputs 3 line information: left lane line, right lane line, and driving path. The desired driving path is calculated as the average of the driving path and the center line of the left and right lane lines. The steering decision is decided by the model predictive control (MPC) [11]. The detected lane lines are represented in the bird's-eye-view (BEV) because the steering decision needs to be decided in a world coordinate system.

On the contrary, all 4 models used in this study detect the lane lines in the front-camera view. We thus project the detected lane lines into the BEV space with perspective transformation [6, 15]. The transformation matrix for this projection is created manually based on road objects such as lane markings, and then calibrated to be able to drive in a straight lane. We create the transformation matrix for each scenario as the position of the camera and the tilt of the ground are different for each scenario. The desired driving path is calculated by the average of the left and right lane lines and fed to the MPC to decide the steering angle decisions.

In addition to the desired driving path, the MPC receives the current speed and steering angle to decide the steering angle decisions. For the steering angle, we use the human

driver's steering angle in the Comma2k19 dataset in the first frame. In the following frames, the steering angle is updated by the kinematic bicycle model [10], which is the most widely-used motion model for vehicles. For the vehicle speed, we use the speed of human driving in the in the comma2k19 dataset as we assume that the vehicle speed is not changed largely in the free-flow scenario, in which a vehicle has at least 5–9 seconds clear headway [5].

G.3. Vehicle actuation

The step sends steering change messages to the vehicle based on the steering angle decisions. In OpenPlot, this step operates at 100 Hz control frequency. As the lane detection and lateral control outputs the steering angle decisions in 20 Hz, the vehicle actuation sends 5 messages every steering angle decision. The steering changes are limited to a maximum value due to the physical constraints of vehicle and for stable and for stability and safety. In this study, we limit the steering angle change to 0.25° following prior work, which is the steering limit for production ALC systems [12].

We update the vehicle states with the kinematic bicycle model based on the steering change. Note that like all motion models, the kinematic bicycle model does have approximation errors to the real vehicle dynamics [7]. However, more accurate motion models require more complex parameters such as vehicle mass, tire stiffness, and air resistance [1]. In this study, since our focus is on understanding the impact of lane detection model robustness on end-to-end driving, the most widely-used kinematic bicycle model is a sufficient choice for simulating closed-loop control behaviors.

H. Additional Discussions and Results

H.1. Additional Discussions

Ground-Truth Road Center. We obtain the ground truth waypoints based on the human driving traces. Ideally, the waypoints should be obtained by measuring roads. However, since this study focuses on the general trends of the 4 lane detection approaches, we consider that the impact of this factor should not have a major effect. If you want to use PSLD to capture more subtle differences between models, the ground truth should be more accurate.

Differentiable PSLD Regularization. We show that PSLD works as a good surrogate for E2E-LD. Next, we may want to minimize this metric directly in the model training. Since the only non-differentiable computation in PSLD is the lateral controller, we can replace this part with a differentiable controller [4, 16] and incorporate it as a regularization term in the loss function for training. Detailed study of this problem is left to future work.

H.2. TuSimple Challenge Dataset

Fig. 3 shows the examples of lane detection results and the accuracy metric in benign scenarios on the TuSimple Challenge dataset [3]. The limitations of the conventional metrics can be found in benign cases as well. As shown, SCNN has always higher accuracy than PolyLaneNet (at most 18% edge). Such a large leading edge is across the dataset as in Table 2 (89% vs 72% in Accuracy, 75% vs 50% in F1 score). However, for downstream AD their performances are almost the same, with PolyLaneNet actually slightly better (Table 3 of the main paper).

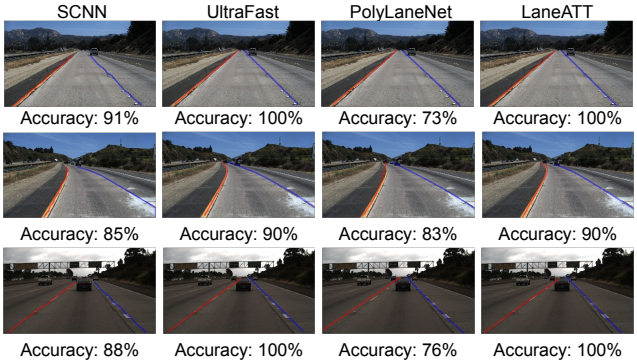


Figure 3. Examples of lane detection results and the accuracy metric in benign scenarios on TuSimple Challenge dataset [3]. As shown, the conventional accuracy metric does not necessarily indicate drivability if used in autonomous driving

H.3. Comma2k19 LD Dataset

We synthesize font-camera frames with a vehicle motion model [10] and perspective transformation [6, 15]. Fig. 4, 5, 6, 7, show the first 20 frames under attack and their detection results of the 4 lane detection methods, respectively. As shown, the generated images are generally complete and the distortion is very slight.

