Dirty Road Can Attack: Security of Deep Learning based Automated Lane Centering under Physical-World Attack

Takami Sato, Junjie Shen, and Ningfei Wang, *University of California, Irvine;* Yunhan Jia, *ByteDance;* Xue Lin, *Northeastern University;* Qi Alfred Chen, *University of California, Irvine*

https://www.usenix.org/conference/usenixsecurity21/presentation/sato

This paper is included in the Proceedings of the 30th USENIX Security Symposium.

August 11–13, 2021

978-1-939133-24-3

Open access to the Proceedings of the 30th USENIX Security Symposium is sponsored by USENIX.

Dirty Road Can Attack: Security of Deep Learning based Automated Lane Centering under Physical-World Attack

Takami Sato*
UC Irvine
takamis@uci.edu

Junjie Shen*
UC Irvine
junjies1@uci.edu

Ningfei Wang
UC Irvine
ningfei.wang@uci.edu

Yunhan Jia
ByteDance
yunhan.jia@bytedance.com

Xue Lin
Northeastern University
xue.lin@northeastern.edu

Qi Alfred Chen
UC Irvine
alfchen@uci.edu

Abstract

Automated Lane Centering (ALC) systems are convenient and widely deployed today, but also highly security and safety critical. In this work, we are the first to systematically study the security of state-of-the-art deep learning based ALC systems in their designed operational domains under physical-world adversarial attacks. We formulate the problem with a safety-critical attack goal, and a novel and domain-specific attack vector: dirty road patches. To systematically generate the attack, we adopt an optimization-based approach and overcome domain-specific design challenges such as camera frame inter-dependencies due to attack-influenced vehicle control, and the lack of objective function design for lane detection models.

We evaluate our attack on a production ALC using 80 scenarios from real-world driving traces. The results show that our attack is highly effective with over 97.5% success rates and less than 0.903 sec average success time, which is substantially lower than the average driver reaction time. This attack is also found (1) robust to various real-world factors such as lighting conditions and view angles, (2) general to different model designs, and (3) stealthy from the driver's view. To understand the safety impacts, we conduct experiments using software-in-the-loop simulation and attack trace injection in a real vehicle. The results show that our attack can cause a 100% collision rate in different scenarios, including when tested with common safety features such as automatic emergency braking. We also evaluate and discuss defenses.

1 Introduction

Automated Lane Centering (ALC) is a Level-2 driving automation technology that automatically steers a vehicle to keep it centered in the traffic lane [1]. Due to its high convenience for human drivers, today it is widely available on various vehicle models such as Tesla, GM Cadillac, Honda Accord, Toyota RAV4, Volvo XC90, etc. While convenient, such system is highly security and safety critical: When the ALC system starts to make wrong steering decisions, the human driver may not have enough reaction time to prevent safety hazards such as driving off road or colliding into vehicles in adjacent lanes. Thus, it is imperative and urgent to understand the security property of ALC systems.

In an ALC system, the most critical step is *lane detection*, which is generally performed using a front camera. So far, Deep Neural Network (DNN) based lane detection achieves the highest accuracy [2] and is adopted in the most performant production ALC systems today such as Tesla Autopilot [3]. Recent works show that DNNs are vulnerable to physical-world adversarial attacks such as malicious stickers on traffic signs [4, 5]. However, these methods cannot be directly applied to attack ALC systems due to two main design challenges. First, in ALC systems, the physical-world attack generation needs to handle *inter-dependencies* among camera frames due to attack-influenced vehicle actuation. For example, if the attack deviates the detected lane to the right in a frame, the ALC system will steer the vehicle to the right accordingly. This causes the following frames to capture road areas more to the right, and thus directly affect their attack generation. Second, the optimization objective function designs in prior works are mainly for image classification or object detection models and thus aim at changing class or bounding box probabilities [4, 5]. However, attacking lane detection requires to change the *shape* of the detected traffic lane, making it difficult to directly apply prior designs.

The only prior effort that studied adversarial attacks on a production ALC is from Tencent [6], which fooled Tesla Autopilot to follow fake lane lines created by white stickers on road regions *without lane lines*. However, it is neither attacking the designed operational domain for ALC, i.e., roads *with lane lines*, nor generating the perturbations systematically by addressing the design challenges above.

To fill this critical research gap, in this work we are the first to systematically study the security of DNN-based ALC systems in their designed operational domains (i.e., roads with lane lines) under physical-world adversarial attacks. Since ALC systems assume a fully-attentive human driver prepared to take over at any time [1, 7], we identify the attack goal as not only causing the victim to drive out of the current

*Co-first authors.

lane boundaries, but also achieving it shorter than the average driver reaction time to road hazard. This thus directly breaks the design goal of ALC systems and can cause various types of safety hazards such as driving off road and vehicle collisions.

Targeting this attack goal, we design a novel physical-world adversarial attack method on ALC systems, called *DRP (Dirty Road Patch) attack*, which is the first to systematically address the design challenges above. First, we identify *dirty road patches* as a novel and domain-specific attack vector for physical-world adversarial attacks on ALC systems. This design has 2 unique advantages: (1) Road patches can appear to be legitimately deployed on traffic lanes in the physical world, e.g., for fixing road cracks; and (2) Since it is common for real-world roads to have dirt or white stains, using similar dirty patterns as the input permutations can allow the malicious road patch to appear more normal and thus stealthier.

With this attack vector, we then design systematic malicious road patch generation following an optimization-based approach. To efficiently and effectively address the first design challenge without heavyweight road testing or simulations, we design a novel method that combines vehicle motion model and perspective transformation to dynamically synthesize camera frame updates according to attack-influenced vehicle control. Next, to address the second design challenge, one direct solution is to design the objective function to directly change the steering angle decisions. However, we find that the lateral control step in ALC that calculates steering angle decisions are generally not differentiable, which makes it difficult to effectively optimize. To address this, we design a novel lane-bending objective function as a differentiable surrogate function. We also have domain-specific designs for attack robustness, stealthiness, and physical-world realizability.

We evaluate our attack method on a production ALC system in OpenPilot [8], which is reported to have close performance to Tesla Autopilot and GM Super Cruise, and better than many others [9]. We perform experiments on 80 attack scenarios from real-world driving traces, and find that our attack is highly effective with over 97.5% success rates for all scenarios, and less than 0.903 sec average success time, which is substantially lower than 2.5 sec, the average driver reaction time (§3.1). This means that even for a fully-attentive driver who can take over as soon as the attack starts to take effect, the average reaction time is still not enough to prevent the damage. We further find this attack is (1) robust to real-world factors such as different lighting conditions, viewing angles, printer color fidelity, and camera sensing capability, (2) general to different lane detection model designs, and (3) stealthy from the driver's view based on a user study.

To understand the potential safety impacts, we further conduct experiments using (1) software-in-the-loop simulation in a production-grade simulator, and (2) attack trace injection in a real vehicle. The simulation results show that our attack can successfully cause a victim running a production ALC to hit the highway concrete barrier or a truck in the opposite direction with 100% success rates. The real-vehicle experiments show that it causes the vehicle to collide with (dummy) road obstacles in all 10 trials even with common safety features such as Automatic Emergency Braking (AEB) enabled. Demo videos are available at: **https://sites.google.com/view/cav-sec/drp-attack/**. We also explore and discuss possible defenses at DNN level and those based on sensor/data fusion.

In summary, this work makes the following contributions:

- We are the first to systematically study the security of DNN-based ALC in the designed operational domains under physical-world adversarial attacks. We formulate the problem with a safety-critical attack goal, and a novel and domain-specific attack vector, dirty road patches.
- To systematically generate attack patches, we adopt an optimization-based approach with 2 major novel and domain specific designs: motion model based input generation, and lane-bending objective function. We also have domain-specific designs for improving the attack robustness, stealthiness, and physical-world realizability.
- We perform evaluation on a production ALC using 80 attack scenarios from real-world driving traces. The results show that our attack is highly effective with $\geq$97.5% success rates and $\leq$0.903 sec average success time, which is substantially lower than the average driver reaction time. This attack is also found (1) robust to various real-world factors, (2) general to different lane detection model designs, and (3) stealthy from the driver's view.
- To understand the safety impacts, we conduct experiments using (1) software-in-the-loop simulation, and (2) attack trace injection in a real vehicle. The results show that our attack can cause a 100% collision rate in different scenarios, including when tested with safety features such as AEB. We also evaluate and discuss possible defenses.

Code and data release. Our code and data for the attack and evaluations are available at our project website [10].

2 Background

2.1 Overview of DNN-based ALC Systems

Fig. 1 shows an overview of a typical ALC system design [8, 11, 12], which operates in 3 steps:

Lane Detection (LD). Lane detection (LD) is the most critical step in an ALC system, since the driving decisions later are mainly made based on its output. Today, production ALC systems predominately use front cameras for this step [3, 13]. On the camera frames, an LD model is used to detect lane lines. Recently, DNN-based LD models achieve the state-of-the-art accuracy [14–16] and thus are adopted in the most performant production ALC systems today such as Tesla Autopilot [3]. Since lane line shapes do not change much across consecutive frames, recurrent DNN structure (e.g., RNN) is widely adopted in LD models to achieve more stable prediction [8, 17, 18]. LD models typically first predict the lane line points, and then post-process them to lane line

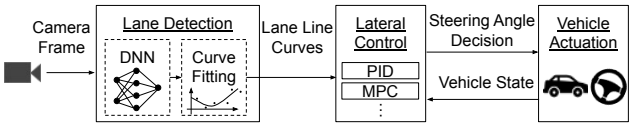

Figure 1: Overview of the typical ALC system design.

curves using curve fitting algorithms [14, 15, 19, 20].

Before the LD model is applied, a Region of Interest (ROI) filtering is usually performed to the raw camera frame to crop the most important area out of it (i.e., the road surface with lane lines) as the model input. Such ROI area is typically around the center and much smaller than the original frame, to improve the model performance and accuracy [21].

Lateral control. This step calculates *steering angle decisions* to keep the vehicle driving at the center of the detected lane. It first computes a desired driving path, typically at the center of the detected left and right lane lines [22]. Next, a control loop mechanism, e.g., Proportional-Integral-Derivative (PID) [23] or Model Predictive Control (MPC) [24], is applied to calculate the optimal steering angle decisions that can follow the desired driving path as much as possible considering the vehicle state and physical constraints.

Vehicle actuation. This step interprets the steering angle decision into actuation commands in the form of *steering angle changes*. Here, such actuated changes are limited by a maximum value due to the physical constraints of the mechanical control units and also for driving stability and safety [22]. For example, in our experiments with a production ALC with 100 Hz control frequency, such limit is 0.25° per control step (every 10 ms) for vehicle models [25]. As detailed later in §3.3, such a steering limit prevents ALC systems from being affected too much from successful attack in one single LD frame, which introduces a unique challenge to our design.

2.2 Physical-World Adversarial Attacks

Recent works find that DNN models are generally vulnerable to adversarial examples, or adversarial attacks [26, 27]. Some works further explored such attacks in the physical world [4, 5, 28–31]. While these prior works concentrate on DNN models for image classification and object detection tasks, we are the first to systematically study such attacks on production DNN-based ALC systems, which requires to address several new and unique design challenges as detailed later in §3.3.

3 Attack Formulation and Challenge

3.1 Attack Goal and Incentives

In this paper, we consider an attack goal that directly breaks the design goal of ALC systems: causing the victim vehicle a lateral deviation (i.e., deviating to the left or right) large enough to drive out of the current lane boundaries. Meanwhile, since ALC systems assume a fully-attentive human driver who is prepared to take over at any moment [1, 7], such deviation needs to be achieved fast enough so that the human driver cannot react in time to take over and steer back. Table 1 shows concrete values of these two requirements for successful attacks on highway and local roads respectively, which will be used as evaluation metrics later in §5. In the table, the required deviations are calculated based on representative vehicle and lane widths in the U.S., and the required success time is determined using commonly-used average driver reaction time to road hazards, which is detailed in Appendix A.

Table 1: Required deviations and success time for successful attacks on ALC systems on highway and local roads. Detailed calculations and explanations are in Appendix A.

Road Type	Required Lateral Deviation	Required Success Time
Highway	0.735 meters	<2.5 seconds (average driver reaction time to road hazard)
Local road	0.285 meters	

Targeted scenario: Free-flow driving. Our study targets the most common driving scenario for using ALC systems: *free-flow* driving scenarios [32], in which a vehicle has at least 5–9 seconds clear headway [33] and thus can drive freely without considering the front vehicle [32].

