Sequential Attacks on Kalman Filter-based Forward Collision Warning Systems

Yuzhe Ma, Jon Sharp, Ruizhe Wang, Earlence Fernandes, Xiaojin Zhu
Department of Computer Sciences, University of Wisconsin–Madison
{yzm234, sharp-jr, ruizhe, earlence, jerryzhu}@cs.wisc.edu

Abstract

Kalman Filter (KF) is widely used in various domains to perform sequential learning or variable estimation. In the context of autonomous vehicles, KF constitutes the core component of many Advanced Driver Assistance Systems (ADAS), such as Forward Collision Warning (FCW). It tracks the states (distance, velocity etc.) of relevant traffic objects based on sensor measurements. The tracking output of KF is often fed into downstream logic to produce alerts, which will then be used by human drivers to make driving decisions in near-collision scenarios. In this paper, we study adversarial attacks on KF as part of the more complex machine-human hybrid system of Forward Collision Warning. Our attack goal is to negatively affect human braking decisions by causing KF to output incorrect state estimations that lead to false or delayed alerts. We accomplish this by sequentially manipulating measurements fed into the KF, and propose a novel Model Predictive Control (MPC) approach to compute the optimal manipulation. Via experiments conducted in a simulated driving environment, we show that the attacker is able to successfully change FCW alert signals through planned manipulation over measurements prior to the desired target time. These results demonstrate that our attack can stealthily mislead a distracted human driver and cause vehicle collisions.

1 Introduction

Advanced Driver Assistance Systems (ADAS) are hybrid human-machine systems that are widely deployed on production passenger vehicles (National Highway Traffic Safety Administration 2020). They use sensing, traditional signal processing and machine learning to detect and raise alerts about unsafe road situations and rely on the human driver to take corrective actions. Popular ADAS examples include Forward Collision Warning (FCW), Adaptive Cruise Control and Autonomous Emergency Braking (AEB).

Although ADAS hybrid systems are designed to increase road safety when drivers are distracted, attackers can negate their benefits by strategically tampering with their behavior. For example, an attacker could convince an FCW or AEB system that there is no imminent collision until it is too late for a human driver to avoid the crash.

We study the robustness of ADAS to attacks. The core of ADAS typically involves tracking the states (e.g., distance and velocity) of road objects using Kalman filter (KF). Downstream logic uses this tracking output to detect unsafe situations before they happen. We focus our efforts on Forward Collision Warning (FCW), a popular ADAS deployed on production vehicles today. FCW uses KF state predictions to detect whether the ego vehicle (vehicle employing the ADAS system) is about to collide with the most important object in front of it and will alert the human driver in a timely manner. Thus, our concrete attack goal is to trick the KF that FCW uses and make it output incorrect state predictions that would induce false or delayed alerts depending on the specific physical situation.

Recent work has examined the robustness of road object state tracking for autonomous vehicles (Jia et al. 2020). Their attacks create an instantaneous manipulation to the Kalman filter inputs without considering its sequential nature, the downstream logic that depends on filter output, or the physical dynamics of involved vehicles. This leads to temporarily hijacked Kalman filter state predictions that are incapable of ensuring that downstream logic is reliably tricked into producing false alerts. By contrast, we adopt an online planning view of attacking KFs that accounts for: (1) their sequential nature where current predictions depend on past measurements; and (2) the downstream logic that uses KF output to produce warnings. Our attack technique also considers a simplified model of human reaction to manipulated FCW warning lights.

We propose a novel Model Predictive Control (MPC)-based attack that can sequentially manipulate measurement inputs to a KF with the goal of stealthily hijacking its behavior. Our attacks force FCW alerts that mask the true nature of the physical situation involving the vehicles until it is too late for a distracted human driver to take corrective actions.

We evaluate our attack framework by creating a high-fidelity driving simulation using CARLA (Dosovitskiy et al. 2017), a popular tool for autonomous vehicle research and development. We create test scenarios based on real-world driving data (National Highway Traffic Safety Administration 2011; European New Car Assessment Programme 2018) and demonstrate the practicality of the attack in causing crashes involving the victim vehicle. Anonymized CARLA simulation videos of our attacks are available at https://sites.google.com/view/attack-kalman-filter.

Main Contributions:

- We develop an optimal control-based attack against the popular FCW driver assistance system. Our attack targets several critical parts of the FCW pipeline – Kalman filter tracking and prediction, FCW alert logic and human decision making in crash and near-crash scenarios.
- We evaluate our control-based attacks in a high-fidelity simulation environment demonstrating that an attacker can compromise *only* the camera-based measurement data and accomplish their goals of creating end-to-end unsafe situations for an FCW system, even under the constraint of limited manipulation to measurements.
- We show that attack planning in advance of the targeted point is beneficial to the attack compared to without planning. Given 25 steps of planning (or 1.25 seconds based on specific physical situations in our evaluation) before the targeted time point, the attacker can cause the desired effect, while the attack fails without planning. Furthermore, via comparisons against a baseline greedy attack, we show that our attack can find near-optimal planning that achieves better overall performance.

2 Background

Forward Collision Warning provides audio-visual alerts to warn human drivers of imminent collisions. Fig. 1 shows the pipeline of a prototypical FCW hybrid system (MATLAB 2020b): (1) It uses camera and RADAR sensors to perceive the environment; (2) It processes sensor data using a combination of traditional signal processing and machine learning algorithms to derive object velocities and distances; (3) A Kalman filter tracks the Most Important Object (MIO) state and makes predictions about its future states; (4) FCW logic uses Kalman filter predictions to determine whether a collision is about to occur and creates audio-visual warnings; (5) A human driver reacts to FCW alerts. These alerts can be either: green – indicating no danger, yellow – indicating potential danger of forward collision, and red – indicating imminent danger where braking action must be taken.

We focus on attacking the core steps of FCW (shaded parts of Fig. 1). Thus, we assume there is a single MIO in front of the ego vehicle and a single Kalman filter actively tracking its state. The steps of measurement assignment and MIO identification will not be considered in this paper.

We have two attack goals that will comprehensively demonstrate the vulnerability of FCW hybrid systems — the attacker should trick FCW into showing no red alerts when there is an imminent collision with the most important object (MIO), and vice versa — the attacker should trick FCW into showing red alerts when there is no collision, inducing a human to react with braking that can potentially lead to a rear-end crash with a trailing vehicle.

2.1 Kalman Filtering

At the core of FCW is the Kalman Filter, which estimates the state of the MIO based on sensor measurements. In this paper, the state of the MIO is represented as $x_t = (d_t^1, v_t^1, a_t^1, d_t^2, v_t^2, a_t^2)$, where d_t^1, v_t^1, a_t^1 are the distance, velocity and acceleration of the MIO along the driving direction, and d_t^2, v_t^2, a_t^2 for the lateral direction (perpendicular to driving direction). Then KF models the evolution of x_t as

$$x_{t+1} = Ax_t + \omega_t, t \geq 1, \tag{1}$$

where A is the state-transition matrix and $\omega_t \sim N(0, \Omega)$ is Gaussian noise. The underlying state x_t is unknown, but one can obtain measurements y_t of the state as

$$y_t = Cx_t + \psi_t, t \geq 1, \tag{2}$$

where C is the measurement matrix and $\psi_t \sim N(0, \Psi)$ is the measurement noise. In our paper, $y_t \in \mathbb{R}^8$ contains vision and radar measurements of the MIO distance and velocity along two directions, i.e., $y_t = (d_t^{1,\nu}, v_t^{1,\nu}, d_t^{2,\nu}, v_t^{2,\nu}, d_t^{1,r}, v_t^{1,r}, d_t^{2,r}, v_t^{2,r})$, where we use superscripts ν, r for vision and radar, and numbers 1, 2 for driving and lateral direction, respectively. Given the state dynamics (1) and measurement model (2), KF provides a recursive formula to estimate the state based on sequential measurements obtained over time. Concretely, KF starts from some initial state and covariance prediction $\hat{x}_1$ and $\hat{\Sigma}_1$. Then for any $t \geq 2$, KF first applies (3) to correct the predictions based on measurements y_t. The corrected state and covariance matrix are denoted by $\bar{x}_t$ and $\bar{\Sigma}_t$.

$$\begin{aligned} \bar{x}_t &= (I - H_{t-1}C)\hat{x}_{t-1} + H_{t-1}y_t, \\ \bar{\Sigma}_t &= (I - H_{t-1}C)\hat{\Sigma}_{t-1}. \end{aligned} \tag{3}$$

where $H_{t-1} = \hat{\Sigma}_{t-1}C^\top(C\hat{\Sigma}_{t-1}C^\top + \Psi)^{-1}$. Next, KF applies (4) to predict state and covariance for the next step.