Figure 4. The first 20 frames (from left-top to right-bottom) of an attack scenario on **SCNN**. The vehicle is deviating to right due to the attack.

Figure 5. The first 20 frames (from left-top to right-bottom) of an attack scenario on **UltraFast**. The vehicle is deviating to right due to the attack.

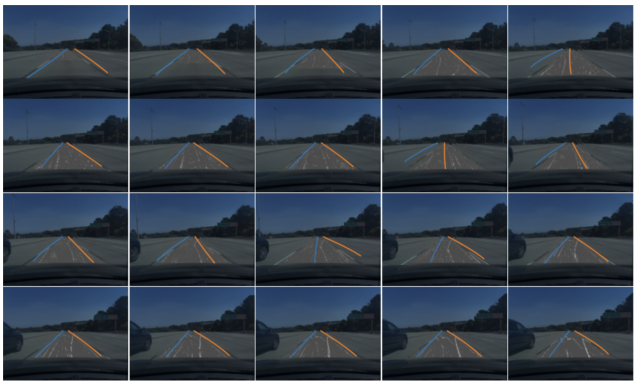


Figure 6. The first 20 frames (from left-top to right-bottom) of an attack scenario on **PolyLaneNet**. The vehicle is deviating to right due to the attack.

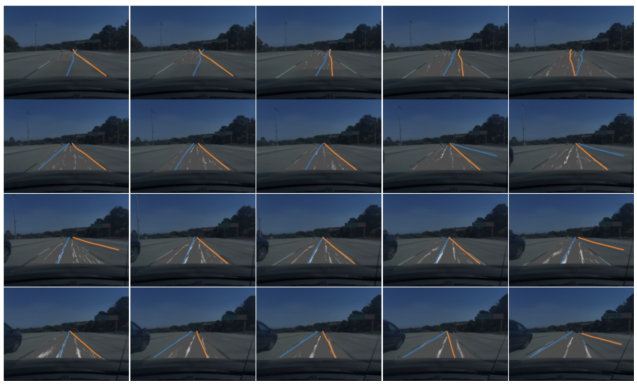


Figure 7. The first 20 frames (from left-top to right-bottom) of an attack scenario on **LaneATT**. The vehicle is deviating to right due to the attack.

References

[1] Modeling a Vehicle Dynamics System. https://www.mathworks.com/help/ident/ug/modeling-a-vehicle-dynamics-system.html. 4

[2] OpenPilot: Open Source Driving Agent. https://github.com/commaai/openpilot. 3

[3] TuSimple Lane Detection Challenge. https://github.com/TuSimple/tusimple-benchmark/tree/master/doc/lane_detection, 2017. 4

[4] Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable MPC for end-to-end planning and control. In *NeurIPS*, 2018. 4

[5] Amardeep Boora, Indrajit Ghosh, and Satish Chandra. Identification of free flowing vehicles on two lane intercity highways under heterogeneous traffic condition. *Transportation Research Procedia*, 21:130–140, 2017. 4

[6] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2003. 3, 4

[7] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design. In *IV*, 2015. 1, 4

[8] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as Deep: Spatial CNN for Traffic Scene Understanding. In *AAAI*, 2018. 2, 3

[9] Qin, Zequn and Wang, Huanyu and Li, Xi. Ultra Fast Structure-Aware Deep Lane Detection. In *ECCV*, 2020. 2, 3

[10] Rajesh Rajamani. *Vehicle Dynamics and Control*. Springer Science & Business Media, 2011. 4

[11] Richalet, J. and Rault, A. and Testud, J. L. and Papon, J. Paper: Model Predictive Heuristic Control. *Automatica*, 14(5):413–428, Sept. 1978. 3

[12] Takami Sato, Junjie Shen, Ningfei Wang, Yunhan Jia, Xue Lin, and Qi Alfred Chen. Dirty Road Can Attack: Security of Deep Learning based Automated Lane Centering under Physical-World Attack. *USENIX Security Symposium*, 2021. 1, 3, 4

[13] Lucas Tabelini, Rodrigo Berriel, Thiago M. Paix ao, Claudine Badue, Alberto Ferreira De Souza, and Thiago Oliveira-Santos. Keep your Eyes on the Lane: Real-time Attention-guided Lane Detection. In *CVPR*, 2021. 2, 3

[14] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixao, Claudine Badue, Alberto F De Souza, and Thiago Oliveira-Santos. Polylanenet: Lane Estimation via Deep Polynomial Regression. In *ICPR*, 2021. 2, 3

[15] Shiho Tanaka, Kenichi Yamada, Toshio Ito, and Takenao Ohkawa. Vehicle Detection Based on Perspective Transformation Using Rear-View Camera. *Hindawi Publishing Corporation International Journal of Vehicular Technology*, 9, 03 2011. 3, 4

[16] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *ICRA*, 2014. 4