Safety implications. The attack goal above can directly cause various safety hazards in the real world: (1) *Driving off road*, which is a direct violation of traffic rules [34] and can cause various safety hazards such as hitting road curbs or falling down the highway cliff. (2) *Vehicle collisions*, e.g., with vehicles parked on the road side, or driving in adjacent or opposite traffic lanes on a local road or a two-lane undivided highway. Even with obstacle or collision avoidance, these collisions are still possible for two reasons. First, today's obstacle and collision avoidance systems are not perfect. For example, a recent study shows that the AEB (Automatic Emergency Braking) systems in popular vehicle models today fail to avoid crashes 60% of the time [35]. Second, even if they can successfully perform emergency stop, they cannot prevent the victim from being hit by other vehicles that fail to yield on time. Later in §7, we evaluate the safety impacts of our attack with a simulator and a real vehicle.

3.2 Threat Model

We assume that the attacker can obtain the same ALC system as the one used by the victim to get a full knowledge of its implementation details. This can be done through purchasing or renting the victim vehicle model and reverse engineering it, which has already been demonstrated possible on Tesla Autopilot [6]. Moreover, there exist production ALC systems that are open sourced [8]. We also assume that the attacker can obtain a motion model [36] of the victim vehicle, which will be used in our attack generation process (§4.2). This is a realistic assumption since the most widely-used motion model (used by us in §4.2) only needs vehicle parameters such as steering ratio and wheelbase as input [36], which can be directly found from vehicle model specifications. We assume the victim drives at the speed limit of the target road, which is the most common case for free-flow driving. In the attack preparation time, we assume that the attacker can collect the ALC inputs (e.g., camera frames) of the target road by driving

the victim vehicle model there with the ALC system on.

3.3 Design Challenges

Compared to prior works on physical-world adversarial attacks on DNNs, we face 3 unique design challenges:

C1. Lack of legitimately-deployable attack vector in the physical world. To affect the camera input of an ALC system, it is ideal if the malicious perturbations can appear legitimately around traffic lane regions in the physical world. To achieve high legitimacy, such perturbations also must not change the original human-perceived lane information. Prior works use small stickers or graffiti in physical-world adversarial attacks [4–6]. However, directly performing such activities to traffic lanes in public is illegal [37]. In our problem setting, the attacker needs to operate in the middle of the road when deploying the attack on traffic lanes. Thus, if the attack vector cannot be disguised as legitimate activities, it becomes highly difficult to deploy the attack in practice.

C2. Camera frame inter-dependency due to attack-influenced vehicle actuation. In real-world ALC systems, a successful attack on one single frame can barely cause any meaningful lateral deviations due to the steering angle change limit at the vehicle actuation step (§2.1). For example, for the vehicle models with 0.25° angle change limit per control loop (§2.1), even if a successful attack on a single frame causes a very large steering angle decision at MPC output (e.g., 90°), it can only cause at most 1.25° actuated steering angle changes before the next frame comes, which can only cause up to *0.3-millimeter* lateral deviations at 45 mph (∼72 km/h). More detailed explanations are in our extended version [38].

Thus, to achieve our attack goal in §3.1, the attack must be *continuously effective on sequential camera frames* to increasingly reach larger actuated steering angles and thus larger lateral deviations per frame. In this process, due to the dynamic vehicle actuation applied by the ALC system, the attack effectiveness for later frames are directly dependent on that for earlier frames. For example, if the attack successfully deviates the detected lane to the right in a frame, the ALC system will steer the vehicle to the right accordingly. This causes the following frames to capture road areas more to the right, and thus directly affect their attack generation. There are prior works considering attack robustness across sequential frames, e.g., using EoT [29, 30] and universal perturbation [39], but none of them consider frame inter-dependencies due to attack-influenced vehicle actuation in our problem setting.

C3. Lack of differentiable objective function design for LD models. To systematically generate adversarial inputs, prior works predominately adopt optimization-based approaches, which have shown both high efficiency and effectiveness [4, 26]. However, the objective function designs in these prior works are mainly for image classification [4, 30] or object detection [4, 5] models, which thus aim at decreasing class or bounding box probabilities. However, as introduced in §2.1, LD models output detected lane line curves, and thus to achieve our attack goal the objective function needs to aim at changing the *shape* of such curves. This is substantially different from decreasing probability values, and thus none of these existing designs can directly apply.

Closer to our problem, prior works that attack end-to-end autonomous driving models [40–43] directly design their objective function to change the final steering angle decisions. However, as described in §2.1, state-of-the-art LD models do not directly output steering angle decisions. Instead, they output lane line curves and rely on the lateral control step to compute the final steering angle decisions. However, many steps in the lateral control module, e.g., the desired driving patch calculation and the MPC framework, are generally not differentiable to the LD model input (i.e., camera frames), which makes it difficult to effectively optimize.

4 Dirty Road Patch Attack Design

In this paper, we are the first to systematically address the design challenges above by designing a novel physical-world attack method on ALC, called *Dirty Road Patch (DRP) attack*.

4.1 Design Overview

To address the 3 design challenges in §3.3, our DRP attack method has the following novel design components:

Dirty road patch: Domain-specific & stealthy physical-world attack vector. To address challenge **C1**, we are the first to identify *dirty road patch* as an attack vector in physical-world adversarial attacks. This design has 2 unique advantages. First, road patches can appear to be legitimately deployed on traffic lanes in the physical world, e.g., for fixing road cracks. Today, deploying them is made easy with adhesive designs [44] as shown in Fig. 2. The attacker can thus take time to prepare the attack in house by carefully printing the malicious input perturbations on top of such adhesive road patches, and then pretend to be road workers like those in Fig. 2 to quickly deploy it when the target road is the most vacant, e.g., in late night, to avoid drawing too much attention.

Second, since it is common for real-world roads to have dirt or white stains such as those in Fig. 2, using similar dirty patterns as the input perturbations can allow the malicious road patch to appear more normal and thus stealthier. To mimic the normal dirty patterns, our design only allows color perturbations on the gray scale, i.e., black-and-white. To avoid changing the lane information as discussed in §3.3, in our design we (1) require the original lane lines to appear exactly the same way on the malicious patch, if covered by the patch, and (2) restrict the brightness of the perturbations to be strictly lower than that of the original lane lines. To further improve stealthiness, we also design parameters to adjust the perturbation size and pattern, which are detailed in §4.3.3.

So far, none of the popular production ALC systems today such as Tesla, GM, etc. [7, 45] identify roads with such dirty road patches as driving scenarios that they do not handle, which can thus further benefit the attack stealthiness.

Motion model based input generation. To address the strong inter-dependencies among the camera frames (**C2**),

Figure 2: Illustration of our novel and domain-specific attack vector: Dirty Road Patch (DRP).

we need to dynamically update the content of later camera frames according to the vehicle actuation decisions applied at earlier ones in the attack generation process. Since adversarial attack generation typically takes thousands of optimization iterations [46, 47], it is practically highly difficult, if not impossible, to drive real vehicles on the target road to obtain such dynamic frame update in every optimization iteration. Another idea is to use vehicle simulators [48, 49], but it requires the attacker to first create a high-definition 3D scene of the target road in the real world, which requires a significant amount of hardware resource and engineering efforts. Also, launching a vehicle simulator in each optimization iteration can greatly harm the attack generation speed.

To efficiently and effectively address this challenge, we combine *vehicle motion model* [36] and *perspective transformation* [50] to dynamically synthesize camera frame updates according to a driving trajectory simulated in a lightweight way. This method is inspired by Google Street View that synthesizes 360° views from a limited number of photos utilizing perspective transformation. Our method only requires one trace of the ALC system inputs (i.e., camera frames) from the target road without attack, which can be easily obtained by the attacker (§3.2).

Optimization-based DRP generation. To systematically generate effective malicious patches, we adopt an optimization-based approach similar to prior works [4, 26]. To address challenge **C3**, we design a novel lane-bending objective function as a differentiable surrogate that aims at changing the derivatives of the desired driving path before the lateral control module, which is equivalent to change the steering angle decisions at the lateral control design level. Besides this, we also have other domain-specific designs in the optimization problem formulation, e.g., for a differentiable construction of the curve fitting process, malicious road patch robustness, stealthiness, and physical-world realizability.

Fig. 3 shows an overview of the malicious road patch generation process, which is detailed in the following sections.

4.2 Motion Model based Input Generation

In Fig. 3, step ①–⑦ belong to the motion model based input generation component. As described earlier in §4.1, the input to this component is a trace of ALC system inputs such as camera frames from driving on the target road without attack. In ①, we apply *perspective transformation*, a widely-used

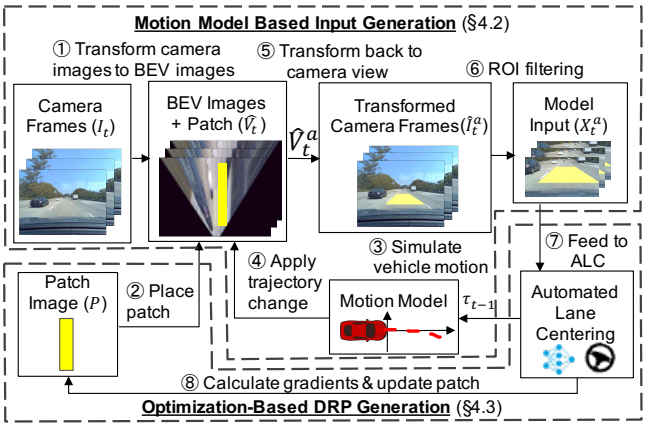

Figure 3: Overview of our DRP (Dirty Road Patch) attack method. ROI: Region of Interest; BEV: Bird's Eye View.

computer vision technique that can project an image view from a 3D coordinate system to a 2D plane [50, 51]. Specifically, we apply it to the original camera frames from the driver's view to obtain their Bird's Eye View (BEV) images. This transformation is highly beneficial since it makes our later patch placement and attack-influenced camera frame updates much more natural and thus convenient. We denote this as $V_t := \text{BEV}(I_t)$, where I_t and V_t are the original camera input and its BEV view respectively at frame t. This process is inversible, i.e., we can also obtain I_t with $\text{BEV}^{-1}(V_t)$.

Next, in ②, we obtain the generated malicious road patch image P from the optimization-based DRP generation step (§4.3) and place it on V_t to obtain the BEV image with the patch, denoted as $\widehat{V}_t := \Lambda(V_t, P)$. To achieve consistent patch placements in the world coordinate across frames, we calculate the *pixel-meter relationship*, i.e., the number of pixels per meter, in BEV images based on the driving trace of the target road. With this, we can place the patch in each frame precisely based on the driving trajectory changes across frames.

Next, we compute the vehicle moving trajectory changes caused by the placed malicious road patch, and reflect such changes in the camera frames. We represent the vehicle moving trajectory as a sequence of vehicle states $S_t := [x_t, y_t, \beta_t, v_t], (t = 1, ..., T)$, where x_t, y_t, β_t, v_t are the vehicle's 2D position, heading angle, and speed at frame t, and T is the total number of frames in the driving trace. Thus, the trajectory change at frame t is $\delta_t := S_t^a - S_t^o$, where S_t^a and S_t^o are vehicle states with and without attack respectively.

To calculate δ_t caused by the attack effect at the frame $t-1$, we need to know the attack-influenced vehicle state S_t^a. To achieve that, we use a *vehicle motion model* to simulate the vehicle state S_t^a by feeding the steering angle decision τ_{t-1} from the lateral control step in the ALC system (§2.1) given the attacked frame at $t-1$ and the previous vehicle state S_{t-1}^a, denoted as $S_t^a := \text{MM}(S_{t-1}^a, \tau_{t-1})$. A vehicle motion model is a set of parameterized mathematical equations representing the vehicle dynamics and can be used to simulate its driving trajectory given the speed and actuation commands. In this

process, we set the vehicle speed as the speed limit of the target road as described in our threat model (§3.2). In our design, we adopt the kinematic bicycle model [52], which is the most widely-used motion model for vehicles [52, 53].