$$\hat{x}_t = A\bar{x}_t, \quad \hat{\Sigma}_t = A\bar{\Sigma}_t A^\top + \Omega. \tag{4}$$

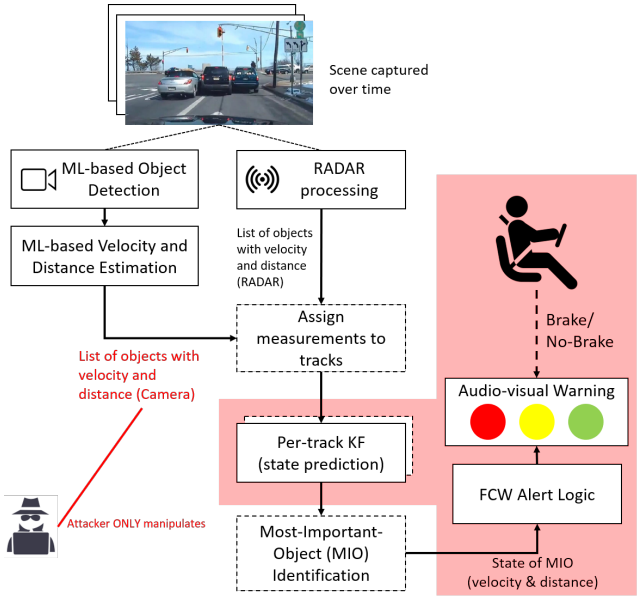


Figure 1: Overview of Forward Collision Warning (FCW) hybrid human-machine system. We take a first step to understanding the robustness of this system to attackers who can compromise sensor measurements. Therefore, we filter the problem to its essence (shaded parts) — the Kalman filter that tracks the most important object (MIO) and the downstream logic that decides how to warn the driver.

The correction and prediction steps are applied recursively as t grows. Note that the derivation of covariance matrix is independent of y_t, thus can be computed beforehand.

2.2 Warning Alert Logic and Human Model

In this paper, we follow the FCW alert logic used in (MATLAB 2020b). Let the state prediction be $\hat{x}_t = (\hat{d}_t^1, \hat{v}_t^1, \hat{a}_t^1, \hat{d}_t^2, \hat{v}_t^2, \hat{a}_t^2)$, then the warning light ℓ_t output by FCW at step t is one of the following three cases:

- Safe (Green): The MIO is moving away, or the distance to MIO remains constant, i.e., $\hat{v}_t^1 \geq 0$.
- Caution (Yellow): The MIO is moving closer, but still at a distance further than the minimum safe distance $d^*(\hat{v}_t^1)$, i.e., $\hat{v}_t^1 < 0$ and $\hat{d}_t^1 > d^*(\hat{v}_t^1)$. We define the safe distance as $d^*(\hat{v}_t^1) = -1.2\hat{v}_t^1 + (\hat{v}_t^1)^2/0.8g$, where g is 9.8 m/s^2.
- Warn (Red): The MIO is moving closer, and at a distance less than the minimum safe distance, i.e., $\hat{v}_t^1 < 0$ and $\hat{d}_t^1 \leq d^*(\hat{v}_t^1)$.

The FCW alert logic can be summarized as:

$$F(\hat{x}_t) = \begin{cases} \text{green} & \text{if } \hat{v}_t^1 \geq 0, \\ \text{yellow} & \text{if } \hat{v}_t^1 < 0, \hat{d}_t^1 > d^*(\hat{v}_t^1), \\ \text{red} & \text{if } \hat{v}_t^1 < 0, \hat{d}_t^1 \leq d^*(\hat{v}_t^1). \end{cases} \tag{5}$$

Given the FCW warning light, the human driver could be in one of the following two states – applying the brake pedal, or not applying/releasing the brake. We take into account human reaction time h^*; warning lights must sustain at least h^* steps before the human driver switches state. That is, the driver brakes after h^* steps since the first red light, and releases the brake after h^* steps since the first yellow/green light. Note that the yellow and green lights are treated identically in both cases because the MIO is outside the safe distance and no brake is needed. In appendix E, we provide an algorithmic description of the human model.

3 Attack Problem Formulation

We assume white-box setting where the attacker can access the KF parameters (e.g., through reverse engineering). The attacker can directly manipulate measurements (i.e., false data injection), but only pertaining to the vision component, and not the RADAR data. Our attack framework is agnostic of whether the attacker manipulates camera or RADAR, but we choose to only manipulate camera because of the increasing presence of deep learning techniques in ADAS and their general vulnerability to adversarial examples (Szegedy et al. 2013; Eykholt et al. 2018b; Athalye et al. 2017; Sharif et al. 2016). We envision that future work can integrate our results into adversarial examples to create physical attacks.

We further restrict the attacker to only making physically plausible changes to the vision measurements. This is because an anomaly detection system might filter out physically implausible measurements (e.g., change of 10^4m/s over one second). Concretely, we require that the distance and velocity measurement after attack must lie in $[\underline{d}, \bar{d}]$ and $[\underline{v}, \bar{v}]$ respectively. We let $[\underline{d}, \bar{d}] = [0, 75]$ and $[\underline{v}, \bar{v}] = [-30, 30]$. Finally, we assume that at any time step, the attacker knows the true measurement only for that time step, but does not know future measurements. To address this difficulty of an unknown future, we propose a model predictive control (MPC)-based attack framework that consists of an outer problem and an inner problem, where the inner problem is an instantiation of the outer problem with respect to attacker-envisioned future in every step of MPC. In the following, we first introduce the outer problem formulation.

3.1 Outer Attack Problem

Our attacker has a pre-specified target interval $\mathcal{T}^\dagger$, and aims at changing the warning lights output by FCW in $\mathcal{T}^\dagger$. As a result, the human driver sees different lights and takes unsafe actions. Specifically, for any time $t \in \mathcal{T}^\dagger$, the attacker hopes to cause the FCW to output a desired target light $\ell_t^\dagger$, as characterized by (12), in which $F(\cdot)$ is the FCW alert logic (5). To accomplish this, the attacker manipulates measurements in an attack interval $\mathcal{T}^a$. In our paper, we assume $\mathcal{T}^\dagger \subset \mathcal{T}^a$. Furthermore, we consider only the scenario where $\mathcal{T}^\dagger$ and $\mathcal{T}^a$ have the same last step, since attacking after the target interval is not needed. Let δ_t be the manipulation at step t, and $\tilde{y}_t = y_t + \delta_t$ be measurement after attack. We refer to the i-th component of δ_t as δ_t^i. We next define the attack effort as the cumulative change over measurements $J = \sum_{t \in \mathcal{T}^a} \delta_t^\top R \delta_t$. where $R \succ 0$ is the effort matrix. The attacker hopes to minimize the attack effort.

Meanwhile, the attacker cannot arbitrarily manipulate measurements. We consider two constraints on the manipulation. First, MIO distance and velocity are limited by simple natural physics, as shown in (11). Moreover, similar to the norm ball used in adversarial examples, we impose another constraint that restricts the attacker's manipulation $\|\delta_t\|_\infty \leq \Delta$ (see (10)). We refer to $\mathcal{T}^s = \mathcal{T}^a \backslash \mathcal{T}^\dagger$, the difference between $\mathcal{T}^a$ and $\mathcal{T}^\dagger$, as the stealthy (or planning) interval. During $\mathcal{T}^s$, the attacker can induce manipulations before the target interval with advance planning, and by doing so, hopefully better achieve the desired effect in the target interval. However, for the sake of stealthiness, the planned manipulation should not change the original lights ℓ_t during $\mathcal{T}^s$. This is characterized by the stealthiness constraint (13).

Given all above, the attack can be formulated as:

$$\min_{\delta_t} \quad J = \sum_{t \in \mathcal{T}^a} \delta_t^\top R \delta_t, \tag{6}$$

$$\text{s.t.} \quad \tilde{y}_t = y_t + \delta_t, \forall t \in \mathcal{T}^a, \tag{7}$$

$$\tilde{x}_t = A(I - H_{t-1}C)\tilde{x}_{t-1} + AH_{t-1}\tilde{y}_t, \tag{8}$$

$$\delta_t^i = 0, \forall i \in \mathcal{I}_{\text{radar}}, \forall t \in \mathcal{T}^a, \tag{9}$$

$$\|\delta_t\| \leq \Delta, \forall t \in \mathcal{T}^a, \tag{10}$$

$$\tilde{d}_t^{1,\nu} \in [\underline{d}, \bar{d}], \tilde{v}_t^{1,\nu} \in [\underline{v}, \bar{v}], \forall t \in \mathcal{T}^a, \tag{11}$$

$$F(\tilde{x}_t) = \ell_t^\dagger, \forall t \in \mathcal{T}^\dagger, \tag{12}$$

$$F(\tilde{x}_t) = \ell_t, \forall t \in \mathcal{T}^s. \tag{13}$$

The constraint (8) specifies the evolution of the state prediction under the attacked measurements $\tilde{y}_t$. (9) enforces no change on radar measurements, where $\mathcal{I}_{\text{radar}} = \{5, 6, 7, 8\}$ contains indexes of all radar components . The attack optimization is hard to solve due to three reasons:

(1). The problem could be non-convex.

