SlowTrack: Increasing the Latency of Camera-Based Perception in Autonomous Driving Using Adversarial Examples

Chen Ma[1*], **Ningfei Wang**[2*], **Qi Alfred Chen**[2], **Chao Shen**[1]
[1]Xi'an Jiaotong University
[2]University of California, Irvine
ershang@stu.xjtu.edu.cn, ningfei.wang@uci.edu, alfchen@uci.edu, chaoshen@xjtu.edu.cn

Abstract

In Autonomous Driving (AD), real-time perception is a critical component responsible for detecting surrounding objects to ensure safe driving. While researchers have extensively explored the integrity of AD perception due to its safety and security implications, the aspect of availability (real-time performance) or latency has received limited attention. Existing works on latency-based attack have focused mainly on *object detection*, i.e., a component in camera-based AD perception, overlooking the entire camera-based AD perception, which hinders them to achieve effective system-level effects, such as vehicle crashes. In this paper, we propose SlowTrack, a novel framework for generating adversarial attacks to increase the execution time of camera-based AD perception. We propose a novel two-stage attack strategy along with the three new loss function designs. Our evaluation is conducted on four popular camera-based AD perception pipelines, and the results demonstrate that SlowTrack significantly outperforms existing latency-based attacks while maintaining comparable imperceptibility levels. Furthermore, we perform the evaluation on Baidu Apollo, an industry-grade full-stack AD system, and LGSVL, a production-grade AD simulator, with two scenarios to compare the system-level effects of SlowTrack and existing attacks. Our evaluation results show that the system-level effects can be significantly improved, i.e., the vehicle crash rate of SlowTrack is around 95% on average while existing works only have around 30%.

Introduction

Autonomous Driving (AD) vehicles, manufactured by various companies, have become ubiquitous in our daily lives. For instance, numerous Tesla vehicles are equipped with the Autopilot feature (Kane 2021; Tesla 2022) running in the real world. For these vehicles, camera-based *perception* is pivotal, enabling them to detect real-time environmental objects such as pedestrians to ensure safety. Given its significance for safety and security, various prior works (Cao et al. 2021; Shen et al. 2022; Sato et al. 2021a; Wang et al. 2023) have studied its security, especially on integrity such as making the object vanished or changing the label of the objects to cause traffic rule violations or safety hazards. We refer to these as *system-level effects* throughout this paper.

Nevertheless, the *availability* aspect (real-time performance) of the system, which is crucial for safety (e.g., causing vehicle collision (Wan et al. 2022)) has been relatively underexplored, especially for the complete camera-based AD perception pipeline. While some existing AD security analysis has studied availability in object detection (Shapira et al. 2023; Chen et al. 2023), they do not encompass the entire AD perception since usually, object detection is a part of the AD perception (Jia et al. 2020). In addition, in the Cyber-Physical System area, it is widely recognized that small component level errors do not necessarily lead to system-level effects (Shen et al. 2022; Wang et al. 2023). Thus, these studies leave a critical research gap: *their proposed attack strategies may not be effective enough to conduct system-level effects in end-to-end AD systems*. As we demonstrate later, existing attacks targeting only object detection do not consistently produce highly potent system-level effects due to lack of entire AD perception consideration.

To fill in this critical research gap, in this paper, we are the first to study availability-based adversarial attacks across the entire camera-based AD perception including both object detection and tracking. Our proposed novel attack framework, SlowTrack, is designed to increase the latency of camera-based AD perception. Instead of solely targeting object detection, which might not yield potent system-level effects due to the limited increase of the latency, we realize the untapped potential of object tracking response time to generate a much more effective latency attack. To illustrate, an attacker focusing only on object detection might attempt to dramatically increase the number of proposed bounding boxes (Chen et al. 2023). Object tracking might filter out a majority of these boxes and in common object detection post-processing (Jocher 2020; Zhang et al. 2021), the maximum number of detection is provided to ensure performance. Thus, the effectiveness of these attack is limited. Due to the importance of object tracking, we first perform availability attack surface analysis by theoretically analyzing the time complexity of the state-of-the-art representative tracking algorithms. Then, we propose a two-stage attack strategy and formulate the attack as an optimization problem, shown in Fig. 1. Additionally, our novel loss function designs, encompassing score loss, bounding box area loss, and feature match loss, fully leverage the entire tracking-by-detection pipeline to generate effective latency-based attack.

[*]These authors contributed equally.

Our experimental evaluation of SlowTrack targets four state-of-the-art camera-based AD perception pipelines. We find that SlowTrack, when compared with existing object detection latency attacks, provides significant improvements in latency under comparable perturbation levels. For instance, SlowTrack on average induces latency 2.9 times more than that for existing approaches (Chen et al. 2023; Shapira et al. 2023). We also demonstrate the system-level effects of our SlowTrack using Baidu Apollo (Apollo 2023) LGSVL (Rong et al. 2020) AD simulator. The results show that SlowTrack induces a 70% vehicle crash rate in two representative AD scenarios while existing methods achieve only a 30% rate. Demo videos are at the project website: https://sites.google.com/view/cav-sec/slowtrack

To sum up, our contributions are as follows:

- We are the first to study availability-based adversarial attacks considering the entire AD perception pipeline and find that previous object detection-based latency attack strategies may not induce potent system-level effects.
- We propose a novel attack framework SlowTrack to systematically generate the latency adversarial attacks on camera-based AD perception by designing a two-stage attack strategy and proposing three novel loss functions.
- SlowTrack is tested on four popular camera-based AD perception pipelines across different hardware, showing increase in latency and boost in system-level effects.