With δ_t, in ④ we then apply affine transformations on the BEV image $\widehat{V}_t$ to obtain the attack-influenced one $\widehat{V}_t^a$, denoted as $\widehat{V}_t^a := T(\widehat{V}_t, \delta_t)$. Fig. 4 shows an example of the shifting and rotation $T(\cdot)$ in the BEV, which synthesizes a camera frame with the vehicle position shifted by 1 meter and rotated by 10° to the right. Although it causes some distortion and missing areas on the edge, the ROI area (red rectangle), i.e., the LD model input, is still complete and thus sufficient for our purpose. Since the ROI area is typically focused on the center and much smaller than the raw camera frame (§2.1), our method can successfully synthesize multiple complete LD model inputs from only 1 ALC system input trace.

Next, in ⑤, we obtain the attack-influenced camera frame at the driver's view $\widehat{I}_t^a$, i.e., the direct input to ALC, by projecting $\widehat{V}_t^a$ back using $\widehat{I}_t^a := \text{BEV}^{-1}(\widehat{V}_t^a)$. Next, in ⑥, the ROI filtering is used to extract the model input $X_t^a := \text{ROI}(\widehat{I}_t^a)$. X_t^a and vehicle state S_t^a are then fed to ALC system in ⑦ to obtain the steering angle decision τ_t, denoted as $\tau_t := \text{ALC}(X_t^a, S_t^a)$. Step ③–⑦ are then iteratively applied to obtain $\widehat{I}_{t+1}^a, \widehat{I}_{t+2}^a, \ldots$ one after one until all the original frames are updated to reflect the moving trajectory changes caused by P. These updated attack-influenced inputs are then fed to the optimization-based DRP generation component, which is detailed next.

4.3 Optimization-Based DRP Generation

In Fig. 3, step ⑧ belongs to the optimization-based road path generation component. In this step, we design a domain-specific optimization process on the target ALC system to systematically generate the malicious dirty road patch P.

DRP attack optimization problem formulation. We formulate the attack as the following optimization problem:

$$\begin{aligned}
\min\ & \mathscr{L} && (1)\\
\text{s.t. } & X_t^a = \text{ROI}(\text{BEV}^{-1}(T(\Lambda(V_t, P), S_t^a - S_t^o))) && (t = 1, \ldots, T) \quad (2)\\
& \tau_t^a = \text{ALC}(X_t^a, S_t^a) && (t = 1, \ldots, T) \quad (3)\\
& S_{t+1}^a = \text{MM}(S_t^a, \tau_t^a) + \varepsilon_t && (t = 1, \ldots, T-1) \quad (4)\\
& S_1^a = S_1^o && (5)\\
& P = \text{BLUR}(\text{FILL}(B) + \Delta) && (6)\\
& \Delta \in \mathscr{P} && (7)
\end{aligned}$$

where the $\mathscr{L}$ in Eq. 1 is an objective function that aims at deviating the victim out of the current lane boundaries as fast as possible (detailed in §4.3.2). Eq. 2–5 have been described in §4.2. In Eq. 6, the patch image $P \in \mathbb{R}^{H \times W \times C}$ consists of a base color $B \in \mathbb{R}^C$ and the perturbation $\Delta \in \mathbb{R}^{H \times W \times C}$, where W, H, and C are the patch image width, height, and the number of color channels respectively. We select an asphalt-like color as the base color B since the image is designed to mimic a road patch. Function FILL: $\mathbb{R}^C \to \mathbb{R}^{H \times W \times C}$ fills B to the entire patch image. Since we aim at generating perturbations that mimic the normal dirty patterns on roads, we restrict Δ to be within a stealthy road pattern space $\mathscr{P}$, which is detailed in §4.3.3. We also include a noise term ε_t in Eq. 4 and an image blurring function BLUR$(\cdot)$ in Eq. 6 to improve the patch robustness to vehicle motion model inaccuracies and camera image blurring, which are detailed in §4.3.4.

4.3.1 Optimization Process Overview

Fig. 5 shows an overview of our iterative optimization process design. Given an initial patch image P, we obtain the model input $X_1^a, \ldots, X_T^a$ from the motion model based input generation process. In step (i), we calculate the gradients of the objective function with respect to $X_1^a, \ldots, X_T^a$, and only keep the gradients corresponding to the patch areas. In step (ii), these gradients are projected into the BEV space. In step (iii), we calculate the average BEV-space gradients weighted by their corresponding patch area sizes in the model inputs. This step involves an approximation of the gradient of $\text{BEV}^{-1}(\cdot)$, which are detailed in our extended version [38]. Next, in step (iv), we update the current patch with Adam [54] using the averaged gradient as the gradient of the patch image. In step (v), we then project the updated patch into the stealthy road pattern space $\mathscr{P}$. This updated patch image is then fed back to the motion model based input generation module, where we also add robustness improvement such as motion noises and image blurring. We terminate this process when the attack-introduced lateral deviations obtained from the motion model are large enough.

4.3.2 Lane-Bending Objective Function Design

As discussed in §4.1, directly using steering angle decisions as $\mathscr{L}$ makes the objective function non-differentiable to $X_1^a, \ldots, X_T^a$. To address this, we design a novel lane-bending objective function $f(\cdot)$ as a differentiable surrogate function. In this design, our key insight is that at the design level, the lateral control step aims at making steering angle decisions that follows a *desired driving path* in the middle of the detected left and right lane line curves from the lane detection step (§2.1). Thus, changing the steering angle decisions is equivalent to changing the derivatives of (or "bending") such desired driving path curve. This allows us to design $f(\cdot)$ as:

$$f(X_1^a, \ldots, X_T^a) = \sum_{t=1}^{T} \sum_{d \in D_t} \nabla \rho_t(d; \{X_j^a | j \le t\}, \theta) + \lambda ||\Omega_t(X_t^a)||_p \quad (8)$$

where $\rho_t(d)$ is a parametric curve whose parameters are decided by (1) both the current and previous model inputs $\{X_j^a | j \le t\}$ due to frame inter-dependencies (§3.3), and (2) the LD DNN parameters θ. D_t is a set of curve point index $d = 0, 1, 2, \ldots$ for the desired driving path curve at frame t. λ is the weight of the p-norm regularization term, designed for stealthiness (§4.3.3). We then can define $\mathscr{L}$ in Eq. 1 as $f(\cdot)$ and $-f(\cdot)$ when attacking to the left and right. Fig. 6 illustrates this surrogate function when attacking to the left. As shown, by maximizing $\nabla \rho_t(d)$ at each curve point in Eq. 8, we can achieve a "lane bending" effect to the desired driving path curve. Since the direct LD output is lane line points (§2.1) but $\rho_t(\cdot)$ require lane line curves, we further perform a differentiable construction of curve fitting process (detailed in our extended version [38]).

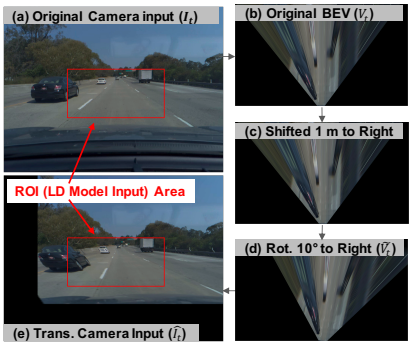

Figure 4: Motion model based input generation from original camera input.

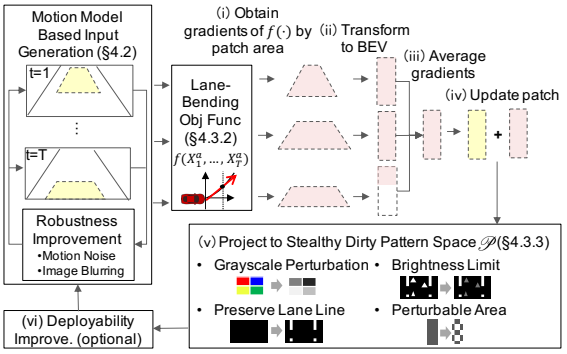

Figure 5: Iterative optimization process design for our optimization-based DRP generation.

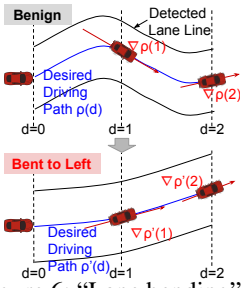

Figure 6: "Lane bending" effect of our objective function by maximizing $\nabla\rho(d)$ at each curve point.

4.3.3 Designs for Dirty Patch Stealthiness

To mimic real-world dirty patterns like in Fig. 2, we have 4 stealthiness designs in stealthy road pattern space $\mathscr{P}$ in Eq. 7:

Grayscale perturbation. Real-world dirty patterns on the road are usually created by dust or white stains (Fig. 2), and thus most commonly just appear white. Thus, we cannot allow perturbations with arbitrary colors like prior works [5]. Thus, our design restricts our perturbation Δ in the grayscale (i.e., black-and-white) by only allowing increase the Y channel in the YCbCr color space [55], denoted as $\Delta_Y \geq 0$.

Preserving original lane line information. We preserve the original lane line information by drawing the same lane lines as the original ones on the patch (if covered by the patch). Note that without this our attack can be easier to succeed, but as discussed in §3.3, it is much more preferred to preserve such information so that the attack deployment can more easily appear as legitimate road work activities and the deployed patch is less likely to be legitimately removed.

Brightness limits. While the dirty patterns are restricted to grayscale, they are still the darker, the stealthier. Also, to best preserve the original lane information, the brightness of the dirty patterns should not be more than the original lane lines. Thus, we (1) add the p-norm regularization term in Eq. 8 to suppress the amount of Δ_Y, and (2) restrict $B_Y + \Delta_Y <$ LaneLine$_Y$, where B_Y and LaneLine$_Y$ are Y channel values for the base color and original lane line color respectively.

Perturbation area restriction. Besides brightness, also the fewer patch areas are perturbed, the stealthier. Thus, we define Perturbable Area Ratio (PAR) as the percentage of pixels on P that can be perturbed. Thus, when PAR=30%, 70% pixels on P will only have the base color B.

4.3.4 Designs for Improving Attack Robustness, Deployability, and Physical-World Realizability

We also have domain-specific designs for improving (1) **attack robustness**, which addresses the driving trajectory/angle deviations and camera sensing inaccuracies in real-world attacks; (2) **attack deployability**, which designs an optional *multi-piece patch attack* mode that allows deploying DRP attack with multiple small and quickly-deployable road patch pieces; and (3) **physical-world realizability**, which addresses the color and pattern distortions due to physical-world factors such as lighting condition, printer color accuracy, and camera color sensing capability. More details are in our extended version [38].

5 Attack Methodology Evaluation

In this section, we evaluate the effectiveness, robustness, generality, and realizability of our DRP attack methodology.

Targeted ALC system. In our evaluation, we perform experiments on the production ALC system in OpenPilot [8], which follows the state-of-the-art DNN-based ALC system design (§2.1). OpenPilot is an open-source production Level-2 driving automation system that can be easily installed in over 80 popular vehicle models (e.g., Toyota, Cadillac, etc.) by mounting a dashcam. We select OpenPilot due to its (1) *representativeness*, since it is reported to have close performance to Tesla Autopilot and GM Super Cruise and better than many others [9], (2) *practicality*, from the large quantity and diversity of vehicle models it can support [8], and (3) *ease to experiment with*, since it is the only production ALC system that is open sourced. In this paper, we mainly evaluate on the lane detection model in OpenPilot v0.7.0, which is released in Dec. 2019. More details of the OpenPilot ALC system are in Appendix C.

Evaluation dataset. We perform experiments using the comma2k19 dataset [56], which contains over 33 hours driving traces between California's San Jose and San Francisco in a Toyota RAV4 2017 driven by human drivers. These traces are collected using the official OpenPilot dashcam device, called EON. From this dataset, we manually look for short free-flow driving periods to make road patch placement convenient. In total, we obtain 40 eligible short driving clips, 10 seconds each, with half of them on the highway, and half on local roads. For each driving clip, we consider two attack scenarios: attack to the left, and to the right. Thus, in total we evaluate 80 different attack scenarios.

5.1 Attack Effectiveness

Evaluation methodology and metrics. We evaluate the attack effectiveness using the evaluation dataset described above. For each attack scenario, we generate an attack road

Figure 7: Driver's view at 2.5 sec (average driver reaction time to road hazards [57]) before our attack succeeds under different stealthiness levels in local road scenarios. Inset figures are the zoomed-in views of the malicious road patches. Larger images are in our extended version [38].

Figure 8: Real-world dirty road patterns.