(2). The problem could be be infeasible.

(3). The optimization is defined on measurements y_t that are not visible until after $\mathcal{T}^a$, while the attacker must design manipulations δ_t during $\mathcal{T}^a$ in an online manner.

We now explain how to address the above three issues.

The only potential sources of non-convexity in our attack are (12) and (13). We now explain how to derive a surrogate convex problem using $\ell_t^\dagger = \ell_t^o =$ red as an example. The other scenarios are similar, thus we leave the details to Appendix B. The constraint $F(\tilde{x}_t) =$ red is equivalent to

$$\tilde{v}_t^{1,\nu} < 0, \tag{14}$$

$$\tilde{d}_t^{1,\nu} \leq -1.2\tilde{v}_t^{1,\nu} + \frac{1}{0.8g}(\tilde{v}_t^{1,\nu})^2., \tag{15}$$

The above constraints result in non-convex optimzation mainly because (15) is nonlinear. To formulate a convex problem, we now introduce surrogate constraints that are tighter than (14), (15) but guarantee convexity.

Proposition 1 *Let* $U(d) = 0.48g - \sqrt{(0.48g)^2 + 0.8gd}$. *Let* $\epsilon > 0$ *be any positive number. Then for any* $d_0 \geq 0$, *the surrogate constraints* (16), (17) *are tighter than* $F(\tilde{x}_t) =$ *red, and induce convex attack optimization.*

$$\tilde{v}_t^{1,\nu} \leq -\epsilon, \tag{16}$$

$$\tilde{v}_t^{1,\nu} \leq U'(d_0)(\tilde{d}_t^{1,\nu} - d_0) + U(d_0) - \epsilon. \tag{17}$$

We provide a proof and guidance on how to select d_0 in Appendix B. With the surrogate constraints, the attack optimization becomes convex. However, the surrogate optimization might still be infeasible. To address the feasibility issue, we further introduce slack variables into (16), (17) to allow violation of stealthiness and target lights:

$$\tilde{v}_t^{1,\nu} \leq -\epsilon + \xi_t, \tag{18}$$

$$\tilde{v}_t^{1,\nu} \leq U'(d_0)(\tilde{d}_t^{1,\nu} - d_0) + U(d_0) - \epsilon + \zeta_t. \tag{19}$$

We include these slack variables in the objective function:

$$J = \underbrace{\sum_{t\in\mathcal{T}^a} \delta_t^\top R\delta_t}_{\text{total manipulation } J_1} + \lambda \underbrace{\sum_{t\in\mathcal{T}^s} (\xi_t^2 + \zeta_t^2)}_{\text{stealthiness violation } J_2} + \lambda \underbrace{\sum_{t\in\mathcal{T}^\dagger} (\xi_t^2 + \zeta_t^2)}_{\text{target violation } J_3}. \tag{20}$$

Then, the surrogate attack optimization is

$$\min_{\delta_t} \quad J = J_1 + \lambda J_2 + \lambda J_3, \tag{21}$$

$$\text{s.t.} \quad (7)\text{-}(11), (18), (19). \tag{22}$$

Proposition 2 *The attack optimization* (21)-(22) *with surrogate constraints and slack variables is convex and feasible.*

3.2 Inner Attack Problem: MPC-based Attack

In the outer surrogate attack (21)-(22), we need to assume the attacker knows the measurements y_t in the entire attack interval $\mathcal{T}^a$ beforehand. However, the attacker cannot know the future. Instead, he can only observe and manipulate the current measurement in an online manner. To address the unknown future issue, we adopt a control perspective and view the attacker as an adversarial controller of the KF, where the control action is the manipulation δ_t. We then apply MPC, an iterative control method that progressively solves (21)-(22). By using MPC, the attacker is able to adapt the manipulation to the instantiated measurements revealed over time while accounting for unknown future measurements.

Input : target interval $\mathcal{T}^\dagger$, target lights $\ell_t^\dagger, t \in \mathcal{T}^\dagger$, stealthy interval $\mathcal{T}^s$, original lights $\ell_t, t \in \mathcal{T}^s$.
Initialize $\hat{x}_1$ and $\hat{\Sigma}_1$. Let $\tilde{x}_1 = \hat{x}_1$, $\mathcal{T}^a = \mathcal{T}^s \cup \mathcal{T}^\dagger$;
for $t \leftarrow 2$ **to** T **do**
  environment generates measurement y_t;
  if $t \in \mathcal{T}^a$ **then**
    attacker infers clean state $\hat{x}_t$ without attack;
    attacker predicts future $\hat{y}_t$ with (23);
    attacker solves (24)-(26) to obtain δ_τ $(\tau \geq t)$;
    attacker manipulates y_t to $\tilde{y}_t = y_t + \delta_t$;
    $\tilde{x}_t$ evolves to $\tilde{x}_{t+1}$ according to $\tilde{y}_t$
  else $\tilde{x}_t$ evolves to $\tilde{x}_{t+1}$ according y_t ;
end

Algorithm 1: MPC-based attack.

Specifically, in each step t, the attacker has observed all past measurements $y_1, ...y_{t-1}$ and the current measurement y_t. Thus, the attacker can infer the clean state $\hat{x}_t$ in the case of no attacker intervention. Based on $\hat{x}_t$, the attacker can recursively predict future measurements by simulating the environmental dynamics without noise, i.e., $\forall \tau > t$:

$$x'_\tau = Ax'_{\tau-1}, \hat{y}_\tau = Cx'_\tau. \tag{23}$$

The recursion starts from $x'_t = \hat{x}_t$. The attacker then replaces the unknown measurements in the outer attack by its prediction $\hat{y}_\tau$ $(\tau > t)$ to derive the following inner attack:

$$\min_{\delta_{\tau:\tau\geq t}} \quad J = \sum_{\tau\in\mathcal{T}^a} \delta_\tau^\top R\delta_\tau + \lambda \sum_{\tau\in\mathcal{T}^a} (\xi_\tau^2 + \zeta_\tau^2), \tag{24}$$

$$\text{s.t.} \quad \tilde{y}_\tau = \hat{y}_\tau + \delta_\tau, \forall \tau \geq t, \tag{25}$$

$$(8)\text{-}(11), (18), (19) \text{ (defined on } \tau \geq t). \tag{26}$$

The attacker solves the above inner attack in every step t. Assume the solution is $\delta_\tau(\tau \geq t)$. Then, the attacker only implements the manipulation on the current measurement, i.e., $\tilde{y}_t = y_t + \delta_t$, and discards the future manipulations. After that, the attacker enters step $t+1$ and applies MPC again to manipulate the next measurement. This procedure continues until the last step of the attack interval $\mathcal{T}^a$. We briefly illustrate the MPC-based attack in algorithm 1.

4 Experiments on CARLA Simulation

In this section, we empirically study the performance of the MPC-based attack. We first describe the simulation setup.

4.1 Simulation Setup

We use CARLA (Dosovitskiy et al. 2017), a high-fidelity vehicle simulation environment, to generate measurement data that we input to the Kalman filter-based FCW. CARLA supports configurable sensors and test tracks. We configure the

simulated vehicle to contain a single forward-facing RGB camera (800x600 pixels), a forward-facing depth camera of the same resolution, and a single forward-facing RADAR (15° vertical detection range, 6000 points/sec, 85 m maximum detection distance). We took this configuration from a publicly-available FCW implementation (MATLAB 2020b). The simulation runs at 20 frames/sec and thus, each sensor receives data at that rate. Furthermore, this configuration is commonly available on production vehicles today (Joseph A. Gregor 2017), and thus, our simulation setup matches real-world FCW systems from a hardware perspective.

For each time step of the simulation, CARLA outputs a single RGB image, a depth map image, and variable number of RADAR points. We use YOLOv2 (Redmon et al. 2016) to produce vehicle bounding boxes, the Hungarian pairwise matching algorithm (Kuhn 1955) to match boxes between frames, and the first derivative of paired depth map image readings to produce vehicle detections from vision with location and velocity components. Details of processing and formatting of CARLA output can be found in Appendix A. This process produces measurements that match ground truth velocity and distance closely.