Background and Related Work

Camera-based AD perception. In the AD system, camera-based perception is primarily constituted by object detection and multi-object tracking (MOT) (Apollo 2023; Kato et al. 2018). This process aims to identify objects in each image frame and track their movement over time (Jia et al. 2020). *Tracking-by-detection* has become the dominant MOT paradigm (Zhang et al. 2022) and is widely used in industry-grade full-stack AD systems such as Baidu Apollo (Apollo 2023) and Autoware.AI (Kato et al. 2018). It incorporates a detection module, a data association module, and a tracker management module. The detection module identifies objects in an image, noting their location, confidence, class scores, as well as other features for later data association. Data association then compares these detection with existing trackers based on features such as location and appearance, matching them based on similarity.

Tracking-by-detection, despite varying in matching strategies, shares a similar tracking management (Jia et al. 2020) to build and delete the moving trajectories, called trackers, and mark trackers and detection boxes as different states. Specifically, unmatched detection boxes are marked as unconfirmed and will be deleted unless they are continuously detected for *H* frames. Matched trackers are marked as re-find or remain activated depending on the trackers' previous states, while unmatched trackers are marked as lost, and will be deleted if no objects are associated with them for *R* frames. All of these trackers involved in matching constitute tracker_pool and trackers with activated states are outputs.

Prior works (Jia et al. 2020; Ma et al. 2023b; Shen et al. 2022) show that MOT poses a general challenge to cause AD system-level attack impact for existing attacks that target object detection since MOT is designed to be robust against errors in object detection. Given this challenge, our work introduces an innovative latency attack against the most representative and popular MOT (tracking-by-detection) and demonstrates the heightened system-level attack effects in AD.

Adversarial attack in AD. DNNs are vulnerable to adversarial attacks (Carlini and Wagner 2017; Sato et al. 2023; Luo et al. 2022), which are maliciously crafted samples to force DNNs to misbehave. Various prior works have explored the adversarial attacks in AD (Cao et al. 2020; Sato et al. 2020a, 2021b, 2020b; Muller et al. 2022; DiPalma et al. 2021). While a majority of these attacks target integrity, our research concentrates on availability, which is another critical problem in AD (Wan et al. 2022). Although some attack works study availability (Liu et al. 2023; Chen et al. 2023; Shapira et al. 2023; Wang et al. 2021), none of them consider the whole AD perception pipelines, which leads to suboptimal system-level effects in AD (Jia et al. 2020).

Availability-based latency attack. Availability-based latency attack can induce delays in the outputting function. Such adversarial methods, when applied to DNN, have been investigated recently (Shapira et al. 2023; Chen et al. 2023; Liu et al. 2023). However, their oversight of MOT within AD perception restricts their potential to achieve potent system-level effects. Thus, in this work, we perform the first availability-based latency attack on the whole AD perception to significantly boost system-level effects.

Methodology

Availability Attack Surface Analysis

To understand the vulnerability of the tracking-by-detection paradigm to latency attacks, we analyze the time complexity of the main three key steps (detection, data association, and tracker management) presented in Algorithm 1. The time consumption of the image preprocessing and backbone network of the detector hinges upon the computational dimension (Shumailov et al. 2021) and the number of computations (Hong et al. 2020; Haque et al. 2020). Given that the dimensions of the input images are unchanged and the majority of the detection model activation values are inherently non-zero for the most of images (Shapira et al. 2023), we do not prioritize the time complexity of this segment.

The boxes obtained by the detection module usually need to be filtered before being passed to subsequent modules. Prevailing filtration techniques encompass non-maximum suppression (NMS) and score filtering with time complexity of $O(n^2)$ and $O(n)$ respectively, where n denotes the number of outputs from the detection network. However, since most tracking algorithms (Zhang et al. 2021) confine the maximum number of detection to $|\mathcal{D}|_{max}$, the maximum time complexities of these are $O(|\mathcal{D}|^2_{max})$ and $O(|\mathcal{D}|_{max})$. Then, data association matches the reserved detection boxes and tracker_pool with features, and the time complexity of this process is $O(mn')$, where m represents the number of trackers in tracker_pool and n' represents the number of reserved detection boxes. In the tracker management module, trackers are created and deleted according to the matching

Algorithm 1: Tracking-by-detection

Input: Video sequence V; detector Det; detection filter F
Output: Activated trackers $\mathcal{T}$ of the video

Initialization: $\mathcal{T} \leftarrow \emptyset$
for *frame* f_k *in* V **do**
  /* detection module */
  $\mathcal{D}_k \leftarrow \texttt{Det}(f_k)$
  $\downarrow O(n^2) + O(n)$
  $\mathcal{D}_{reserved} \leftarrow F(D_k)$
  $\downarrow O(m)$
  /* predict locations of trackers */
  for t *in* $\mathcal{T}$ **do**
    $t \leftarrow \texttt{KalmanFilter}(t)$
  end
  $\downarrow O(mn')$
  /* trackers management */
  Details in (Ma et al. 2023a)
end
Return: $\mathcal{T}$

results and are marked with different states with a time complexity of $O(m)$. Meanwhile, the velocity and location of the trackers are also updated with a time complexity of $O(m)$.