Figure 9: Stop sign hiding and appearing attacks [5].

patch, and use the motion model based input generation method in §4.2 to simulate the vehicle driving trajectory influenced by the malicious road patch. To judge the attack success, we use the attack goal defined in §3.1 and concrete metrics listed in Table 1, i.e., achieving over 0.735m and 0.285m lateral deviations on highway and local road scenarios respectively within the average driver reaction, 2.5 sec. We measure the achieved deviation by calculating the lateral distances at each time point between the vehicle trajectories with and without the attack, and use the earliest time point to reach the required deviation to calculate the success time.

Since ALC systems assume a human driver who is prepared to take over, it is better if the malicious road patch can also look stealthy enough at 2.5 sec (driver reaction time) before the attack succeeds so that the driver will not be alerted by its looking and decide to take over. Thus, in this section, we also study the stealthiness of the generated road patches. Specifically, we quantify their perturbation degrees using the average pixel value changes from the original road surface in $\mathcal{L}_1, \mathcal{L}_2$ and $\mathcal{L}_{\text{inf}}$ distances [58, 59] and also a user study.

Experimental setup. For each scenario in the evaluation dataset, we manually mark the road patch placement area in the BEV view of each camera frame based on the lane width and shape. To achieve consistent road patch placements in the world coordinate across a sequence of frames, we calculate the number of pixels per meter in the BEV images and adjust the patch position in each frame precisely based on the driving trajectory changes across consecutive frames. The road patch sizes we use are 5.4 m wide, and 24–36 m long to ensure at least a few seconds of visible time at high speed. The patches are placed 7 m far from the victim at the starting frame. For stealthiness levels, we evaluate the $\mathcal{L}_2$ regularisation coefficient $\lambda = 10^{-2}, 10^{-3}$, and 10^{-4}, with PAR set to 50%. According to Eq. 8, larger λ value means more suppression of the perturbation, and thus should lead to a higher stealthiness level. For the motion model, we directly use the vehicle parameters (e.g., wheelbase) of Toyota RAV4 2017, the vehicle model that collects the traces in our dataset.

Results. As shown in Table 2, our attack has high effectiveness ($\geq$97.5%) under all the 3 stealthiness levels. Fig. 7 shows the malicious road patch appearances at different stealthiness levels from the driver's view at 2.5 seconds before our attack succeeds. As shown, even for the lowest stealthiness level ($\lambda = 10^{-4}$) in our experiment, the perturbations are still smaller than some real-world dirty patterns such as the left one in Fig. 8. In addition, the perturbations for all these 3 stealthiness levels are a lot less intrusive than those in previous physical-world adversarial attacks in the image space [5], e.g., in Fig. 9. Among the successful cases, the average success time is all under 0.91 sec, which is substantially lower than 2.5 sec, the required success time. This means that even for a fully attentive human driver who is always able to take over as soon as the attack starts to take effect, the average reaction time is still far from enough to prevent the damage. A more detailed result discussion is in our extended version [38].

Table 2: Attack success rate and time under different stealthiness levels. Larger λ means stealthier. Average success time is calculated only among the successful cases. Pixel $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_{inf}$ are the average pixel value changes from the original road surface in the RGB space and normalized to $[0, 1]$.

Stealth. Level λ	Succ. Rate	Succ. Time (s)	Pixel $\mathcal{L}_1$	Pixel $\mathcal{L}_2$	Pixel $\mathcal{L}_{inf}$
10^{-2}	97.5%	0.903	0.018	0.045	0.201
10^{-3}	100%	0.887	0.033	0.066	0.200
10^{-4}	100%	0.886	0.071	0.109	0.200

Stealthiness user study. To more rigorously evaluate the attack stealthiness, we conduct a user study with 100 participants, and find that (1) even for the lowest stealthiness level at $\lambda = 10^{-4}$, only *less than 25%* of the participants decide to take over the driving before the attack starts to take effect. This suggests that the majority of human drivers today do not treat dirty road patches as road conditions where ALC systems cannot handle; and (2) at 2.5 seconds before the attack succeeds, the attack patches with $\lambda = 10^{-2}$ and 10^{-3} appear to be *as innocent as normal clean road patches to human drivers*, with only less than 15% participants deciding to take over. More detailed results and discussion are in Appendix B.

From these results, the stealthiness level with $\lambda = 10^{-3}$ strikes an ideal balance between attack effectiveness and stealthiness: it does not increase driver suspicion compared to even a benign clean road patch at 2.5 seconds before our attack succeeds, while having no sacrifice of attack effectiveness as shown in Table 2. We thus use it as the default stealthiness configuration in our following experiments.

5.2 Comparison with Baseline Attacks

Evaluation methodology. To understand the benefits of our current design choices over possible alternatives, we evaluate against 2 baseline attack methods: (1) *single-frame EoT attack*, which still uses our lane-bending objective function but optimizes for the EoT (Expectation over Transformation) of the patch view (e.g., different positions/angles) in a single camera frame, and (2) *drawing-lane-line attack*, which directly draws straight solid white lane line instead of placing dirty road patches. EoT is a popular design in prior works to improve attack robustness across sequential frames [29, 30]. Thus, comparing with such a baseline attack can evaluate the benefit of our motion model based input generation design (§4.2) in addressing the challenge of frame inter-dependencies due to attack-influenced vehicle actuation (C2 in §3.3).

The drawing-lane-line attack is designed to evaluate the type of ALC attack vector identified in the prior work by Tencent [6], which uses straightly-aligned white stickers to fool Tesla Autopilot on road regions *without lane lines*. In our case, we perform evaluations in road regions *with lane lines*, and use a more powerful form of it (directly drawing solid lane lines) to understand the upper-bound attack capability of this style of perturbation for ALC systems.

Experimental setup. For single-frame EoT attack, we apply random transformations of the patch in BEV via (1) lateral and longitudinal position shifting. We apply up to $\pm$0.735m and $\pm$0.285m for highway and local respectively, which are their maximum in-lane lateral shifting from the lane center; and (2) viewing angle changes. we apply up to $\pm 5.8^\circ$ changes, the largest average angle deviations under possible real-world trajectory variations based on our experiments (detailed in our extended version [38]). For each scenario, we repeat the experiments for each frame with a complete patch view (usually the first 4 frames), and take the most successful one to obtain the upper-bound effectiveness. Other settings are the same as the DRP attack, e.g., $\lambda = 10^{-3}$.

For the drawing-lane-line attack, we use the same perturbation area (i.e., the patch area) as the others for a fair comparison. Specifically, we sample points every 20cm at the top and bottom patch edges respectively, and form possible attacking lane lines by connecting a point at the top with one at the bottom. We exhaustively try all possible top and bottom point combinations and take the most successful one. The attacking lane lines are 10cm wide (a typical lane marking width [60]) with the same white color as the original lane lines.

Results. Table 3 shows the results under different patch area lengths. As shown, the DRP attack always has the highest attack success rate than these two baselines (with a $\geq$46% margin). When the patch area length is shorter and thus the perturbation capability is more limited, such advantage becomes larger; when the length is 12m, the success rates of single-frame EoT attack and the drawing-lane-line attack drops to 0% and 2.5%, while that for DRP is still 66%. This shows that our motion model based input generation can indeed benefit attack effectiveness, as it can more accurately synthesize subsequent frame content based on attack-influenced vehicle actuation, instead of the blind synthesis in EoT. Also note that the single-frame EoT attack still uses our domain-specific lane-bending objective function design. The drawing-lane-line attack only has 2.5% success rate when the length is 12m; the length used in the Tencent work is actually even shorter (<5m) [6]. This shows that in the road regions *with lane lines*, simply adding lane-line-style perturbations, especially a short one, can barely affect production ALC systems. Instead, an attack vector with larger perturbation area, e.g., in DRP attack, may be necessary.

Table 3: Attack success rates of the DRP attack and 2 baseline attacks under different patch area lengths.

Attack	Patch Area Length			
	12m	18m	24m	36m
DRP	66.25%	82.50%	90.75%	100%
Single-frame EoT	0.00%	8.75%	21.25%	50.00%
Drawing-lane-line	2.50%	13.75%	31.25%	53.75%

5.3 Attack Robustness, Generality, and Deployability Evaluations

Robustness to run-time driving trajectory and angle deviations. As described in §4.3.4, the run-time victim driving trajectories and angles will be different from the motion model predicted ones in attack generation time due to run-time driving dynamics. To evaluate attack robustness against such deviations, we use (1) 4 levels of vehicle position shifting at each vehicle control step in attack evaluation time, and (2) 27 vehicle starting positions to create a wide range of approaching angles and distances to the patch, e.g., from (almost) the leftmost to the rightmost position in the lane. Our attack is shown to maintains a high effectiveness ($\geq$ 95% success rate) even when the vehicle positions at the attack evaluation time has 1m shifting on average from those at the attack generation time at each control step. Details are in our extended version [38].

Attack generality evaluation. To evaluate the generality of our attack against LD models of different designs, ideally we hope to evaluate on LD models from other production ALC besides OpenPilot, e.g., from Tesla Autopilot. However, OpenPilot is the only one that is currently open sourced. Fortunately, we find that the LD models in some older versions of OpenPilot actually have different DNN designs, which thus can also serve for our purpose. We evaluate on 3 versions of LD models with large DNN architecture differences, and find that our attack is able to achieve $\geq$90% success rates against all 3 LD models, with an average attack transferability of 63%. More details are in our extended version [38].

Attack deployability evaluation. We evaluate the attack deployability by estimating the required efforts to deploy the attack road patch. We perform experiments using our multi-piece patch attack mode design (§4.3.4), and find that the attack success rate can be *93.8% with only 8 pieces of quickly-*

deployable road patches, each requiring only 5-10 sec for 2 people to deploy based on videos of adhesive patch deployment [61]. More details are in our extended version [38].

5.4 Physical-World Realizability Evaluation

While we have shown high attack effectiveness, robustness, and generality on real-world driving traces, the experiments are performed by synthesizing the patch appearances digitally, which is thus still different from the patch appearances in the physical world. As discussed in §4.3.4, there are 3 main practical factors that can affect the attack effectiveness in physical world: (1) the lighting condition, (2) printer color accuracy, and (3) camera sensing capability. Thus, in this section we perform experiments to understand the physical-world attack realizability against these 3 main practical factors.

Evaluation methodology: miniature-scale experiments. To perform the DRP attack, a real-world attacker can pretend to be road workers and place the malicious road patch on public roads. However, due to the access limit to private testing facilities, we cannot do so ethically and legally on public roads with a real vehicle. Thus, we try our best to perform such evaluation by designing a *miniature-scale experiment*, where the road and the malicious road patch are first physically printed out on papers and placed according to the physical-world attack settings but in miniature scale. Then the real ALC system camera device is used to get camera inputs from such a miniature-scale physical-world setting. Such miniature-scale evaluation methodology can capture all the 3 main practical factors in the physical-world attack setting, and thus can sufficiently serve for the purpose of this evaluation.

Experimental setup. As shown in Fig. 10, we create a miniature-scale road by printing a real-world high-resolution BEV road texture on multiple ledger-size papers and concatenating them together to form a long straight road. In the attack evaluation time, we create the miniature-scale malicious road patch using the same method, and place it on top of the miniature-scale road following our DRP attack design. The patch is printed with a commodity printer: RICOH MP C6004ex Color Laser Printer. We mount EON, the official OpenPilot dashcam device, on a tripod and face it to the miniature-scale road. The road size, road patch size, and the EON mounting position are carefully calculated to represent OpenPilot installed on a Toyota RAV4 driving on a standard 3.6-meter wide highway road at 1:12 scale. We also create different lighting conditions with two studio lights. The patch size is set to represent a 4.8me wide and 12m long one in the real world scale. The other settings are the same as in §5.3.

Evaluation metric. Since the camera is mounted in a static position, we evaluate the attack effectiveness directly using the steering angle decision at the frame level instead of the lateral deviation used in previous sections. This is equivalent from the attack effectiveness point of view since the large lateral deviation is essentially created by a sequence of large steering angle decisions at the frame level. Specifically, we first find the camera frame that has the same relative position between the camera and the patch as that in the miniature-scale experimental setup. Then we compare its *designed* steering angle at the attack generation time and its *observed* steering angle that the ALC system in OpenPilot intends to apply to the vehicle in the miniature-scale experiment. Thus, the more similar these two steering angles are, the higher realizability our attack has in the physical world.