Although there are infinitely many possible physical situations where an FCW alert could occur involving two vehicles, they reside in a small set of equivalence classes. The National Highway Traffic Safety Administration (NHTSA) has outlined a set of testing conditions for assessing the efficacy of FCW alerts (National Highway Traffic Safety Administration 2011). It involves a two vehicles on a straight test track at varying speeds. Based on these real-world testing guidelines, we develop the following two scenarios:

MIO-10: Collision between two moving vehicles The ego and MIO travel on a straight road, with a negative relative velocity between the two vehicles. Specifically, the ego travels at 27 m/s (˜60 mph) and the MIO at 17 m/s (˜38 mph). These correspond to typical freeway speed differences of adjacent vehicles. In the absence of any other action, the ego will eventually collide with the MIO. In our simulations, we let this collision occur and record camera and RADAR measurements throughout. Since the relative velocity of the MIO to the ego is -10m/s, we refer to this dataset as MIO-10.

MIO+1: No collision The ego and MIO travel on a straight road, with a positive relative velocity between the two vehicles. Specifically, the ego travels at 27 m/s (˜60 mph) and the MIO at 28 m/s (˜63 mph). A trailing vehicle moving at 27 m/s follows the ego 7 m behind. In the absence of any other action, the ego and trailing vehicle will not collide. We collect measurements until the MIO moves out of sensor range of the ego. We refer to this dataset as MIO+1.

The above scenarios correspond to basic situations where the ego vehicle has an unobstructed view of the MIO and represents a best-case for the FCW system. Attacks on these two settings are the hardest to achieve and comprehensively demonstrate the efficacy of our MPC-based attack.

4.2 Attack Setup

We perform preprocessing of CARLA measurements to remove outliers and interpolate missing data (see Appendix C). Each step of our KF corresponds to one frame of the CARLA simulated video sequence (i.e., 0.05 seconds). We assume that the KF initializes its distance and velocity prediction to the average of the first vision and RADAR measurements. The acceleration is initialized to 0 in both directions. The covariance matrix is initialized to that used by Matlab FCW (MATLAB 2020b). Throughout the experiments, we let the effort matrix $R = I$, the margin parameter $\epsilon = 10^{-3}$, and $\lambda = 10^{10}$. We assume the human reaction time is $h^* = 24$ steps (i.e., 1.2 seconds in our simulation).

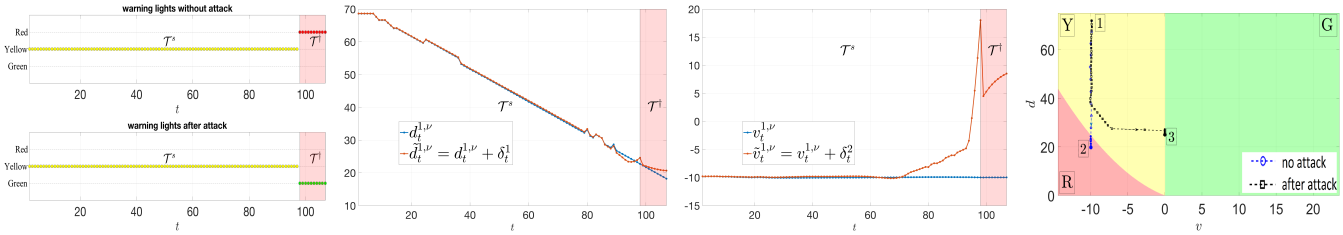
MIO-10 dataset We first simulate FCW to obtain the original warning lights without attack. The first red light appears at step 98. Before this step, the lights are all yellow. Without attack, the human driver will notice the red warning at step 98. After 1.2 seconds of reaction time (24 steps), the driver will start braking at step 122. The ground-truth distance to the MIO at the first application of brakes is 14.57m. During braking, the distance between the ego vehicle and the MIO reduces by $10^2/0.8g \approx 12.76$m before stabilizing. Since this is less than the ground-truth distance of 14.58m before braking, the crash can be avoided. This validates the potential effectiveness of FCW.

Our attacker aims to cause a crash. To accomplish this, the attacker suppresses the first 10 red warnings, so that the first red warning is delayed to step 108. As a result, the driver starts braking at step 132. The ground-truth distance to MIO at this step is 9.58m, which is below the minimum distance needed to avoid collision (12.76m). As such, a collision will occur. Therefore, we let the target interval be $\mathcal{T}^\dagger = [98, 107]$, and the target lights be $\ell_t^\dagger = \text{green}, \forall t \in \mathcal{T}^\dagger$.

MIO+1 dataset In this scenario, the original warning lights without attack are all green. There is a trailing vehicle 7 m behind the ego vehicle, driving at the same velocity as the ego vehicle. Our attacker aims at causing the FCW to output red lights, so that the ego vehicle suddenly brakes unnecessarily and causes a rear collision with the trailing vehicle. To this end, the attacker changes the green lights in the interval [100, 139] to red, in which case the ego vehicle driver starts braking at step 124, after 1.2 seconds of reaction time. If the warning returns to green at step 140, the driver will react after 1.2 seconds and stop braking at step 164. Therefore, the driver continuously brakes for at least $(164 - 124) \times 0.05 = 2$ seconds. Assuming the driver of the trailing vehicle is distracted, then during those 2 seconds, the distance between the trailing and the ego vehicle reduces by $0.2g \times 2^2 = 7.84m > 7m$, thus causing a rear-collision. Therefore, we let the target interval be $\mathcal{T}^\dagger = [100, 139]$ and the target lights be $\ell_t^\dagger = \text{red}, \forall t \in \mathcal{T}^\dagger$.

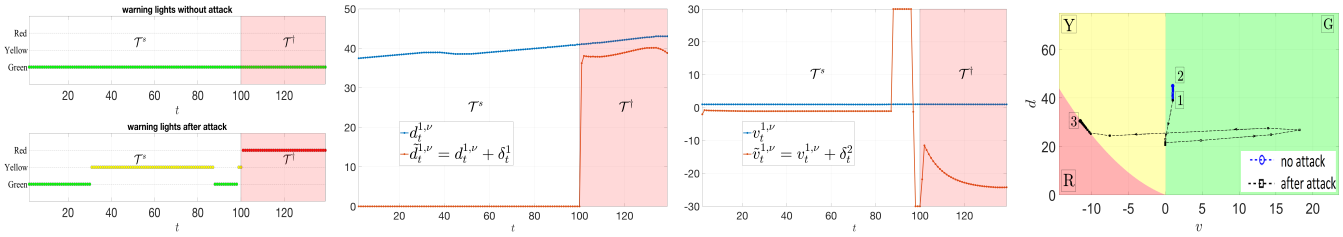
4.3 The MPC-based Attack Is Successful

Our first result shows that the MPC-based attack can successfully cause the FCW to output the desired warning lights in the target interval $\mathcal{T}^\dagger$. In this experiment, we let $\Delta = \infty$ and the stealthy interval $\mathcal{T}^s$ start at step 2. In Fig. 2a and 3a, we show the warning lights in $\mathcal{T}^\dagger$ (shaded in red). For MIO-10, the attacker achieves the desired red lights in the entire $\mathcal{T}^\dagger$, while maintaining the original yellow lights in $\mathcal{T}^s$. For



(a) The warning lights. (b) The manipulation on distance. (c) The manipulation on velocity. (d) The state trajectory.

Figure 2: Attacks on the MIO-10 dataset.



(a) The warning lights. (b) The manipulation on distance. (c) The manipulation on velocity. (d) The state trajectory.

Figure 3: Attacks on the MIO+1 dataset.

MIO+1, the attacker failed to achieve the red warning at step 100, but is successful in all later steps. We verified that the attack still leads to a collision. In fact, the attacker can tolerate at most two steps of failure in the beginning of $\mathcal{T}^\dagger$ while still ensuring that the collision occurs. There is an unintended side effect in $\mathcal{T}^s$ where green lights are changed to yellow. However, this side effect is minor since the driver will not brake when yellow lights are produced. In many production vehicles, green and yellow lights are not shown to the driver — only the red warnings are shown.

In Fig. 2b, 2c, we note that for MIO-10, the manipulation is mostly on velocity, and there are early planned manipulations starting from step 70. A large increase in velocity happens at step 100 (the first step of $\mathcal{T}^\dagger$), which causes the KF's velocity estimation to be positive, resulting in a green light. After that, velocity measurements are further increased to maintain a positive velocity estimation. In Fig. 3b, 3c, we show manipulations on MIO+1. The overall trend is that the attacker reduces the perceived MIO distance and velocity. As a result, KF estimates the MIO to be close than the safe distance in $\mathcal{T}^\dagger$, thus red lights are produced. During interval [88,96], There is an exceptional increase of velocity. We provide a detailed explanation for that increase in Appendix D.