Threat Model, Formulation, and Attack Overview

Threat model. Our attack method assumes white-box settings for the detection model, wherein both its architecture and parameters are known. For the tracking, we do not force the attackers to know the specific parameters and implementation details, which is a similar threat model as prior latency attack works (Shapira et al. 2023; Chen et al. 2023).

Formulation. Detectors often restrict the maximum number of detection boxes. This constraint results in a worst-case time complexity for the detection module of $O(|\mathcal{D}|^2_{max})$, effectively reducing the impact of previous latency attacks on object tracking (Chen et al. 2023; Shapira et al. 2023). Thus, we propose an attack methodology that focuses on increasing the latency of the subsequent tracking stage under the constraint of limiting the maximum number of detection boxes. For a camera-based perception pipeline P, given the original image x, the attack goal is to craft an adversarial example x^* to maximize the tracking pipeline latency, while keeping the added adversarial perturbations imperceptible. We formulate it as the following optimization problem:

$$\arg\max_{x^*} T(P(x^*)) \quad \text{s.t.} \begin{cases} |\mathcal{D}|_{max} = N \\ \Delta(x^*, x) \leq \epsilon \end{cases} \tag{1}$$

, where T indicates the time function. Our attack strategy is designed to create detection boxes that exploit vulnerabilities in the tracker management module of AD. Specifically, for avoiding the temporary loss of objects in consecutive video frames due to occlusion, etc., the lost tracker will not be deleted immediately until the object is lost for R consecutive frames. Leveraging this mechanism, we are able to inject more and more trackers into the tracking module by strategically creating detection boxes through carefully designed perturbations. The detection boxes that appear in each frame are not associated with existing trackers

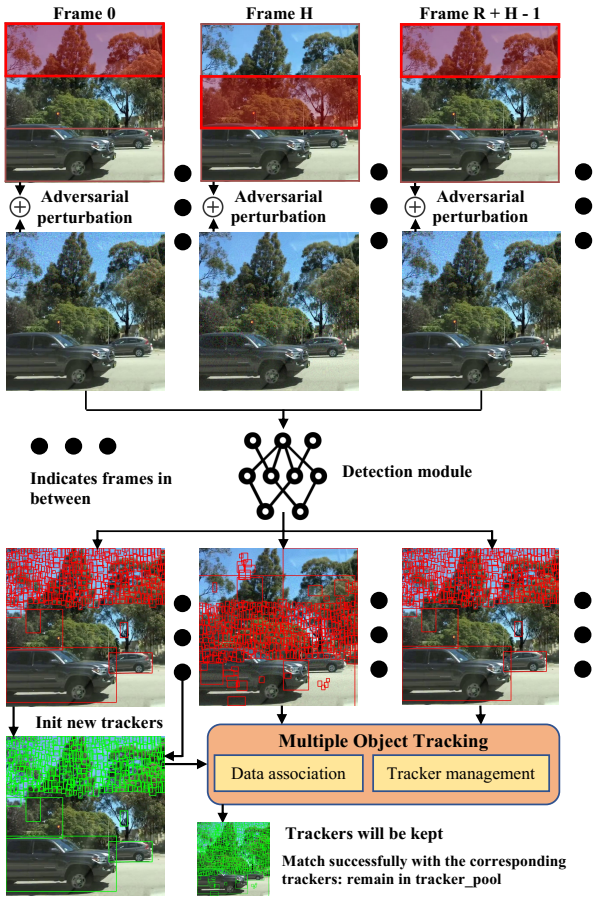


Figure 1: Overview of our SlowTrack attack.

and cause new tracking boxes to be created. As a result, the worst-case time complexity of the tracking module under our attack method is $O(R|\mathcal{D}|^2_{max})$, where $m = R|\mathcal{D}|_{max}$ and $n' = |D|_{max}$. Thus, it can lead to more computation cost than detection attacks in prior works.

Overview. In this paper, we propose SlowTrack, the first adversarial attack maximizing the latency of the whole camera-based perception pipeline, leveraging object tracking, which can significantly increase the latency of AD perception under the constraint shown in Eq (1). The two attack stages of SlowTrack are: 1) in the attack initialization stage, make detection boxes created as many new trackers as possible, which requires detection boxes not to be associated with existing trackers, 2) make the lost trackers re-found before they are deleted, and kept in track_pool, which requires detection boxes to be associated with corresponding trackers. Thus, we need to construct sets of detection boxes with the same matching features and make these sets of boxes appear or disappear in the corresponding video frames using adversarial attacks. To our best knowledge, representative tracking algorithms always use motion-based features for association. Thus, we divide the images into different regions and use the candidates in these regions as sets, which facilitates the inter-association of detection boxes within sets and the

disassociation of detection boxes between sets.