Results. Fig. 11 shows a visualization of the lane detection results of the benign and attacked scenarios in the miniature-scale experiment using the OpenPilot's official visualization tool. As shown, in the benign scenario, both detected lane lines align accurately with the actual lane lines, and the desired driving path is straight as expected. However, when the malicious road patch is placed, it bends the detected lane lines significantly to the left and causes the desired driving path to be curving to the left, which is exactly the designed attack effect of our lane-bending objective function (§4.3.2). In this case, the designed steering angle is 23.4° to the left at the digital attack generation time, and the observed one in the physical miniature-scale experiment is 24.5° to the left, which only differs by 4.7%. In contrast, in the benign scenario the observed steering angle for the same frame is 0.9° to the right.

Robustness under different lighting conditions. We repeat this experiment under 12 lighting conditions ranging from 15 lux (corresponding to sunset/sunrise) to 1210 lux (corresponding to midday of overcast days). The results show that the same attack patch above is able to *maintain a desired steering angle of 20-24° to the left under all 12 lighting conditions*, which are all significantly different from the benign case (0.9° to right). Details are in our extended version [38].

Robustness to different viewing angles. We evaluate the robustness from 45 different viewing angles created by different distances to the patch and lateral offsets to the lane center. Our results show that our attack always achieves over 23.4° to the left from all viewing angles. We record videos in which we dynamically change viewing angles in a wide range while showing real-time lane detection results under attack, available at **https://sites.google.com/view/cav-sec/drp-attack/**.

6 Software-in-the-Loop Simulation

To understand the safety impact, we perform software-in-the-loop evaluation on LGSVL, a production-grade autonomous driving simulator [48]. We overcame several engineering challenges in enabling this setup, which are detailed in our extended version [38] and open-sourced via our website [10].

Evaluation scenarios. We construct 2 attack scenarios for highway and local road settings respectively, as shown in Fig. 12. For the former, we place a concrete barrier on the left, and for the latter, we place a truck driving on an opposite direction lane. The attack goals are to hit the concrete barrier or the truck. Detailed setup are in Table 4.

Experimental setup and evaluation metrics. We perform evaluation on OpenPilot v0.6.6 with the Toyota RAV4

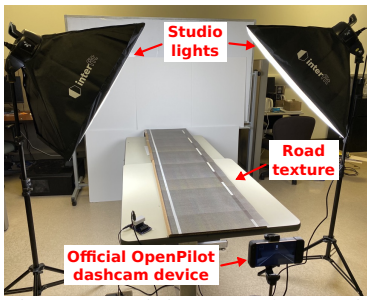

Figure 10: Miniature-scale experiment setup. Road texture/patch are printed on ledger-size papers.

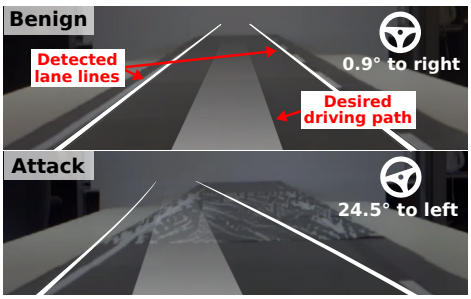

Figure 11: Lane detection and steering angle decisions in benign and attacked scenarios in the miniature-scale experiment.

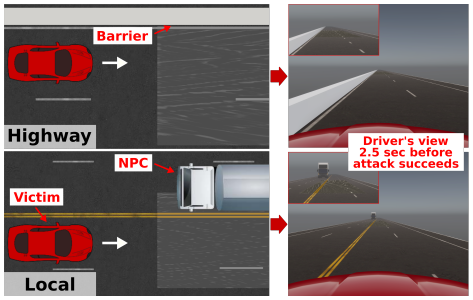

Figure 12: Software-in-the-loop simulation scenarios and driver's view 2.5 sec before attack succeeds. Larger images are in [38]

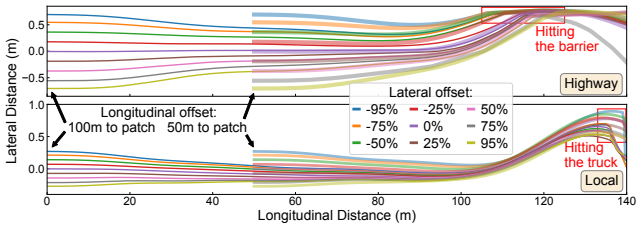

Figure 13: Victim driving trajectories in the software-in-the-loop evaluation from 18 different starting positions for highway and local road scenarios. Lateral offset values are percentages of the maximum in-lane lateral shifting from lane center; negative and positive signs mean left and right shifting.

parameters. We follow the methodology in §4.3.4 to obtain and apply the color mapping in our simulation environment. The patch size is 5.4m wide and 70m long, and we place it in the simulation environment by importing the generated patch image into Unity. The other parameters are the same as §5.3. To evaluate the attack effectiveness from different victim approaching angles, for each scenario we evaluate the same patch from 18 different starting positions, created from the combinations of 2 longitudinal distances to the patch (50 and 100 m) and 9 lateral offsets (from -95% to 95%) as shown in Fig. 13. The patch is visible at all these starting positions. We repeat 10 times for each starting position in each scenario.

Results and video demos. Our attack achieves 100% success rates from all 18 starting positions in both highway and local road scenarios as shown in Table 4. Fig. 13 shows the averaged vehicle trajectories from each starting positions. As shown, the vehicle always first drives toward the lane center since the ALC system tries to correct the initial lateral deviations. After that, the patch starts to take effect, and causes the vehicle to deviate to the left significantly and hit the barrier or truck. We record demo videos at **https://sites.google.com/view/cav-sec/drp-attack/**. In the highway scenario, after the victim hits the concrete barrier, it bounces away quickly due to the abrupt collision. For local road, the victim crashes to the front of the truck, causing both the victim and truck to stop. This suggests that the safety impacts of our attack can be severe.

Table 4: Simulation scenario configurations and evaluation results. Lane widths and vehicle speeds are based on standard/common ones in the U.S. [62]. Simulation results without attack are confirmed to have 0% success rates with ≤0.018 m (std: ≤9e-4) average maximum deviations.

Sim. Scenario	Lane Width	Veh. Speed	Attack Goal	Ave. Max Dev. (std)	Succ. Rate	Succ. Time
Highway	3.6 m	65 mph (29 m/s)	Hit barrier on the left	0.76 m (5e-3)	100% (100/100)	0.97 s
Local	2.7 m	45 mph (20 m/s)	Hit truck in the opposite lane	0.55 m (7e-2)	100% (100/100)	1.36 s

7 Safety Impact on Real Vehicle

While the simulation-based evaluation above has shown severe safety impacts, it does not simulate other driver assistance features that are commonly used with ALC at the same time in real-world driving, for example Lane Departure Warning (LDW), Adaptive Cruise Control (ACC), Forward Collision Warning (FCW), and Automatic Emergency Braking (AEB). This makes it unclear whether the safety damages shown in §6 are still possible when these features are used, especially the safety-protection ones such as AEB. In this section, we thus use a real vehicle to more directly understand this.

Evaluation methodology. We install OpenPilot on a Toyota 2019 Camry, in which case OpenPilot provides ALC, LDW, and ACC, and the Camry's stock features provide AEB and FCW [8]. We then use this real-world driving setup to perform experiments on a rarely-used dead-end road, which has a double-yellow line in the middle and can only be used for U-turn. The driver's view of this road is shown on the left of Fig. 14. In our miniature-scale experiment in §5.4, the attack realizability from the physically-printed patch to the LD model output has already been validated under 12 different lighting conditions. Thus, in this experiment we evaluate the safety impact by directly injecting an attack trace at the LD model output level (detailed in Appendix C). This can also avoid blocking the road for sticking patches to the ground and cleaning them up, which may affect other vehicles.

To create safety-critical driving scenarios, we place cardboard boxes adjacent to but outside of the current lane as shown in Fig. 14, which can mimic road barriers and obstacles in opposite direction as in §6 while not causing damages

Figure 14: Safety impact evaluation for our attack on a Toyota 2019 Camry with OpenPilot engaged. Even with other driver assistance features such as Automatic Emergency Braking (AEB), our attack still causes collisions in all the 10 trials.

to the vehicle and driver safety. Similar setup is also used in today's vehicle crash tests [63–66]. To ensure that we do not affect other vehicles, we place the cardboard boxes only when the entry point of this dead-end road has no other driving vehicles in sight, and quickly remove them right after our vehicle passes them as required by the road code of conduct [67].

Experiment setup. We perform experiments in day time with and without attack, each 10 times. The driving speed is kept at ∼28 mph (∼45 km/h), the min speed for engaging OpenPilot on our Camry. The injected attack trace is from our simulation environment (§6) at the same driving speed.

Results. Our experiment results show that our attack causes the vehicle to hit the cardboard boxes in all the 10 attack trials (100% collision rate), including 5 front and 5 side collisions. The collision variations are caused by randomness in the dynamic vehicle control and the timing differences in OpenPilot engaging and attack launching. In contrast, in the trials without attack, OpenPilot can always drive correctly and does not hit or even touch the objects in any of the 10 trials.

These results thus show that driver assistance features such as LDW, ACC, FCW, and AEB are not able to effectively prevent the safety damages caused by our attack on ALC. We examine the attack process and find that LDW is not triggered since it relies on the same lane detection module as ALC and thus are affected simultaneously by our attack. ACC does not take any action since it does not detect a front vehicle to follow and adjust speed in these experiments. FCW is triggered 5 times out of the 10 collisions, but it is only a warning and thus cannot prevent the collision by itself. Moreover, in our experiments FCW is triggered only *0.46 sec* before the collision on average, which is far too short to allow human drivers to react considering the 2.5-second average driver reaction time to road hazard (§3).

In our Camry model, FCW and AEB are turned on together as a bundled safety feature [68]. However, while we have observed some triggering of FCW, we were not able to observe any triggering of AEB among the 10 attack trials, leading to a *100% false negative rate*. We check the vehicle manual [68] and find that this may be because the AEB feature (called *pre-collision braking* for Toyota) is used very conservatively: it is triggered only when the possibility of a collision is *extremely high*. This observation is also consistent with the previously-reported high failure rate (60%) for AEB features on popular car models today [35]. Such conservative use of AEB can reduce false alarms and thus avoid mistaken sudden emergency brakes in normal driving, but also makes it difficult to effectively preventing the safety damages caused by our attack — in our experiments, it was not able to prevent any of the 10 collisions. The video recordings for these real-vehicle experiments are available at **https://sites.google.com/view/cav-sec/drp-attack/**.

8 Limitations and Defense Discussion

8.1 Limitations of Our Study

Attack deployability. As evaluated in §5.3, our attack can achieve a high success rate (93.8%) with only 8 pieces of quickly-deployable road patches, each requiring only 5-10 sec to deploy for 2 people. To further increase stealthiness, the attacker can pretend to be road workers like in Fig. 2 to avoid suspicion, and pick a deployment time when the target road is the most vacant, e.g., at late night. Nevertheless, lower deployment efforts is always more preferred for attackers to reduce risks. One potential direction to further improve this is to explore other common road surface patterns besides dirty patterns, which we leave as future work.

Generality evaluation. Although we have shown high attack generality against LD models with different designs (§5.3), all our evaluations are performed on only one production ALC in OpenPilot. Thus, it is still unclear whether other popular ALC, e.g., Tesla Autopilot and GM Cruise, are vulnerable to our attack. Unfortunately, to the best of our knowledge, the OpenPilot ALC is the only production one that is open sourced. Due to the same reason, we are also unable to evaluate the transfer attacks from OpenPilot to these other popular ALC systems. Nevertheless, since the OpenPilot ALC is representative at both design and implementation levels (§5), we think our current discovery and results can still generally benefit the understanding of the security of production ALC today. Also, since DNNs are generally vulnerable to adversarial attacks [4, 5, 26, 27, 29–31, 46], if these other ALC systems also adopt the state-of-the-art DNN-based design, at least at design level they are also vulnerable to our attack.