In Fig. 2d, 3d, we show the trajectory of KF state prediction projected onto the distance-velocity space during interval $\mathcal{T}^a$. We partition the 2D space into three regions, green (G), yellow (Y) and red (R). Each region contains the states that trigger the corresponding warning light. The trajectory without attack (blue) starts from location 1 and ends at 2. After attack, the trajectory (dark) is steered into the region of the desired warning light, ending at location 3. Note that during $\mathcal{T}^\dagger$, the state after attack lies on the boundary of the desired region. This is because our attack minimizes manipulation effort. Forcing a state deeper into the desired region would require more effort, increasing the attacker's cost.

4.4 Attack Is Easier with More Planning Space

Our second result shows that the attack is easier when the attacker has more time to plan, or equivalently, a longer stealthy interval $\mathcal{T}^s$. The stealthy interval is initially of full length, which starts from step 2 until the last step prior to $\mathcal{T}^\dagger$. Then, we gradually reduce the length by 1/4 of the full length until the interval is empty. This corresponds to 5, 3.75, 2.5, 1.25 and 0 seconds of planning space before the target interval $\mathcal{T}^\dagger$. We denote the number of light violations in $\mathcal{T}^\dagger$ as $V^\dagger = \sum_{t\in\mathcal{T}^\dagger} \tilde{\ell}_t \neq \ell_t^\dagger$, and similarly V^s for $\mathcal{T}^s$. We let $\Delta = \infty$. In Table 1 and 2, we show $V^\dagger$, V^s together with J_1, J_2, J_3 and J as defined in (20) for MIO-10 and MIO+1 respectively. Note that on both datasets, the violation $V^\dagger$ and the total objective J decrease as the length of $\mathcal{T}^s$ grows, showing that the attacker can better accomplish the attack goal given a longer interval of planning.

On MIO-10, when $\mathcal{T}^s$ is empty, the attack fails to achieve the desired warning in all target steps. However, given 1.25s of planning before $\mathcal{T}^\dagger$, the attacker forces the desired lights throughout $\mathcal{T}^\dagger$. Similarly, on MIO+1, when $\mathcal{T}^s$ is empty, the attack fails in the first three steps of $\mathcal{T}^\dagger$, and the collision will not happen. Given 1.25s of planning before $\mathcal{T}^\dagger$, the attack only fails in the first step of $\mathcal{T}^\dagger$, and the collision happens. This demonstrates that planning in $\mathcal{T}^s$ benefits the attack.

4.5 Attack Is Easier as Δ Increases

In this section, we show that the attack becomes easier as the upper bound on the manipulation Δ grows. In this experiment, we focus on the MIO-10 dataset and let $\mathcal{T}^\dagger$ start from step 2. In Fig 4, we show the manipulation on measurements for $\Delta = 14, 16, 18$ and ∞. The number of green

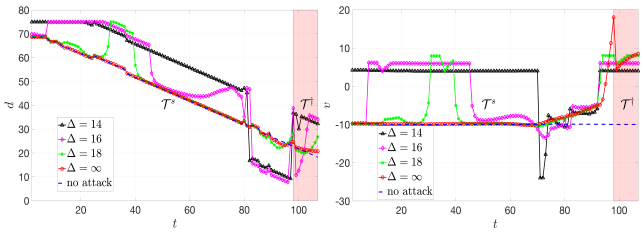
Table 1: $V^\dagger$, V^s, J_1, J_2, J_3 and J for the MIO-10 dataset.

	MPC-based attack						Greedy attack					
$\mathcal{T}^s$	$V^\dagger$	V^s	J_1	J_2	J_3	J	$V^\dagger$	V^s	J_1	J_2	J_3	J
0	1	0	7.1e3	0	7.4	7.4e10	1	0	4.6e3	98.4	7.4	1.1e12
1.25	0	0	4.4e3	0	0	4.3e3	0	23	1.3e5	3.3e3	0	3.3e13
2.5	0	0	4.4e3	0	0	4.4e3	0	47	2.0e5	5.4e3	0	5.4e13
3.75	0	0	4.4e3	0	0	4.4e3	0	71	2.5e5	7.6e3	0	7.5e13
5	0	0	4.4e3	0	0	4.4e3	0	96	2.9e5	9.2e3	0	9.2e13

Table 2: $V^\dagger$, V^s, J_1, J_2, J_3 and J for the MIO+1 dataset.

	MPC-based attack						Greedy attack					
$\mathcal{T}^s$	$V^\dagger$	V^s	J_1	J_2	J_3	J	$V^\dagger$	V^s	J_1	J_2	J_3	J
0	3	0	3.3e4	0	1.2e2	1.2e12	3	0	1.1e5	0	1.2e2	1.2e12
1.25	1	14	7.6e4	6.8	11.0	1.8e11	0	25	1.7e5	6.1e3	0	6.1e13
2.5	1	39	1.1e5	4.2	6.9	1.1e11	0	49	2.3e5	1.1e4	0	1.1e14
3.75	1	58	1.5e5	3.5	5.9	9.4e10	0	74	3.0e5	1.6e4	0	1.6e14
5	1	58	1.8e5	3.3	5.6	9.0e10	0	98	3.5e5	2.0e4	0	2.0e14

lights achieved by the attacker in the target interval is 0, 4, 10 and 10 respectively. This shows the attack is easier for larger Δ. Note that for smaller Δ, the attacker's manipulation becomes flatter due to the constraint $\|\delta_t\| \leq \Delta$. But, more interestingly, the attacker needs to start the attack earlier to compensate for the decreasing bound. We also note that the minimum Δ to achieve the desired green lights over the entire target interval (to integer precision) is 18.



(a) Manipulation on distance. (b) Manipulation on velocity.

Figure 4: Manipulation on measurements with different upper bound Δ. As Δ grows, the attack becomes easier.

4.6 Comparison Against Greedy Attacker

In this section, we introduce a greedy baseline attacker. For MIO-10, since the attack goal is to achieve green lights in $\mathcal{T}^\dagger$, the greedy attacker always increases the distance and velocity to the maximum possible value, i.e.,

$$\tilde{d}_t^{1,\nu} = \min\{d_t^{1,\nu}+\Delta, \bar{d}\}, \tilde{v}_t^{1,\nu} = \min\{v_t^{1,\nu}+\Delta, \bar{v}\}, \forall t \in \mathcal{T}^a.$$

Similarly, for MIO+1, the attacker always decreases the distance and velocity to the minimum possible value.

In table 1 and 2, we compare the performance of greedy and our MPC-based attack. On both datasets, each attack strategy achieves a small number of violations $V^\dagger$ in $\mathcal{T}^\dagger$. However, the greedy attack suffers significantly more violations V^s in $\mathcal{T}^s$ than does MPC. Furthermore, these violations are more severe, reflected by the much larger J_2 of the greedy attack. As an example, on MIO+1, the greedy attack changes the original green lights in $\mathcal{T}^s$ to red, while our attack only changes green to yellow. The greedy attack also results in larger total effort J_1 and objective value J. Therefore, we conclude that our attack outperforms the baseline greedy attack overall. In appendix F, we provide more detailed results of the greedy attack.

5 Related Work

Attacks on Object Tracking. Recent work has examined the vulnerability of multi-object tracking (MOT) (Jia et al. 2020). Although this work does consider the downstream logic that uses the outputs of ML-based computer vision, our work goes beyond in several ways. First, we consider a hybrid system that involves both human and machine. Second, we consider the more realistic case of sensor fusion involving RADAR and camera measurements that is deployed in production vehicles today. Prior work assumed a system that only uses a single camera sensor. Third, we examine a complete FCW pipeline that uses object tracking data to predict collisions and issue warnings. Prior work only considered MOT without any further logic that is necessarily present in realistic systems. Finally, our attack algorithm accounts for the sequential nature of decision making in ADAS.

Vision Adversarial Examples. ML models are vulnerable to adversarial examples (Szegedy et al. 2013), with a bulk of research in the computer vision space (Goodfellow, Shlens, and Szegedy 2014; Papernot et al. 2016; Carlini and Wagner 2017; Shafahi et al. 2018; Chen et al. 2017). Recent work has demonstrated physical attacks in the real world (Brown et al. 2017; Athalye et al. 2017; Eykholt et al. 2018a; Sharif et al. 2016). For example, attackers can throw inconspicuous stickers on stop signs and cause the model to output a speed limit sign (Eykholt et al. 2018b). However, all of this work studies the ML model in isolation without considering the cyber-physical system that uses model decisions. By contrast, we contribute the first study that examines the security of FCW — a hybrid human-machine system, and we introduce a novel control-based attack that accounts for these aspects while remaining stealthy to the human driver.

Control-based Attacks on KF. Prior work in control theory has studied false data injection attacks on Kalman filters (Bai, Gupta, and Pasqualetti 2017; Kung, Dey, and Shi 2016; Zhang and Venkitasubramaniam 2016; Chen, Kar, and Moura 2016; Yang, Chang, and Yu 2016; Chen, Kar, and Moura 2017). Our work assumes a similar attack modality – the attacker can manipulate measurements. However, prior work does not consider the downstream logic and human behavior that depends on the KF output. By contrast, we provide an attack framework demonstrating end-to-end effects that cause crashes in distracted driving scenarios.