The overview of SlowTrack is in Fig. 1. Assuming that we divide the image into 3 regions, in frame 0, we select the top region and make the detection boxes appear in it, which are initialized as new track boxes. Similarly, in frame $H = 1$ we make the fake detection boxes appear in the middle region, not associated with the existing track boxes, and make the tracker management module create new track boxes, while the track boxes in frame 0 do not disappear. These frames are not associated with existing frames, making the tracker management module create new trackers, and the trackers initialized in frame 0 are not deleted, allowing to inject $|\mathcal{D}|_{max}$ trackers into the track_pool. Meanwhile, we make the detection boxes of the corresponding region reappear before the R frames to keep the trackers. Since the tracker management module only initializes the detection boxes that detected H frames consecutively, our attack strategy can be formulated: divide the image into K regions, each of which needs to be selected for H frames consecutively in the initialization stage, and then be selected once more before R frames. We use the greedy algorithm with the maximum value of K: $R-H+1$ (when $R-K=H-1$, no region can be added).

Attack Strategy Generating. Algorithm 1 outlines the full process of Tracking-by-detection. We analyze the representative tracker management module, which is also used in some joint-tracking algorithms and propose the attack strategy generation algorithm shown in Algorithm 2. Specifically, we use a greedy algorithm to generate attack strategy to continuously inject different regions of detection boxes into the tracker management module. Such an attack strategy can also be generalized to different joint-tracking algorithm.

Algorithm 2: Generate Attack Strategy

Input: A video sequence $\mathbb{V} = [v_0, v_1, ..., v_{K-1}]$; reserved age R; hit count H
Output: attack strategy $\mathcal{S}$

```
/* region_idx is selected image region */
Initialization: S ← ∅; region_idx = 0; n = 0
while n < K do
    /* Re is the next time each region
       needs to be reactivated */
    if n == 0 then
        S ← S ∪ {region_idx}
        Re_region_idx = n + R + 1
        region_idx = region_idx + 1
    end
    else
        Re_min, idx ← minimum value and index in Re
        if Re_min − n<H then
            S ← S ∪ {idx}
            Re_idx = n + R + 1
        end
        else
            for i = 1 to H do
                S ← S ∪ {region_idx}
            end
            Re_region_idx = n + R + 1
            region_idx = region_idx + 1
            n = n + H − 1
        end
    end
    n = n + 1
end
Return: S
```

Loss Function Design

Score loss. To boost the number of selected boxes, it is necessary to raise the number of prediction candidates that bypass the detection filter, which selects candidates based on their confidence scores. Thus, to increase confidence score of selected candidates $\mathcal{C}_{sel}$, we propose a novel score loss:

$$\mathcal{L}_{score} = \frac{1}{|\mathcal{C}_{sel}|} \sum_{c \in \mathcal{C}_{sel}} \max((T_{conf} - c_{conf}), \lambda)$$

where T_{conf} represents the filtering confidence threshold set by the detection model, c_{conf} represents the confidence scores of the object detector, and λ is a hyper-parameter.

Bounding box area loss. To make more candidates to be reserved in the NMS employed by some detection filters, we need to compress the dimensions of the boxes to reduce the IOU between the candidates. This is expressed as:

$$\mathcal{L}_{area} = \frac{1}{|\mathcal{C}_{sel}|} \sum_{c \in \mathcal{C}_{sel}} (\frac{b_c^w \cdot b_c^h}{S_W \times S_H})^2$$

where the bounding box is b, with b^w and b^h being its width and height. S_W and S_H are the width and height of the input image. This loss is added only when the filter contains NMS.

Feature matching loss. To successfully match the selected detection box with corresponding lost tracker so that the lost tracker is re-found before being deleted, the feature distance for the data association module needs minimizing.

$$\mathcal{L}_{match} = \Psi(\mathcal{T}_i, F'(\mathcal{C}_{sel}))$$

where Ψ is the feature distance function in data association, $\mathcal{T}_i$ represents i-th set of trackers, and F' represents detection filters without score threshold filtering.

Pairwise computation for obtaining feature distances could be expensive if feature extraction is complex or if the number of detection boxes is too large. Therefore, we propose a less computationally intensive method to make the corresponding trackers and detection frames match successfully. The images that need to appear with the same batch of detection boxes use the same perturbation. However, this makes the perturbation accumulation more obvious and the result of the attack decreases. Therefore, for balance, we use the universal perturbation method in the attack initialization stage and feature matching loss after that. Finally, the adversarial loss is represented by:

$$\mathcal{L}_{adv} = \lambda_1 \mathcal{L}_{score} + \lambda_2 \mathcal{L}_{area} + \lambda_3 \mathcal{L}_{match} \quad (2)$$

Similar to existing works (Carlini and Wagner 2017), to make the perturbation invisible, we constrain the $\mathcal{L}_2$ norm:

$$\min_{x^*} \mathcal{L}_{adv} + \mu \|x^* - x\|_2$$

where μ is the hyper-parameter and x is the original image.