End-to-end evaluation in real world. In this work, we evaluate our attack against various possible real-world factors such as lighting conditions, patch viewing angles, victim approaching angles/distances, printer color accuracy, and camera sensing capability (§5.3 and §5.4), and also evaluate the safety impact using software-in-the-loop simulation (§6) and attack trace injection in a real vehicle (§7). However, these setups still have a gap to real-world attacks as we did not perform direct end-to-end attack evaluation with real vehicles in the physical world. Such a limitation is caused by safety issues (vehicle-enforced minimum OpenPilot engagement speed at 28 mph, or 45 km/h) and access limits to private testing facilities (for patch placement). In the future, we hope to overcome this by finding ways to lower the minimum engagement speed and obtain access to private testing facilities.

8.2 Defense Discussion

8.2.1 Machine Learning Model Level Defenses

In the recent arms race between adversarial machine learning attacks and defenses, numerous defense/mitigation techniques have been proposed [69–72]. However, so far none of them studied LD models. As a best effort to understand the effectiveness of existing defenses on our attack, we perform evaluation on 5 popular defense methods that only require model input transformation without re-training: JPEG compression [73], bit-depth reduction [71], adding Gaussian noise [74], median blurring [71], and autoencoder reformation [75], since they are directly applicable to LD models. Descriptions and our configurations of these methods are in our extended version [38]. Our experiments use the same dataset and success metrics as in §5. Meanwhile, we also evaluate a *benign-case success rate*, defined as the percentage of scenarios where the ALC can behave correctly (i.e., not driving out of lane) when the defense method is applied.

Fig. 15 shows the evaluation results. As shown, for each defense method we also vary the parameters to explore the trade-off between attack success rate and benign-case success rate. As shown, while all methods can effectively decrease the attack success rate with certain parameter configurations, the benign-case success rates are also decreased at the same time. In particular, when the benign-case success rates are still kept at 100%, the attack success rates are still 99 to 100% for all methods. This shows that *none of these methods can effectively defend against our attack without harming ALC performance in normal driving scenarios*. This might be because these defenses are mainly for disrupting digital-space human-imperceptible perturbations, and thus are less effective for physical-world realizable attacks with human-perceptible (but seemingly-benign) perturbations.

These results show that directly-applicable defense methods cannot easily defeat our attack. Thus, it is necessary to explore (1) novel adaptions of more advanced defenses such as adversarial training to LD, or (2) new defenses specific to LD and our problem setting, which we leave as future work.

8.2.2 Sensor/Data Fusion Based Defenses

Besides securing LD models, another direction is to fuse camera-based lane detection with other independent sensor/data sources such as LiDAR and High Definition (HD) map [76]. For example, LiDAR can capture the tiny laser reflection differences for lane line markings, and thus is possible to perform lane detection [77]. However, while LiDARs are commonly used in high-level (e.g., Level-4) AD systems such as Google Waymo [78] that provide self-driving taxi/truck, so far they are not generally used in production low-level (e.g., Level-2) AD such as ALC, e.g., Tesla, GM Cadillac, Toyota RAV4, etc. [3, 45, 79]. This is mainly because LiDAR is quite costly for vehicle models sold to individuals (typically $\geq$\$4,000 each for AD [80]). For example, Elon Musk, the co-founder of Tesla, claims that LiDARs are "*expensive sensors that are unnecessary (for autonomous vehicles)*" [81].

Another possible fusion source is lane information from a pre-built HD map of the targeted road, which can be used to cross-check with the run-time detected lane lines to detect our attack. However, this requires ALC providers to collect and maintain accurate lane line information for each road, which can be time consuming, costly, and also hard to scale. To the best of our knowledge, ALC systems in production Level-2 AD systems today do not use HD maps in general. For instance, Tesla explicitly claims that it does not use HD map for Autopilot driving since it is a *"non-scalable approach"* [82].

Nevertheless, considering that Level-4 AD systems today are able to build and heavily utilize HD maps [83, 84], we think leveraging HD maps is still a more feasible solution than requiring production Level-2 vehicle models to install LiDARs. If such a map can be available, a follow-up research question is how to effectively detect our attack without raising too many false alarms, since mismatched lane information can also occur in benign cases due to (1) vehicle position and heading angle inaccuracies when localized on the HD map, e.g., due to sensor noises in GPS and IMU, and (2) normal-case LD model inaccuracies.

9 Related Work

Autonomous Driving (AD) system security. For AD systems, there are mainly two types of security research: *sensor security* and *autonomy software security*. For *sensor security*, prior works studied spoofing/jamming on camera [85–87], LiDAR [31, 85, 88], RADAR [86], ultrasonic [86], and IMU [89]. For *autonomy software security*, prior works have studied the security of object detection [4, 5, 88], tracking [90], localization [91], traffic light detection [92], and end-to-end AD models [41, 43]. Our work studies autonomy software security in production ALC. The only prior effort is from Tencent [6], but it neither attacks the designed operational domain for ALC (i.e., roads with lane lines), nor generates perturbations systematically by addressing the design challenges in §3.3.

Physical-world adversarial attacks. Multiple prior works have explored image-space adversarial attacks in the physical world [4, 5, 28–30]. In particular, various techniques have been designed to improve the physical-world robustness, e.g., non-printability score [4, 93–95], low-saturation colors [5], and EoT [4, 5, 29, 30]. In comparison, prior efforts concentrate on image classification and object detection, while we are the first to systematically design physical-world adversarial attacks on ALC, which require to address various new and unique design challenges (§3.3).

10 Conclusion

In this work, we are the first to systematically study the security of DNN-based ALC in its designed operational domains under physical-world adversarial attacks. With a novel attack vector, dirty road patch, we perform optimization-based attack generation with novel input generation and objective function

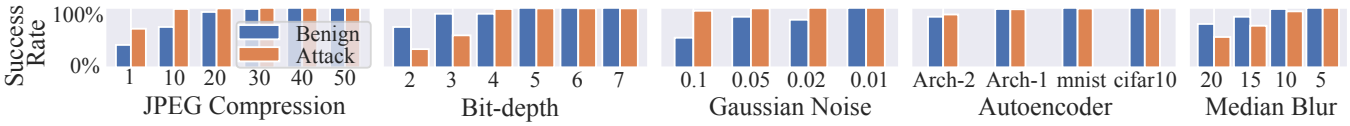

Figure 15: Evaluation results for 5 directly-applicable DNN model level defense methods. *Attack*: Attack success rate. *Benign*: Percentage of scenarios where the ALC can still behave correctly (i.e., not driving out of current lane) with defense applied.

designs. Evaluation on a production ALC using real-world traces shows that our attack has over 95% success rates with success time substantially lower than average driver reaction time, and also has high robustness, generality, physical-world realizability, and stealthiness. We further conduct experiments using both simulation and a real vehicle, and find that our attack can cause a 100% collision rate in different scenarios. We also evaluate and discuss possible defenses. Considering the popularity of ALC and the safety impacts shown in this paper, we hope that our findings and insights can bring community attention and inspire follow-up research.

Acknowledgements

We would like to thank Ziwen Wan, Chen Wang, and the anonymous reviewers for valuable feedback on our work. This research was supported in part by the National Science Foundation under grants CNS-1850533, CNS-1929771, CNS-1932351, CNS-1932464, and USDOT grant 69A3552047138 for CARMEN UTC (University Transportation Center).

References

[1] "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," *SAE International*, 2016.

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

[3] "Tesla Autopilot." https://www.tesla.com/autopilot.

[4] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust Physical-World Attacks on Deep Learning Visual Classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[5] Y. Zhao, H. Zhu, R. Liang, Q. Shen, S. Zhang, and K. Chen, "Seeing isn't Believing: Practical Adversarial Attack Against Object Detectors," in *ACM SIGSAC Conference on Computer and Communications Security (ACM CCS)*, p. 1989–2004, 2019.

[6] "Experimental Security Research of Tesla Autopilot." https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Research_of_Tesla_Autopilot.pdf, 2019.

[7] "Tesla Autopilot Support." https://www.tesla.com/support/autopilot.

[8] "OpenPilot." https://github.com/commaai/openpilot.

[9] "Is a $1000 Aftermarket Add-On as Capable as Tesla's Autopilot and Cadillac's Super Cruise?." https://www.caranddriver.com/features/a30341053/self-driving-technology-comparison/, 2020.

[10] "Dirty Road Patch Attack Project Website." https://sites.google.com/view/cav-sec/drp-attack.

[11] "Lane Keeping Assist System Using Model Predictive Control." https://www.mathworks.com/help/mpc/ug/lane-keeping-assist-system-using-model-predictive-control.html, 2020.

[12] J.-W. Lee and B. Litkouhi, "A Unified Framework of the Automated Lane Centering/Changing Control for Motion Smoothness Adaptation," in *International IEEE Conference on Intelligent Transportation Systems*, pp. 282–287, 2012.

[13] "Super Cruise - Hands Free Driving | Cadillac Ownership." https://www.cadillac.com/world-of-cadillac/innovation/super-cruise.

[14] Z. Wang, W. Ren, and Q. Qiu, "LaneNet: Real-Time Lane Detection Networks for Autonomous Driving," *arXiv:1807.01726*, 2018.

[15] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as Deep: Spatial CNN for Traffic Scene Understanding," in *AAAI Conference on Artificial Intelligence*, 2018.

[16] Y. Ko, J. Jun, D. Ko, and M. Jeon, "Key Points Estimation and Point Instance Segmentation Approach for Lane Detection," *arXiv:2002.06604*, 2020.

[17] J. Li, X. Mei, D. Prokhorov, and D. Tao, "Deep Neural Network for Structural Prediction and Lane Detection in Traffic Scene," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 690–703, 2016.

[18] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, and Q. Wang, "Robust Lane Detection From Continuous Driving Scenes Using Deep Neural Networks," *IEEE Transactions on Vehicular Technology*, 2019.

[19] P. Smuda, R. Schweiger, H. Neumann, and W. Ritter, "Multiple Cue Data Fusion With Particle Filters for Road Course Detection in Vision Systems," in *IEEE Intelligent Vehicles Symposium (IV)*, 2006.

[20] C. Gackstatter, P. Heinemann, S. Thomas, and G. Klinker, "Stable Road Lane Model Based on Clothoids," in *Advanced Microsystems for Automotive Applications*, pp. 133–143, Springer, 2010.

[21] S. Yenikaya, G. Yenikaya, and E. Düven, "Keeping the Vehicle on the Road - A Survey on On-Road Lane Detection Systems," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, pp. 1–43, 2013.

[22] C. Becker, L. J. Yount, S. Rozen-Levy, and J. D. Brewer, "Functional Safety Assessment of an Automated Lane Centering System," in *National Highway Traffic Safety Administration*, 2018.

[23] R. C. Dorf and R. H. Bishop, *Modern Control Systems*. Pearson, 2011.

[24] Richalet, J. and Rault, A. and Testud, J. L. and Papon, J., "Model Predictive Heuristic Control," *Automatica*, vol. 14, p. 413–428, 1978.

[25] "Tinkla: Tinkering with Tesla." https://tinkla.us/.

[26] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing Properties of Neural Networks," in *International Conference on Learning Representation (ICLR)*, 2014.

[27] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *arXiv:1412.6572*, 2014.

[28] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial Examples in the Physical World," *arXiv:1607.02533*, 2016.

[29] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing Robust Adversarial Examples," in *International Conference on Machine Learning (ICML)*, 2018.

[30] T. Brown, D. Mane, A. Roy, M. Abadi, and J. Gilmer, "Adversarial Patch," *arXiv:1712.09665*, 2017.

[31] Y. Cao, N. Wang, C. Xiao, D. Yang, J. Fang, R. Yang, Q. A. Chen, M. Liu, and B. Li, "Invisible for both Camera and LiDAR: Security of Multi-Sensor Fusion based Perception in Autonomous Driving Under Physical-World Attacks," in *IEEE Symposium on Security and Privacy (SP)*, 2021.

[32] D. Zhao, Y. Guo, and Y. J. Jia, "Trafficnet: An Open Naturalistic Driving Scenario Library," in *IEEE International Conference on Intelligent Transportation Systems*, pp. 1–8, 2017.

[33] A. Boora, I. Ghosh, and S. Chandra, "Identification of Free Flowing Vehicles on Two Lane Intercity Highways under Heterogeneous Traffic condition," *Transportation Research Procedia*, vol. 21, pp. 130–140, 2017.

[34] "California Vehicle Code 21663." https://leginfo.legislature.ca.gov/faces/codes_displaySection.xhtml?lawCode=VEH&sectionNum=21663, 1959.