Attacks on Sequential Systems. There are recent works that study attacks of other sequential learning systems from a control perspective (Chen and Zhu 2020; Zhang, Zhu, and Lessard 2020; Zhang et al. 2020; Jun et al. 2018). Most of them focus on analyzing theoretical attack properties, while we contribute an application of control-based attacks in a practical domain.

6 Conclusion

We formulate the adversarial attack of Kalman Filter as an optimal control problem. We demonstrate that our planning-based attack can manipulate the FCW to output incorrect warnings, which mislead human drivers to behave unsafely and cause crash. Our study incorporates human behaviors and applies to general machine-human hybrid systems.

7 Acknowledgments

This work was supported in part by the University of Wisconsin-Madison Office of the Vice Chancellor for Research and Graduate Education with funding from the Wisconsin Alumni Research Foundation. XZ acknowledges NSF grants 1545481, 1704117, 1836978, 2041428, 2023239 and MADLab AF CoE FA9550-18-1-0166.

8 Ethics Statement

Our paper studies attacks on advanced driver assistance systems (ADAS) with the goal of initiating research into defenses. We do not intend for the attacks to be deployed in the real world. However, studying attacks is critical to understanding what types of defenses must be built and where defense efforts should be focused. We take a first step towards robust ADAS by studying attacks on Kalman filters that are popularly used in these systems.

References

A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Sep. 2016. Simple Online and Realtime Tracking. *2016 IEEE International Conference on Image Processing* .

Athalye, A.; Engstrom, L.; Ilyas, A.; and Kwok, K. 2017. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397* .

Bai, C.-Z.; Gupta, V.; and Pasqualetti, F. 2017. On Kalman filtering with compromised sensors: Attack stealthiness and performance bounds. *IEEE Transactions on Automatic Control* 62(12): 6641–6648.

Brown, T. B.; Mané, D.; Roy, A.; Abadi, M.; and Gilmer, J. 2017. Adversarial Patch. *arXiv preprint arXiv:1712.09665* .

Carlini, N.; and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, 39–57. IEEE.

Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* .

Chen, Y.; Kar, S.; and Moura, J. M. 2016. Cyber physical attacks with control objectives and detection constraints. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, 1125–1130. IEEE.

Chen, Y.; Kar, S.; and Moura, J. M. 2017. Optimal attack strategies subject to detection constraints against cyber-physical systems. *IEEE Transactions on Control of Network Systems* 5(3): 1157–1168.

Chen, Y.; and Zhu, X. 2020. Optimal attack against autoregressive models by manipulating the environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3545–3552.

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017. CARLA: An Open Urban Driving Simulator. volume 78 of *Proceedings of Machine Learning Research*, 1–16. PMLR. URL http://proceedings.mlr.press/v78/dosovitskiy17a.html.

Edmund Optics. 2020. Understanding Focal Length and Field of View. URL https://www.edmundoptics.com/knowledge-center/application-notes/imaging/understanding-focal-length-and-field-of-view/.

European New Car Assessment Programme. 2018. Euro NCAP LSS Test Protocol. Version 2.0.1. URL https://cdn.euroncap.com/media/26996/euro-ncap-aeb-c2c-test-protocol-v20.pdf.

Eykholt, K.; Evtimov, I.; Fernandes, E.; Li, B.; Rahmati, A.; Tramèr, F.; Prakash, A.; Kohno, T.; and Song, D. 2018a. Physical Adversarial Examples for Object Detectors. In *Proceedings of the 12th USENIX Conference on Offensive Technologies*, WOOT'18.

Eykholt, K.; Evtimov, I.; Fernandes, E.; Li, B.; Rahmati, A.; Xiao, C.; Prakash, A.; Kohno, T.; and Song, D. 2018b. Robust Physical-World Attacks on Deep Learning Visual Classification. In *Computer Vision and Pattern Recognition (CVPR)*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* .

Jia, Y.; Lu, Y.; Shen, J.; Chen, Q. A.; Chen, H.; Zhong, Z.; and Wei, T. 2020. Fooling Detection Alone is Not Enough: Adversarial Attack against Multiple Object Tracking. *2020 International Conference on Learning Representations (ICLR)* .

Joseph A. Gregor. 2017. Tesla Driver Assistance System. URL https://dms.ntsb.gov/public/59500-59999/59989/604889.pdf.

Jun, K.-S.; Li, L.; Ma, Y.; and Zhu, J. 2018. Adversarial attacks on stochastic bandits. *Advances in Neural Information Processing Systems* 31: 3640–3649.

Kuhn, H. W. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly, vol. 2, no. 1–2* .

Kung, E.; Dey, S.; and Shi, L. 2016. The Performance and Limitations of ϵ-Stealthy Attacks on Higher Order Systems. *IEEE Transactions on Automatic Control* 62(2): 941–947.

MATLAB. 2020a. Detect vehicles using YOLO v2 Network - MATLAB vehicleDetectorYOLOv2. URL https://www.mathworks.com/help/driving/ref/vehicledetectoryolov2.html.

MATLAB. 2020b. Forward Collision Warning Using Sensor Fusion. URL https://www.mathworks.com/help/driving/examples/forward-collision-warning-using-sensor-fusion.html.

Murray, S. 2017. Real-Time Multiple Object Tracking - A Study on the Importance of Speed. *arXiv:1709.03572 [cs]* URL http://arxiv.org/abs/1709.03572.

National Highway Traffic Safety Administration. 2011. A Test Track Protocol for Assessing Forward Collision Warning Driver-Vehicle Interface Effectiveness. URL https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/811501.pdf.

National Highway Traffic Safety Administration. 2020. Common Driver Assistance Technologies. URL https://www.nhtsa.gov/equipment/driver-assistance-technologies.

Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z. B.; and Swami, A. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, 372–387. IEEE.

R. Collins. 2007. Lecture 12: Camera Projection. URL http://www.cse.psu.edu/~rtc12/CSE486/lecture12.pdf.

Redmon, J.; Divvala, S.; Girshick, R. B.; and Farhadi, A. 2016. You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 779–788.

Shafahi, A.; Huang, W. R.; Najibi, M.; Suciu, O.; Studer, C.; Dumitras, T.; and Goldstein, T. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, 6103–6113.

Sharif, M.; Bhagavatula, S.; Bauer, L.; and Reiter, M. K. 2016. Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, 1528–1540.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* .

Yang, Q.; Chang, L.; and Yu, W. 2016. On false data injection attacks against Kalman filtering in power system dynamic state estimation. *Security and Communication Networks* 9(9): 833–849.

Zhang, R.; and Venkitasubramaniam, P. 2016. Stealthy control signal attacks in vector lqg systems. In *2016 American Control Conference (ACC)*, 1179–1184. IEEE.

Zhang, X.; Ma, Y.; Singla, A.; and Zhu, X. 2020. Adaptive Reward-Poisoning Attacks against Reinforcement Learning. *arXiv preprint arXiv:2003.12613* .

Zhang, X.; Zhu, X.; and Lessard, L. 2020. Online data poisoning attacks. In *Learning for Dynamics and Control*, 201–210. PMLR.

A Simulated Raw Data Processing

CARLA outputs a single RGB image, a depth map image, and variable number of RADAR points for each 0.05 second time step of the simulation. We analyze this data at each time step to produce object detections in the same format of MATLAB FCW (MATLAB 2020b):

$$\begin{bmatrix} d^1 & v^1 & d^2 & v^2 \end{bmatrix}$$

where d^1 and d^2 are the distance, in meters, from the vehicle sensor in directions parallel and perpendicular to the vehicle's motion, respectively. v^1 and v^2 are the detected object velocities, in m/s, relative to the ego along these parallel and perpendicular axes.

To produce these detections from vision data, we first find bounding boxes around probable vehicles in each RGB image frame using an implementation of a YOLOv2 network in MATLAB which has been pre-trained on vehicle images (MATLAB 2020a). Each bounding box is used to create a distinct object detection. The d^1 value, or depth, of each object is taken to be the depth recorded by the depth map at the center pixel of each bounding box.

The d^2 value of each detection is then computed as

$$d^2 = u * \frac{d^1}{l_{foc}} \tag{27}$$

where u is the horizontal pixel coordinate of the center of a bounding box in a frame, and l_{foc} is the focal length of the RGB camera in pixels (R. Collins 2007). l_{foc} is not directly specified by CARLA, but can be computed using the image length, 800 pixels, and the camera field of vision, 90 degrees (Edmund Optics 2020).