Experiments

Experimental Setup

Datasets and models. We use the MOT17DET

Model	Dataset	Hardware	PS (Shapira et al. 2023)				Overload (Chen et al. 2023)				SlowTrack			
			R-Track	R-Lat	#Track	$\mathcal{L}_2$	R-Track	R-Lat	#Track	$\mathcal{L}_2$	R-Track	R-Lat	#Track	$\mathcal{L}_2$
SORT (Y5)	BDD	Titan V	73.1	6.2	69.1	0.042	156.1	12.1	94.6	0.011	**247.0**	**17.5**	**141.8**	**0.010**
		2080 Ti	64.9	11.7			205.1	34.7			**310.3**	**49.4**		
		3090	76.4	6.2			186.3	13.8			**338.3**	**23.0**		
	MOT17	Titan V	37.5	3.8	32.4	0.041	93.0	8.7	47.0	**0.013**	**208.4**	**18.5**	**73.5**	**0.013**
		2080 Ti	33.2	8.6			82.6	19.3			**225.5**	**50.9**		
		3090	39.8	3.7			111.1	9.8			**283.9**	**25.7**		
FairMOT	BDD	Titan V	517.7	10.4	401.1	0.036	1505.9	29.1	939.9	0.029	**2647.6**	**51.3**	**1334.9**	**0.028**
		2080 Ti	390.1	8.7			1258.7	26.7			**2260.7**	**48.1**		
		3090	236.0	3.8			836.4	13.3			**1572.9**	**25.2**		
	MOT17	Titan V	67.4	8.6	48.7	0.036	181.7	21.5	106.2	**0.026**	**341.0**	**41.5**	**159.0**	**0.026**
		2080 Ti	49.3	7.1			149.0	19.9			**279.5**	**38.4**		
		3090	32.4	3.2			108.5	10.1			**220.7**	**21.3**		
ByteTrack	BDD	Titan V	40.2	2.5	79.2	0.032	173.0	9.2	224.7	0.027	**290.0**	**14.7**	**307.0**	**0.022**
		2080 Ti	39.5	2.3			184.0	9.0			**266.4**	**12.5**		
		3090	57.8	3.0			217.7	10.1			**341.4**	**15.1**		
	MOT17	Titan V	29.0	1.9	38.0	0.030	95.0	5.4	78.3	0.025	**173.0**	**9.8**	**130.1**	**0.022**
		2080 Ti	26.9	1.8			97.7	5.4			**168.7**	**9.5**		
		3090	45.2	2.4			115.2	5.9			**204.0**	**10.5**		
BoT-SORT	BDD	Titan V	53.2	19.4	70.0	0.033	79.0	28.9	221.9	0.029	**92.1**	**33.6**	**280.8**	**0.025**
		2080 Ti	57.6	19.6			89.1	30.6			**103.6**	**35.5**		
		3090	61.0	22.1			93.9	34.1			**135.3**	**49.7**		
	MOT17	Titan V	31.7	14.0	40.7	0.034	42.1	18.5	83.9	**0.025**	**52.2**	**23.0**	**127.6**	**0.025**
		2080 Ti	34.2	14.6			45.6	19.5			**59.1**	**25.3**		
		3090	44.0	19.9			47.2	21.1			**69.4**	**30.8**		

Table 1: Effectiveness results of tracking-stage and whole perception latency with number of trackers and average $\mathcal{L}_2$ norm in different models and hardware. Bold denotes the best results (i.e., highest R-Track, R-Lat, #Track, and lowest $\mathcal{L}_2$) in each row.

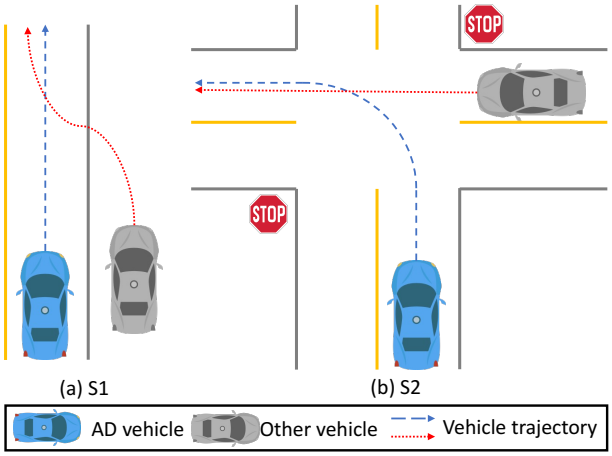


Figure 2: Two scenarios (S1 and S2) for our simulation evaluation setup on the system-level effects.