[35] "Does Your Car Have Automated Emergency Braking? It's a Big Fail for Pedestrians." https://www.zdnet.com/article/does-your-car-have-automated-emergency-braking-its-a-big-fail-for-pedestrians/, 2019.

[36] R. Rajamani, *Vehicle Dynamics and Control*. Springer Science & Business Media, 2011.

[37] "California Penal Code 594." https://leginfo.legislature.ca.gov/faces/codes_displaySection.xhtml?lawCode=PEN&sectionNum=594, 1872.

[38] T. Sato, J. Shen, N. Wang, Y. J. Jia, X. Lin, and Q. A. Chen, "Dirty Road Can Attack: Security of Deep Learning based Automated Lane Centering under Physical-World Attack," *arXiv:2009.06701*, 2021.

[39] S. Li, A. Neupane, S. Paul, C. Song, S. V. Krishnamurthy, A. K. Roy-Chowdhury, and A. Swami, "Stealthy Adversarial Perturbations Against Real-Time Video Classification Systems," in *Annual Network and Distributed System Security Symposium (NDSS)*, 2019.

[40] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated Whitebox Testing of Deep Learning Systems," in *Symposium on Operating Systems Principles*, pp. 1–18, 2017.

[41] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars," in *International Conference on Software Engineering*, pp. 303–314, 2018.

[42] A. Chernikova, A. Oprea, C. Nita-Rotaru, and B. Kim, "Are Self-Driving Cars Secure? Evasion Attacks Against Deep Neural Networks for Steering Angle Prediction," in *IEEE Security and Privacy Workshops (SPW)*, pp. 132–137, 2019.

[43] H. Zhou, W. Li, Y. Zhu, Y. Zhang, B. Yu, L. Zhang, and C. Liu, "Deepbillboard: Systematic Physical-World Testing of Autonomous Driving Systems," in *International Conference on Software Engineering*, 2020.

[44] "Adhesive Patch can Seal Potholes and Cracks on the Road." https://www.startupselfie.net/2019/05/07/american-road-patch-seals-potholes-road-cracks/, 2019.

[45] "GM Cadillac CT6 Owner's Manual." https://www.cadillac.com/content/dam/cadillac/na/us/english/index/ownership/technology/supercruise/pdfs/2020-cad-ct6-owners-manual.pdf, 2019.

[46] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," in *IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, 2017.

[47] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," in *International Conference on Learning Representation (ICLR)*, 2018.

[48] "LGSVL Simulator: An Autonomous Vehicle Simulator." https://github.com/lgsvl/simulator/.

[49] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Annual Conference on Robot Learning*, 2017.

[50] S. Tanaka, K. Yamada, T. Ito, and T. Ohkawa, "Vehicle Detection Based on Perspective Transformation Using Rear-View Camera," *Hindawi Publishing Corporation International Journal of Vehicular Technology*, vol. 9, 03 2011.

[51] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 ed., 2003.

[52] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design," in *IEEE Intelligent Vehicles Symposium (IV)*, pp. 1094–1099, 2015.

[53] D. Watzenig and M. Horn, *Automated Driving: Safer and More Efficient Future Driving*. Springer, 2016.

[54] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representation (ICLR)*, 2015.

[55] E. Hamilton, "JPEG File Interchange Format," 2004.

[56] H. Schafer, E. Santana, A. Haden, and R. Biasini, "A Commute in Data: The comma2k19 Dataset," *arXiv:1812.05752*, 2018.

[57] S. of California Department of Motor Vehicles, *California Commercial Driver Handbook: Section 2 – Driving Safely*. 2019. Available at https://www.dmv.ca.gov/portal/uploads/2020/06/comlhdbk.pdf.

[58] X. Ling, S. Ji, J. Zou, J. Wang, C. Wu, B. Li, and T. Wang, "Deepsec: A Uniform Platform for Security Analysis of Deep Learning Model," in *IEEE Symposium on Security and Privacy (SP)*, pp. 673–690, 2019.

[59] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples," in *International Conference on Machine Learning (ICML)*, 2018.

[60] "Manual on Uniform Traffic Control Devices Part 3 Markings." https://mutcd.fhwa.dot.gov/pdfs/millennium/06.14.01/3ndi.pdf, 2020.

[61] "American Road Patch - Deployment Demonstration Video from 4:54 to 5:04." https://youtu.be/Vr_Dxg1LdxU?t=294, 2019.

[62] A. A. of State Highway and T. O. (AASHTO), *Policy on Geometric Design of Highways and Streets (7th Edition)*. American Association of State Highway and Transportation Officials (AASHTO), 2018.

[63] "Toyota Safety Sense Pre-Collision System (PCS) Settings and Controls." https://youtu.be/IY4g_zG1Qj0, 2017.

[64] "Honda's Collision Mitigation Braking System CMBS." https://youtu.be/NJcy5ySOrM4, 2013.

[65] "IIHS Issues First Crash Avoidance Ratings Under New Test Program." https://www.iihs.org/news/detail/iihs-issues-first-crash-avoidance-ratings-under-new-test-program, 2013.

[66] "Collision Avoidance Strikeable Targets for AEB." http://www.pedstrikeabletargets.com/, 2020.

[67] "California Vehicle Code 23113." https://leginfo.legislature.ca.gov/faces/codes_displaySection.xhtml?lawCode=VEH&sectionNum=23113, 2000.

[68] "Toyota 2019 Camry Owner's Manual." https://www.toyota.com/t3Portal/document/om-s/OM06142U/pdf/OM06142U.pdf, 2019.

[69] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense Against Adversarial Attacks Using High-Level Representation Guided Denoiser," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[70] C. Xie, Y. Wu, L. v. d. Maaten, A. L. Yuille, and K. He, "Feature Denoising for Improving Adversarial Robustness," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[71] W. Xu, D. Evans, and Y. Qi, "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks," *arXiv:1704.01155*, 2017.

[72] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified Adversarial Robustness via Randomized Smoothing," in *International Conference on Machine Learning (ICML)*, pp. 1310–1320, 2019.

[73] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A Study of the Effect of JPG Compression on Adversarial Images," *arXiv:1608.00853*, 2016.

[74] Y. Zhang and P. Liang, "Defending Against Whitebox Adversarial Attacks via Randomized Discretization," in *International Conference on Artificial Intelligence and Statistics*, vol. 89, pp. 684–693, 2019.

[75] D. Meng and H. Chen, "Magnet: a Two-pronged Defense Against Adversarial Examples," in *ACM SIGSAC Conference on Computer and Communications Security (ACM CCS)*, pp. 135–147, 2017.

[76] "HD Maps: New Age Maps Powering Autonomous Vehicles." https://www.geospatialworld.net/article/hd-maps-autonomous-vehicles/, 2017.

[77] M. Bai, G. Mattyus, N. Homayounfar, S. Wang, S. K. Lakshmikanth, and R. Urtasun, "Deep Multi-Sensor Lane Detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3102–3109, 2018.

[78] "Waymo Has Launched its Commercial Self-Driving Service in Phoenix — and it's Called 'Waymo One'." https://www.businessinsider.com/waymo-one-driverless-car-service-launches-in-phoenix-arizona-2018-12, 2018.

[79] "Toyota 2020 RAV4 Owner's Manual." https://www.toyota.com/t3Portal/document/om-s/OM0R024U/xhtml/OM0R024U.html.

[80] "Velodyne Just Cut the Price of Its Most Popular Lidar Sensor in Half." https://www.thedrive.com/tech/17297/velodyne-just-cut-the-price-of-its-most-popular-lidar-sensor-in-half, 2018.

[81] "'Anyone Relying on Lidar is Doomed,' Elon Musk Says." https://techcrunch.com/2019/04/22/anyone-relying-on-lidar-is-doomed-elon-musk-says/, 2019.

[82] "Tesla Admits its Approach to Self-Driving is Harder But Might be Only Way to Scale." https://electrek.co/2020/06/18/tesla-approach-self-driving-harder-only-way-to-scale/, 2020.

[83] "Building Maps for a Self-Driving Car." https://link.medium.com/Bo5pCOov95, 2016.

[84] "Baidu Apollo HD Map." http://ggim.un.org/unwgic/presentations/2.2_Ma_Changjie.pdf, 2018.

[85] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote Attacks on Automated Vehicles Sensors: Experiments on Camera and Lidar," *Black Hat Europe*, vol. 11, p. 2015, 2015.
[86] C. Yan, W. Xu, and J. Liu, "Can You Trust Autonomous Cehicles: Contactless Attacks Against Sensors of Self-Driving Vehicle," *DEF CON*, vol. 24, no. 8, p. 109, 2016.
[87] B. Nassi, D. Nassi, R. Ben-Netanel, Y. Mirsky, O. Drokin, and Y. Elovici, "Phantom of the ADAS: Phantom Attacks on Driver-Assistance Systems," in *IACR Cryptol. ePrint Arch.*, 2020.
[88] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao, "Adversarial Sensor Attack on Lidar-Based Perception in Autonomous Driving," in *ACM SIGSAC Conference on Computer and Communications Security (ACM CCS)*, 2019.
[89] Y. Tu, Z. Lin, I. Lee, and X. Hei, "Injected and Delivered: Fabricating Implicit Control over Actuation Systems by Spoofing Inertial Sensors," in *USENIX Security Symposium*, pp. 1545–1562, 2018.
[90] Y. Jia, Y. Lu, J. Shen, Q. A. Chen, H. Chen, Z. Zhong, and T. Wei, "Fooling Detection Alone is Not Enough: Adversarial Attack Against Multiple Object Tracking," in *International Conference on Learning Representations (ICLR)*, 2019.
[91] J. Shen, J. Y. Won, Z. Chen, and Q. A. Chen, "Drift with Devil: Security of Multi-Sensor Fusion based Localization in High-Level Autonomous Driving under GPS Spoofing," in *USENIX Security Symposium*, 2020.
[92] K. Tang, J. Shen, and Q. A. Chen, "Fooling Perception via Location: A Case of Region-of-Interest Attacks on Traffic Light Detection in Autonomous Driving," in *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, 2021.
[93] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition," in *ACM SIGSAC Conference on Computer and Communications Security (ACM CCS)*, pp. 1528–1540, 2016.
[94] S.-T. Chen, C. Cornelius, J. Martin, and D. H. P. Chau, "Shapeshifter: Robust Physical Adversarial Attack on Faster R-CNN Object Detector," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 52–68, Springer, 2018.
[95] Z. Zhong, W. Xu, Y. Jia, and T. Wei, "Perception Deception: Physical Adversarial Attack Challenges and Tactics for DNN-Based Object Detection," in *Black Hat Europe*, 2018.
[96] N. S. Council, *Reference Material for DDC Instructors, 5th Edition*. 2005.
[97] UK ACPO Road Policing Enforcement Technology Committee, *ACPO Code of Practice for Operational Use of Enforcement Equipment*. 2002.
[98] U. D. for Transport, *The Official Highway Code Book*. 2015.
[99] H. Loeb, A. Belwadi, J. Maheshwari, and S. Shaikh, "Age and Gender Differences in Emergency Takeover from Automated to Manual Driving on Simulator," *Traffic injury prevention*, pp. 1–3, 2019.
[100] "Watch Tesla Drivers Apparently Asleep at the Wheel, Renewing Autopilot Safety Questions." https://www.cnbc.com/2019/09/09/watch-tesla-drivers-apparently-asleep-at-the-wheel-renewing-safety-questions.html, 2019.
[101] "Amazon Mechanical Turk." https://www.mturk.com/.
[102] "Driver Take-Over Decision Survey with Automated Lane Centering System in our User Study." https://storage.googleapis.com/driving-decision-survey/driving_decision_survey.pdf, 2020.

A Required Deviations and Success Time

Required deviations. The required deviations for the highway and local roads are calculated based on Toyota RAV4 width (including mirrors) and standard lane widths in the U.S. [62] as shown in Fig. 16. We use Toyota RAV4 since it is the reference vehicle used by the OpenPilot team when collecting the comma2k19 data set [56]. For the lane widths, we refer to the design guidelines [62] published by the U.S. Department of Transportation Federal Highway Administration. The required deviations to touch the lane line are calculated using $\frac{L-C}{2} = 0.735m$ (highway) and $0.285m$ (local), where L is the lane width and C is the vehicle width.