To compute v^1 and v^2 for detections of the current time step, we also consider detections from the previous time step. First, we attempt to match each bounding box from the current time step to a single bounding box from the previous step. Box pairs are evaluated based on their Intersection-Over-Union (IoU) (A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft Sep. 2016). Valued between 0 and 1, a high IoU indicates high similarity of size and position of two boxes, and we enforce a minimum threshold of 0.4 for any two boxes to be paired. For two adjacent time steps, A and B, we take the IoU of all possible pairs of bounding boxes with one box from step A, and one from B. These IoU values form the cost matrix for the Hungarian matching algorithm (Murray 2017), which produces the best possible pairings of bounding boxes from the current time step to the previous.

This matching process results in a set of detections with paired bounding boxes, and a set with unpaired boxes. For each detection with a paired box, we calculate its velocity simply as the difference between respective d^1 and d^2 values of the current detection and its paired observation from the previous time step, multiplied by the frame rate, fps_{cam}. For a detection, a, paired with a previous detection, b:

$$< v_a^1, v_a^2 > = < d_b^1 - d_a^1, d_b^2 - d_a^2 > * fps_{cam} \tag{28}$$

For each detection left unpaired after Hungarian matching, we make no conclusions about v^1 or v^2 for that detection, and treat each as zero.

Each RADAR measurement output by CARLA represents an additional object detection. RADAR measurements contain altitude (al) and azimuth (az) angle measurements, as well as depth (d) and velocity (v), all relative to the RADAR sensor. We convert these measurements into object detection parameters as follows

$$d^1 = d * \cos az * \cos al \qquad v^1 = v * \cos az * \cos al$$
$$d^2 = d * \sin az * \cos al \qquad v^2 = v * \sin az * \cos al$$

B Derivation of Surrogate Constraints

The original attack optimization (6)-(13) may not be convex due to that (12) and (13) could be nonlinear. Our goal in this section is to derive convex surrogate constraints that are good approximations to (12) and (13). Furthermore, we require the surrogate constraints to be tighter than the original constraints, so that solving the attack under the surrogate constraints will always give us a feasible solution to the original attack. Concretely, we want to obtain surrogate constraints to $F(x) = \ell$, where $\ell \in \{\text{green, yellow, red}\}$. We analyze each case of ℓ separately:

- $\ell =$ green

  In this case, $F(x) = \ell$ is equivalent to $v \geq 0$ according to (5). While this constraint is convex, when we actually solve the optimization, it might be violated due to numerical inaccuracy. To avoid such numerical issues, we tighten it by adding a margin parameter $\epsilon > 0$, and the derived surrogate constraint is $v \geq \epsilon$.

- $\ell =$ red

  In this case, $F(x) = \ell$ is equivalent to

  $$v \quad < 0 \tag{29}$$
  $$d \quad \leq -1.2v + \frac{1}{0.8g} v^2. \tag{30}$$

  Similar to case 1, we tighten the first constraint as

  $$v \leq -\epsilon. \tag{31}$$

  Note that by the first constraint, we must have $v < 0$. The second constraint is $d \leq -1.2v + \frac{1}{0.8g} v^2$. Given $v < 0$, this is equivalent to

  $$v \leq 0.48g - \sqrt{(0.48g)^2 + 0.8gd}. \tag{32}$$

  We next define the following function

  $$U(d) = 0.48g - \sqrt{(0.48g)^2 + 0.8gd}. \tag{33}$$

  The first derivative is

  $$U'(d) = -\frac{0.4g}{\sqrt{(0.48g)^2 + 0.8gd}}, \tag{34}$$

  which is increasing when $d \geq 0$. Therefore, the function $U(d)$ is convex. We now fit a linear function that lower bounds $U(d)$. Specifically, since $U(d)$ is convex, for any $d_0 \geq 0$, we have

  $$U(d) \geq U'(d_0)(d - d_0) + U(d_0). \tag{35}$$

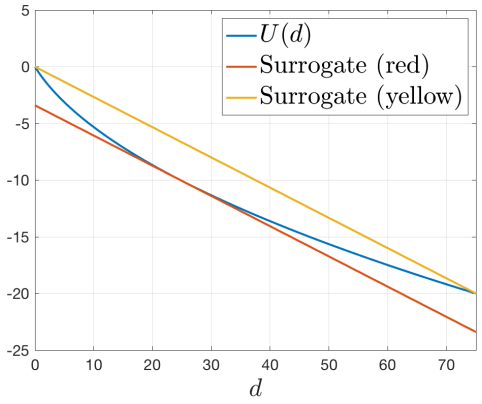


Figure 5: Surrogate light constraints.

Therefore, $v \leq U'(d_0)(d - d_0) + U(d_0)$ is a tighter constraint than $v \leq U(d)$. The two constraints are equivalent at $d = d_0$. Again, we need to add a margin parameter to avoid constraint violation due to numerical inaccuracy. With this in mind, the surrogate constraint becomes

$$v \leq U'(d_0)(d - d_0) + U(d_0) - \epsilon, \tag{36}$$

Or, equivalently:

$$v \leq -\epsilon, \tag{37}$$
$$v \leq U'(d_0)(d - d_0) + U(d_0) - \epsilon, \tag{38}$$

This concludes the proof of our Proposition 1.

However, we still need to pick an appropriate d_0. In our scenario, the distance d has physical limitation $d \in [\underline{d}, \bar{d}]$ with $\underline{d} = 0$ and $\bar{d} = 75$. The $U(d)$ curve for $d \in [0, 75]$ is shown in Fig 5. Based on the figure, we select d_0 such that $U'(d_0)$ is equal to the slope of the segment connecting the two end points of the curve, i.e.,

$$U'(d_0) = \frac{U(75) - U(0)}{75} = \frac{U(75)}{75}. \tag{39}$$

We now derive the concrete surrogate constraints used in our experiment section. We begin with the following equation:

$$0.48g + \frac{0.4g}{U'(d)} = U(d). \tag{40}$$

From which, we can derive d_0:

$$d_0 = \frac{1}{0.8g}\left((\frac{30g}{U(75)})^2 - (0.48g)^2\right) \tag{41}$$

and

$$U(d_0) = 0.48g + \frac{30g}{U(75)}. \tag{42}$$

By substituting d_0 and $U(d_0)$ into (36), we find that the surrogate constraint is

$$v \leq \frac{U(75)}{75}(d - d_0) + 0.48g + \frac{30g}{U(75)} + \epsilon. \tag{43}$$

- $\ell =$ yellow

  In this case, $F(x) = \ell$ is equivalent to

$$v < 0 \tag{44}$$
$$d \geq -1.2v + \frac{1}{0.8g}v^2. \tag{45}$$

  Similarly, we tighten the first constraint to

$$v \leq -\epsilon. \tag{46}$$

  For the second constraint, the situation is similar to $\ell =$ red. $\forall d_0 > 0$. We have

$$v \geq \frac{U(d_0)}{d_0}d, \forall d \in [0, d_0] \tag{47}$$

  The above inequality is derived by fitting a linear function that is always above the $U(d)$ curve. Next, we select $d_0 = 75$ and add a margin parameter ϵ to derive the surrogate constraint:

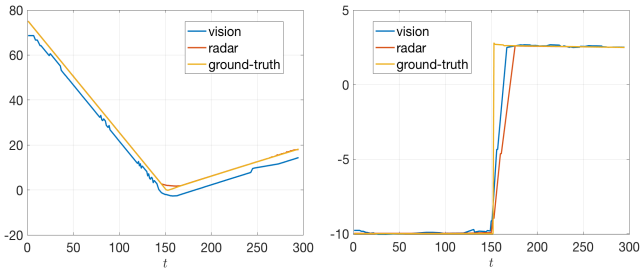
$$v \leq -\epsilon \tag{48}$$
$$v \geq \frac{U(75)}{75}d + \epsilon. \tag{49}$$

  To summarize, we have derived surrogate constraints for $F(x) = \ell$, where $l \in \{\text{green, yellow, red}\}$. When we solve the attack optimization, we replace each individual constraint of (12) and (13) by one of the above three surrogate constraints. In Fig 5, we show the surrogate constraints for red and yellow lights with $\epsilon = 10^{-3}$.