Model	Attack	Titan V	2080 Ti	3090
SORT (Y5)	PS	211	199	160
	Overload	406	411	337
	SlowTrack	**847**	**1082**	**1018**
FairMOT	PS	417	359	259
	Overload	970	914	685
	SlowTrack	**1848**	**1731**	**1726**
ByteTrack	PS	174	188	166
	Overload	379	427	330
	SlowTrack	**584**	**621**	**555**
BoT-SORT	PS	2395	2460	2372
	Overload	3093	3195	2485
	SlowTrack	**3768**	**4101**	**3245**

Table 2: Latency Time (ms) on MOT 17 dataset.

(MOT17) (Milan et al. 2016) and BDD (Seita 2018) datasets to evaluate our attack SlowTrack performance. MOT17 includes 14 videos with more than 10K images and BDD contains 100K images with various attributes such as weather, scene, and time of day, resulting in a diverse dataset. For MOT17, we use all the data while for BDD, we randomly select 10 videos. As for the models, we select the most representative perception models: SORT (Y5) (Bewley et al. 2016) (a Kalman filtering-based MOT generally used in AD (Apollo 2023; Shen et al. 2022), with YOLO v5 (Jocher 2020) as detector), FairMOT (Zhang et al. 2021), ByteTrack (Zhang et al. 2022), and BoT-SORT (Aharon, Orfaig, and Bobrovsky 2022). We use the default parameters and settings for each model in the evaluation.

	w/o $\mathcal{L}_{score}$	w/o $\mathcal{L}_{area}$	w/o $\mathcal{L}_{match}$	SlowTrack
R-Track	0.0	153.2	171.4	225.5
R-Lat	0.0	34.0	38.3	50.9
#Track	0.1	63.9	66.9	73.5

Table 3: Ablation study for loss designs ($\mathcal{L}_{score}$, $\mathcal{L}_{area}$, and $\mathcal{L}_{match}$ in Eq. (2)) on R-Track, R-Lat, and #Track with SORT (Y5) and MOT17 using 2080 Ti. w/o: without

Model	S1			S2		
	PS	Overload	SlowTrack	PS	Overload	SlowTrack
SORT(Y5)	10%	30%	**100%**	20%	40%	**90%**
FairMOT	20%	40%	**100%**	20%	40%	**100%**
ByteTrack	0%	40%	**80%**	0%	40%	**90%**
BoT-SORT	40%	50%	**100%**	50%	50%	**100%**

Table 4: System-level evaluation (vehicle crash rate) with Baidu Apollo and LGSVL simulator. 10 runs for each cell.

Evaluation metrics. We design the metrics as follows:

$$\text{R-Track} = \frac{\text{Track-Lat}(x^*) - \text{Track-Lat}(x)}{\text{Track-Lat}(x)}$$
$$\text{R-Lat} = \frac{\text{Total-Lat}(x^*) - \text{Total-Lat}(x)}{\text{Total-Lat}(x)}$$
$$\text{\#Track} = \frac{\text{Tracker\#}(x^*) - \text{Tracker\#}(x)}{\text{Tracker\#}(x)}$$

where R-Track and R-Lat represent the rate of increase for the tracking latency and whole perception latency and #Track represents the rate of increase for the number of tracker. To measure the imperceptibility, we use average $\mathcal{L}_2$ norm (Carlini and Wagner 2017; Zhang et al. 2020). We define system-level effects metric as vehicle crash rate: $\frac{N_{\text{crash}}}{N_{\text{total}}}$, where N_{crash} denotes the number of runs causing vehicle crashes and N_{total} is the number of total runs.

Testing hardware. Given that latency is intrinsically tied to the hardware device, we test SlowTrack on multiple hardware: TiTAN V, GeForce RTX 2080 Ti (shown to be used in real AD (Shen et al. 2022)), and GeForce RTX 3090.

Baselines comparison. To our best knowledge, we are the first to propose a latency attack against whole camera-based AD perception pipeline, while the existing attacks focus on object detection alone. Thus, we select two representative latency attacks on object detection as our baselines: PS (Shapira et al. 2023) and Overload (Chen et al. 2023).

Simulation evaluation. To study the system-level effects, we perform an end-to-end attack evaluation on Baidu Apollo (Apollo 2023), an industry-grade full-stack AD system, with LGSVL simulator (Rong et al. 2020), a production-grade AD simulator. Our experiments are conducted on the Borregas Ave map and the Lincoln2017MKZ AD vehicle with default configuration. To simulate our attack impact, we model the latency of the camera-based AD perception and inject it into the AD system. Due to the representativeness of SORT (Y5), we use its latency results tested on 2080 Ti GPU (used in real AD vehicles (Shen et al. 2022)), as our latency modeling results. Our evaluation focuses on two representative scenarios as shown in Fig. 2, where the blue vehicle is the victim AD vehicle and the blue and red lines are the trajectories of the two vehicles. S1 (Fig. 2 (a)) is a common driving scenario for other vehicles to change the lane line and S2 (Fig. 2 (b)) is another common driving scenario for the STOP sign-controlled intersection. We perform 10 runs on each scenario and compare SlowTrack with the two baselines.