Required success time. Since ALC systems assume a fully attentive human driver who is prepared to take over at any moment [1, 7], the required deviation above needs to be achieved fast enough so that the human driver cannot react in time to take over and steer back. Thus, when we define the attack goal, we require not only the required deviation above, but also an attack success time that is smaller than the average driver reaction time to road hazards. We select the average driver reaction time based on different government-issued transportation policy guidelines [57, 96]. In particular, in the California Department of Motor Vehicles Commercial Driver Handbook Section 2.6.1 [57], it describes (1) a 1.75 seconds *average perception time*, i.e., the time from the time the driver's eyes see a hazard until the driver's brain recognizes it, and (2) a 0.75 to 1 seconds *average reaction time*, i.e., the time from the driver's brain recognizing the hazard to physically take actions. Thus, in total it's **2.5 to 2.75 seconds** from the driver's eyes seeing a hazard to physically take actions. The UK "Highway Code Book" and "Code of Practice for Operational Use of Road Policing Enforcement Technology" use 3 seconds for driver reaction time [97, 98]. National Safety Council also adopts a 3-second driver reaction time to calculate the minimum spacing between vehicles [96]. Among them, we select the **smallest** one, i.e., 2.5 seconds from the California Department of Motor Vehicles [57], as the required success time in this paper to avoid possible overestimation of the attack effectiveness in our evaluation.

Note that the driver reaction time above is commonly referring to the reaction time to apply the brake, instead of steering. In our paper, we use such reaction time to apply the brake as the reaction time to take over the steering wheel when the ALC systems are in control of the steering wheel. This is because in traditional driving, the driver is *actively* steering the vehicle but *passively* applying the brake. However, when the ALC system is controlling the steering, the human driver is *passively* steering the vehicle, i.e., her hands are not actively controlling the steering wheel. Thus, the reaction time to take over the steering wheel during passive steering is analogous to that to apply the brake during passive braking.

In fact, the actual average driver reaction time when the ALC system is taking control is likely to be much higher than the 2.5 seconds measured in traditional driving, due to the reliance of human drivers on such convenient driving automation technology today. A recent study performed a simulation-based user study on Tesla Autopilot, and found that 40% drivers fail to react in time to avoid a crash happening *6.2 seconds* after the Autopilot fails to operate [99]. In the real world, it is found multiple times that Tesla drivers fall asleep with Autopilot controlling the vehicle in high speed [100]. Thus, the required success time of 2.5 seconds used in this paper is a relatively conservative estimation, and thus the

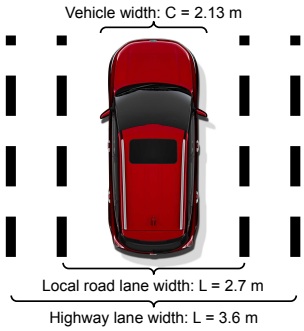

Figure 16: Vehicle and lane widths used in this paper.

attack effectiveness reported in our evaluations is likely only a **lower bound** of the actual effectiveness of our attack in the real world.

B Attack Stealthiness User Study

In this section, we conduct a user study to more directly evaluate the stealthiness of the DRP attack. We have gone through the IRB process and our study is determined as in the IRB Exempt category since it does not involve the collection of any Personally Identifiable Information (PII) or target any sensitive population.

Evaluation methodology. We use the generated attacks on real-world driving traces in §5.1 to perform the user study. For an attack scenario, we ask the participants to imagine that they are driving with the ALC system taking control, and then show a sequence of image frames with the malicious road patch from the driver's view at 3, 2.5, 2, 1.5, and 1 second(s) before the attack succeeds. Here, 1 second before the attack succeeds is right before the attack starts to take effect. For each image frame, we ask whether they will decide to take over the driving to avoid danger or potential safety risks. These questions are also asked for the image frames with a benign road patch that only has the base color without the malicious dirty patterns as a control group.

Since our attack is designed for drivers who are in favor of using ALC system in normal cases, the same set of questions are asked at the beginning for the original image frames without attack, and we only accept a participant if she does not choose to take over the driving for these cases. This process also helps filter out ill-behaved participants who just provide random answers. Since DRP is a new form of attack vectors on the road, we do not tell the participants that the study is related to security attacks. Instead, we only tell them that our focus is on surveying driver's decisions under ALC systems for different road surface patterns such as road patches and scratches. At the beginning of the study, we also provide an introduction of ALC systems with demo videos to ensure that the participants fully understand what driving technology we are surveying about. To understand the distribution of the participant background, we also ask demographic information and background information related to driving and ALC usage. None of the questions in our study involve PII or target any sensitive population; our study is thus determined as in the IRB Exempt category.

Evaluation setup. We use Amazon Mechanical Turk [101] to perform this study, and in total collected 100 participants. All of them have driving experience, which is confirmed by asking them the age when first licensed and the weekly driving mileage. A local-road driving trace is used in this study, and for the scenarios with attack, we evaluate 3 stealthiness levels as in §5.1 (i.e., $\lambda = 10^{-2}, 10^{-3}, 10^{-4}$). The survey is available at [102]. Among the 100 participants, 56% are male and 44% are female. The average age is 32.3 years old. 79% have experienced at least one ALC system, among which Tesla Autopilot has the largest share (28%). Statistics of ALC experiment and demographic information are shown in Fig. 18.

Results. Fig. 17 shows the study results. As shown, the closer it is to the attack success time, the more participants choose to take over the driving in the attacked scenarios since the dirty patterns become increasingly larger and clearer. Among the 3 stealthiness levels, the driver decisions are consistent with our design: the lowest stealthiness level ($\lambda = 10^{-4}$) has the highest take-over rate, while the highest level ($\lambda = 10^{-2}$) has the lowest. In particular, we find that even for the lowest stealthiness level ($\lambda = 10^{-4}$), only *less than 25%* of the participants decide to take over before the attack starts to take effect. As shown in Fig. 7, at this stealthiness level the white dirty patterns are quite dense and prominent. Thus, these results suggest that the majority of human drivers today do not treat dirty road patches as road conditions where ALC systems cannot handle.

As introduced in §3.1, 2.5 seconds is commonly used as the average driver reaction time to road hazards. Thus, at 2.5 seconds or more before the attack succeeds, the human driver still has a chance to take over the driving to prevent the damage in common cases, as long as she can realize that it is a road hazard. However, our results show that only less than 20% of the participants decide to take over at 2.5 and 3 seconds before our attack succeeds even for the lowest stealthiness level. In particular, when the stealthiness levels are $\lambda = 10^{-2}$ and $\lambda = 10^{-3}$, the take-over rates at these 2 time points are similar to the rates for the benign road patch with only the base color. This suggests that at the time when there is still a chance to prevent the damage in common cases, our attack patches at $\lambda = 10^{-2}$ and 10^{-3} *appear to be as innocent as normal clean road patches to human drivers*. In these cases, the take-over rates are only *less than 15%*, which are from participants who will take over even for normal clean road patches. Note that the take-over rates in practice are likely to be lower than this since (1) this study is performed for a local road scenario, while the road patches in highway scenarios are much farther and thus much less noticeable as shown in Fig. 7, and (2) the road patches in this study are digitally synthesized into the image frames, which may appear less natural and thus may more easily alert the participants.

Stealthiness from pedestrian view. In local road scenar-

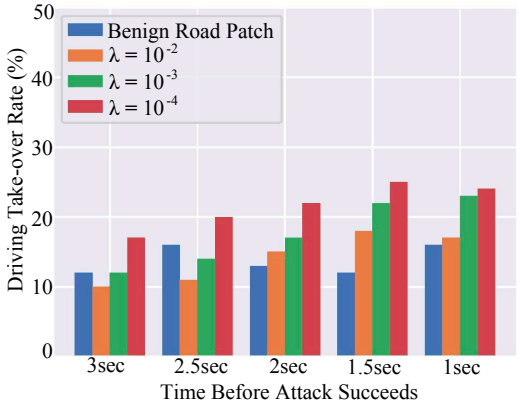

Figure 17: Results of the attack stealthiness user study. Driving take-over rate is the percentage of participants who choose to take over the driving at a particular time point before the attack succeeds.

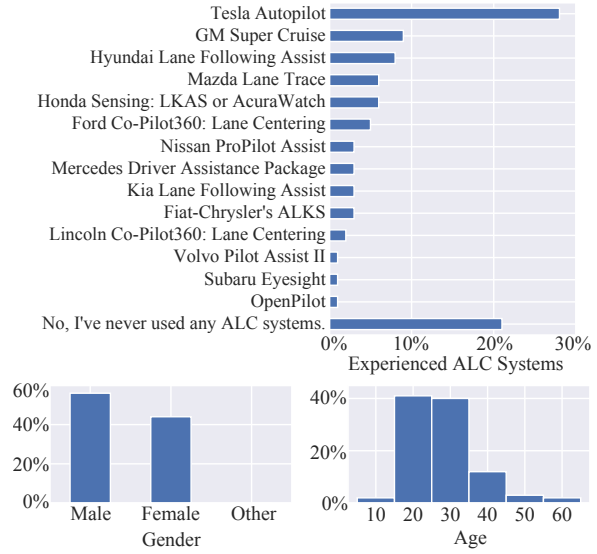

Figure 18: Statistics of the ALC system experience and demographic information in the attack stealthiness user study.

ios, the stealthiness from the pedestrian's view is also an aspect worth considering, as pedestrians may report anomalies if our attack patch looks too suspicious. Our user study includes the driver's view at 1 second before the attack succeeds, which is 7 meters to the driver's eyes so similar to the distance from the pedestrian on local roads. However, only <25% of the participants choose to take over driving, meaning that >75% do not think our attack patch at this distance looks suspicious enough to affect driving. This may be because the general public today does not know that dirty road patches can be a road hazard. We hope that our paper can expose this and thus help raise such awareness.

C Details of OpenPilot ALC system

In this section, we describe the implementation details of the OpenPilot ALC system, which follows the typical modular ALC system design introduced in §1:

Lane Detection (LD). The LD model used in OpenPilot uses recurrent DNN structures (e.g., RNN and GRU), which are more detailed in our extended version [38] for 3 specific versions of it. In each frame, the recurrent model receives a front-camera input of 512 pixels wide by 256 pixels high and 512-dimensional recurrent features from the previous frame. The recurrent features are the output of a middle layer. The final output for ALC consists of information of 3 lines (left and right lines and driving path). Each line has coordinates of 192 points (1 m interval from the vehicle to driving direction), uncertainty scores of each coordinate, and a confidence score of its lane. Thus, there are $(192 \times 2 + 1) \times 3 = 1,155$ output values in total. The desired driving path is calculated by the weighted average of the driving path and the center line of the left and right lines weighted by the uncertainty and confidence scores. See OpenPilot code [8] for more details.

Such recurrent structure is stateful: it allows leveraging the previous detection results to enhance the current frame detection since lane line shapes are typically not changed largely across consecutive frame. OpenPilot LD models output the detected lane line points of the left line, right line, and predicted driving path. Each line is fitted to the 3-degree polynomial, and the desired driving path is then calculated as the weighted average of the three lines with their confidence levels. OpenPilot LD operates at 20 Hz (every 50 ms). In §7, we inject the attack traces at the end of this step by modifying the ALC source code to replace the real-time LD model outputs with a sequence of attacked ones obtained from the software-in-the-loop simulation at the same driving speed (simulation environment described in §6).

Lateral control. OpenPilot adopts Model Predictive Control (MPC) [24] to decide the desired steering angle, which will then be sent to the vehicle actuation step. The input of the MPC is the desired driving path, the current speed, and the current steering angle. This step works at the same frequency as LD, i.e., the desired steering angle is decided every 50 ms. The MPC is stateful: it reuses the solution of the previous frame as the initial solution for the current frame.

Vehicle actuation. Based on the obtained desired steering angle from MPC, OpenPilot vehicle actuation decides the steering angle *change* to actuate in the control step and sends actuation messages through CAN (Controller Area Network) bus. This thus makes the absolute value of the actuated steering angle stateful: the new actuated steering angle is built upon the previous one, by applying the angle *change* actuations. OpenPilot actuation works at 100 Hz control frequency. The actuated steering angle change is *up to 0.25° per control step* (every 10 ms). As described in §2.1, such limit is typically imposed in production ALC systems due to the physical constraints of the mechanical control units and also for driving stability and safety [22]. OpenPilot is integrated to a vehicle by overriding the stock cruise control system. It thus is engaged to control the steering and throttle when the driver turns on the cruise control mode, and can work with stock safety features such as AEB and FCW [8].