C Preprocessing of CARLA Measurements

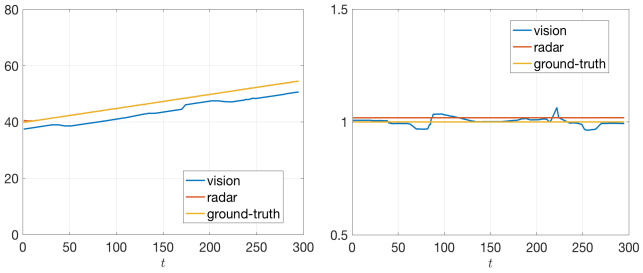
In this section, we describe how we preprocess the measurements obtained from CARLA simulation. The measurement in each time step takes the form of $t_t = [y_t^1, y_t^2] \in \{\mathbb{R} \cup \text{NaN}\}^8$, where $y_t^1 \in \{\mathbb{R} \cup \text{NaN}\}^4$ is the vision detection produced by ML-based objection detection algorithm YOLOv2, and y_t^2 is the detection generated by radar (details in Appendix A). Both vision and radar measurements contain four components: (1) the distance to MIO along driving direction, (2) the velocity of MIO along driving direction, (3) the distance to MIO along lateral direction, and (4) the velocity of MIO along lateral direction. The radar measurements are relatively accurate, and do not have missing data or outliers. However, there are missing data (NaN) and outliers in vision measurements. The missing data problem arises because the MIO sometimes cannot be detected, e.g., in the beginning of the video sequence when the MIO is out of the detection range of the camera. Outliers occur because YOLOv2 may not generate an accurate bounding box of the MIO, causing it to correspond to a depth map reading of an object at a different physical location. As such, a small inaccuracy in the location of the bounding box could lead to dramatic change to the reported distance and velocity of the MIO.

In our experiment, we preprocess detections output from CARLA to address missing data and outlier issues. First, we identify the outliers by the Matlab "filloutliers" method,



(a) Distance measurements. (b) Velocity measurements.

Figure 6: On the MIO-10 dataset, the preprocessed vision measurements and the radar measurements match the ground-truth reasonably well.



(a) Distance measurements. (b) Velocity measurements.

Figure 7: On the MIO+1 dataset, the preprocessed vision measurements and the radar measurements match the ground-truth reasonably well.

where we choose "movmedian" as the detector and use linear interpolation to replace the outliers. The concrete Matlab command is:

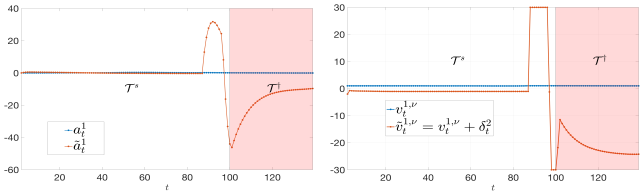
filloutliers(Y, 'linear', 'movmedian', 0, 'ThresholdFactor', 0.5),

where $Y \in \mathbb{R}^{T\times 8}$ is the matrix of measurements and T is the total number steps. In our experiment $T = 295$. We perform the above outlier detection and replace operation twice to smooth the measurements.

Then, we apply the Matlab "impute" function to interpolate the missing vision measurements. In Fig 6 and 7, we show the preprocessed distance and velocity measurements from vision and radar compared with the ground-truth for both MIO-10 and MIO+1 datasets. Note that after preprocessing, both radar and vision measurements match with the ground-truth well.

D Velocity Increase in Figure 3c

In Figure 8b, we show again the manipulation on the velocity measurement for the MIO+1 dataset. The attacker's goal is to cause the FCW to output red warnings in the target interval $[100, 139]$. Intuitively, the attacker should decrease the distance and velocity. However, in Figure 8b, the attacker instead chooses to increase the velocity during interval [88,96]. We note that this is because the attacker hopes to force a very negative KF acceleration estimation. To accomplish that, the attacker first strategically increases the velocity from step 88 to 96, and then starting from step 97, the



(a) Acceleration estimation. (b) Manipulation on velocity.

Figure 8: Acceleration reduces significantly as the velocity measurement drops after step 96. This in turn causes the KF velocity estimation to decrease fast.

attacker suddenly decreases the velocity dramatically. This misleads the KF to believe that the MIO has a very negative acceleration. In Fig 8a, we show the acceleration estimation produced by KF. At step 96, the estimated acceleration is 8.1m/s^2. However, at step 97, the estimated acceleration suddenly drops to -16m/s^2, and then stays near -30m/s^2 until the target interval. The very negative acceleration in turn causes the KF velocity estimation to decrease quickly. The resulting velocity estimation reached around -10m/s during the target interval, which causes FCW to output red lights.

E Human Behavior Algorithm

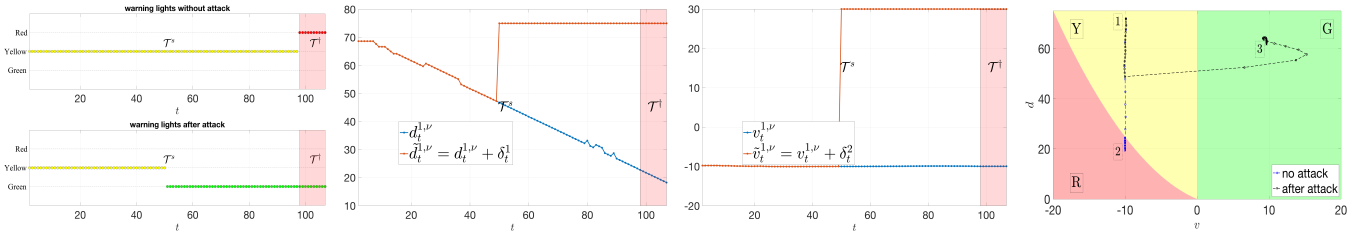
In this section, we provide an algorithmic description of the human behavior model.

Input: light sequence $\ell_t (1 \le t \le T)$, reaction time h^*.
Initialize $s = 0$;
for $t \leftarrow 1$ **to** T **do**
  if $t! = 1$ *and* $\ell_t! = \ell_{t-1}$ **then**
    $s = 0$;
  else if $\ell_t =$ *red* **then**
    $s = s + 1$;
  else
    $s = s - 1$;
  end
  if $s \ge h^*$ **then**
    human applies pressure on pedal;
  else if $s \le -h^*$ **then**
    human releases brake;
  else
    human stays in the previous state;
  end
end

Algorithm 2: Human Behavior Algorithm.

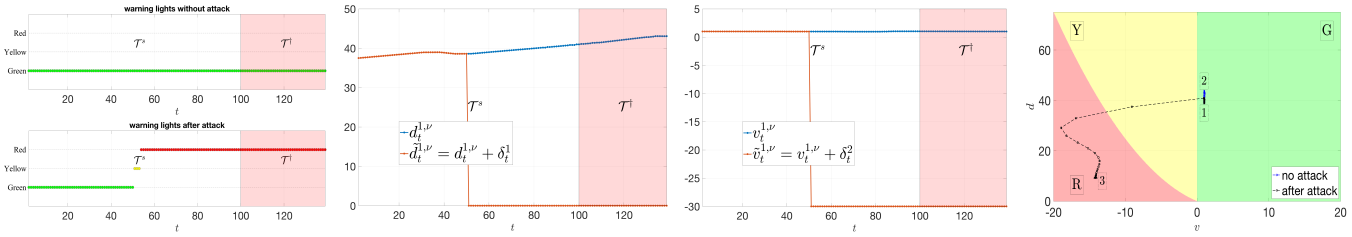
F Detailed Results of Greedy Attack

In this section, we provide more detailed results of the greedy attack, including warning lights before and after attack, manipulations on measurements, and the trajectory of KF state predictions. We notice that the results are very similar for different lengths of the stealthy interval $\mathcal{T}^s$. There-



(a) The warning lights. (b) The manipulation on distance. (c) The manipulation on velocity. (d) The state trajectory.

Figure 9: Greedy attack on the MIO-10 dataset.



(a) The warning lights. (b) The manipulation on distance. (c) The manipulation on velocity. (d) The state trajectory.

Figure 10: Greedy attack on the MIO+1 dataset.

fore, here we only show the results for $\mathcal{T}^s = 2.5$ seconds (i.e., half of the full length) as an example.

Fig. 9 shows the greedy attack on MIO-10, where the stealthy interval $\mathcal{T}^s = [50, 97]$. By manipulating the distance and velocity to the maximum possible value, the attacker successfully causes the FCW to output green lights in the target interval $\mathcal{T}^\dagger$. However, the attack induces side effect in $\mathcal{T}^s$, where the original yellow lights are changed to green. In contrast, our MPC-based attack does not have any side effect during $\mathcal{T}^s$. Also note that the trajectory of the KF state prediction enters "into" the desired green region during $\mathcal{T}^\dagger$. This is more than necessary and requires larger total manipulation (J_1) than forcing states just on the boundary of the desired region, as does our attack.

In Fig. 10, we show the greedy attack on MIO+1. The stealthy interval is $\mathcal{T}^s = [51, 99]$. Again, the attack results in side effect during the stealthy interval $\mathcal{T}^s$. Furthermore, the side effect is much more severe (green to red) than that of our MPC-based attack (green to yellow). The KF state trajectory enters "into" the desired red region, and requires larger total manipulation (J_1) than our attack.