Experimental Results

Effectiveness. As shown in Table 1, we compare our SlowTrack with the baselines. SlowTrack can increase the number of tracker up to 1334.9 on FairMOT, which is much better than existing works: at most 939.9 on FairMOT. Especially, for the tracking stage, SlowTrack provides 453.8 times slowing on average compared to the existing works which only have 256.4 times for Overload and 89.1 times for PS. As for the latency of the whole camera-based AD perception, we find that SlowTrack can provide 28.4 times slowing down on average, which is 2.9 times more than the two existing works. Especially, for the practical SORT (Y5) on 2080 Ti, we observe 50.9 times slowing down, while existing works can only have 19.3 times at most. For the imperceptibility, our average $\mathcal{L}_2$ norm (around 0.021 on average) is very small compared to the baselines and existing adversarial attacks on integrity (Zhang et al. 2020).

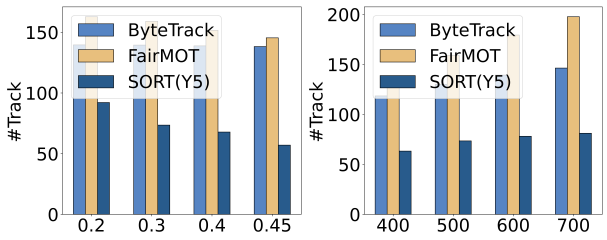
We measure the latency time in Table 2. For instance, while existing representative attack Overload (Chen et al. 2023) can trigger 411 ms latency on SORT (Y5) and MOT17 dataset with 2080 Ti, SlowTrack can provide 1,082 ms, which is 1.6 times more than Overload. Thus, SlowTrack can significantly outperform the existing baselines on camera-based AD perception, with similar levels of imperceptibility.

Ablation study. The ablation study evaluating the designs of Eq. (2) is presented in Table 3. In this study, each loss component is sequentially removed, and the attack is tested on the most practical SORT (Y5) tracking method and the MOT17 dataset, utilizing a 2080 Ti. The findings in Table 3 highlight the indispensability of all three loss designs in achieving high attack effectiveness. Notably, the $\mathcal{L}_{score}$ performs as the most pivotal element in enhancing attack effects. In the absence of $\mathcal{L}_{area}$ and $\mathcal{L}_{match}$, the R-Lat experiences reductions of approximately 33% and 25%, respectively. Thereby, these results indicate the importance of integrating all three loss designs together.

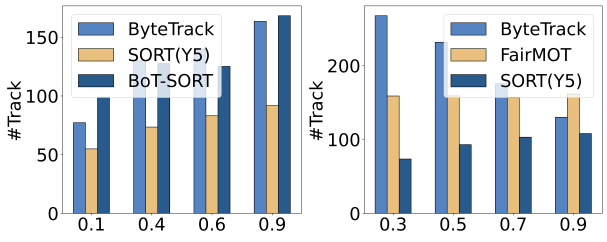
Generality to different thresholds. The effectiveness of SlowTrack is most impacted by different thresholds: confidence score threshold in object detection, maximum number of detection boxes, IOU threshold for NMS, and IOU threshold for data association. To evaluate their impact, we vary the thresholds with different values and measure the #Track on the MOT17 dataset with 2080 Ti. The results are shown in Fig. 3. SlowTrack can generally generate stable results across different threshold parameters, which indicates that setting better thresholds cannot fully defend against SlowTrack. For instance, setting the confidence score threshold to 0.2, the #Track of SORT (Y5) is around 92.1 while setting the confidence score threshold to 0.4, the #Track is around

Model	Titan V		2080 Ti		3090	
	R-Track	R-Lat	R-Track	R-Lat	R-Track	R-Lat
SORT(Y5)	0.032	0.001	0.106	0.007	0.459	0.007
FairMOT	0.402	0.004	0.963	0.010	0.835	0.005
ByteTrack	0.156	0.002	0.075	0.003	0.050	0.001
BoT-SORT	0.006	0.001	0.158	0.032	0.192	0.054

Table 5: Variances of R-Track and R-Lat in repeated experiments (5 times) with different hardware on MOT17 dataset.



(a) Confidence Threshold (b) Max. Number of Detections



(c) IOU Threshold for NMS (d) IOU Threshold for Match

Figure 3: Attack effectiveness under different thresholds.

67.8. Additionally, finetuning the thresholds usually will significantly reduce the benign performance. Thus, the results show the generality of SlowTrack to different thresholds.

End-to-end simulation evaluation results. As shown in Table 4, SlowTrack can achieve 95% vehicle crash rate on average while for other attacks, they can only have around 30%. Note that the vehicle crash rate in benign cases is always 0%. The results demonstrate that the existing latency attacks on object detection alone cannot trigger sufficient latency to highly effectively cause vehicle crashes, which motivates our new attack design on entire perception. SlowTrack significantly improves the system-level effects.

Running time variance. Recognizing the potential variability of system latency due to many factors, we repeat our experiments 5 times (Rybel 2017) and measure the variance of R-Track and R-Lat. As shown in Table 5, the variance is negligible compared to the original values (Liu et al. 2023).

Discussion

Physical-World Attack Realizability

Our study investigates the runtime robustness of camera-based AD perception by adversarial attacks. Such attack is physically realizable as demonstrated in AttrackZone attack (Muller et al. 2022), which leverages the projector to project the noise-level adversarial attack in physical world at night. To improve our attack realizability, we take a small step forward to generate a patch-based adversarial attack which is generally demonstrated as a physical-world realizable attack (Wang et al. 2023, 2022).

Adversarial patch generation. To formulate the patch into a patch δ, we design the following method:

$$\min_{\delta} E_{x\sim\mathcal{X}}\mathcal{L}_{adv}(x+\delta)$$

where the $\mathcal{L}_{adv}$ is introduced in Eq. (2), $\mathcal{X}$ denotes EoT (Athalye et al. 2018) distribution for robustness such as different pre-processing method.

Evaluation setup and preliminary results. We use a similar setup in the experiment section, where we select the most practical model: SORT (Y5) and MOT17 dataset with 2080 Ti. The results values for R-Lat, R-Track, and #Track are 82.3, 336.1, and 80.9, respectively, aligning closely with the findings in Table 1. These results suggest that the patch attack can potentially induce substantial system-level effects with practicality. Note that the patch results are slightly better than the noise attack. Since the perturbation strength of the patch is much larger, it enables the attacker to generate bounding boxes with high confidence and to precisely dictate their locations. Thus, the tracking can be easily controlled by the attacker and the attack effects can be improved. However, the patch attack requires more complex designs such as patch size and patch location, which are correlated with the practicality and effectiveness. In this paper, we only provide a preliminary evaluation demonstrating the potential to transfer the noise attack to the patch attack. We leave the study on patch-based attack as our future work.

Limitations and Future Work

First, although we have some physical-world realizability improvements and some existing works demonstrate that such attack is realizable, it is still unclear whether SlowTrack can indeed work well in physical world, which can be a potential future direction to explore. Second, exploring SlowTrack under a black-box threat model, a more practical one, is another potential future work. Third, while several availability-based latency attacks in AD have been identified, the exploration of defense directions in this context remains limited. Consequently, we consider the investigation of defenses as a part of our future work.

Conclusion

This paper presents a first study on availability-based latency adversarial attacks considering the entire camera-based AD perception pipeline, i.e., both object detection and object tracking. We design a novel attack framework, SlowTrack, with a two-stage attack strategy and three novel loss functions. Our results show that SlowTrack can outperform all the existing latency attacks on camera-based object detection and significantly improve system-level effects, i.e., 95% vehicle crash rate. Due to the critical role of perception, we hope that our findings and insights can inspire more future research into this largely overlooked research perspective.

Acknowledgments

We would like to thank Ziwen Wan, Yunpeng Luo, Tong Wu, Xinyang Zhang, and the anonymous reviewers for their valuable and insightful feedback. This research was supported in part by National Key R&D Program of China (2020AAA0107702); the NSF under grants CNS-1929771, CNS-2145493, and CNS-1932464; National Natural Science Foundation of China (U21B2018, 62161160337, 62132011, 62376210, 62006181, U20B2049); Shaanxi Province Key Industry Innovation Program (2021ZDLGY01-02); and Fundamental Research Funds for the Central Universities under grant (xtr052023004, xtr022019002). Chao Shen and Qi Alfred Chen are the corresponding authors.

References

Aharon, N.; Orfaig, R.; and Bobrovsky, B.-Z. 2022. BoT-SORT: Robust Associations Multi-Pedestrian Tracking. *arXiv preprint arXiv:2206.14651*.

Apollo. 2023. Baidu Apollo team, Apollo: Open Source Autonomous Driving. https://github.com/ApolloAuto/apollo.

Athalye, A.; Engstrom, L.; Ilyas, A.; and Kwok, K. 2018. Synthesizing Robust Adversarial Examples. In *International conference on machine learning*, 284–293. PMLR.

Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; and Upcroft, B. 2016. Simple Online and Realtime Tracking. In *2016 IEEE international conference on image processing (ICIP)*, 3464–3468. IEEE.

Cao, Y.; Wang, N.; Xiao, C.; Yang, D.; Fang, J.; Yang, R.; Chen, Q. A.; Liu, M.; and Li, B. 2020. 3d adversarial object against msf-based perception in autonomous driving. In *Proceedings of the 3rd Conference on Machine Learning and Systems*.

Cao, Y.; Wang, N.; Xiao, C.; Yang, D.; Fang, J.; Yang, R.; Chen, Q. A.; Liu, M.; and Li, B. 2021. Invisible for both Camera and LiDAR: Security of Multi-Sensor Fusion based Perception in Autonomous Driving Under Physical World Attacks. In *Proceedings of the 42nd IEEE Symposium on Security and Privacy (IEEE S&P 2021)*.

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, E.-C.; Chen, P.-Y.; Chung, I.; Lee, C.-r.; et al. 2023. Overload: Latency Attacks on Object Detection for Edge Devices. *arXiv preprint arXiv:2304.05370*.

DiPalma, C.; Wang, N.; Sato, T.; and Chen, Q. A. 2021. Security of camera-based perception for autonomous driving under adversarial attack. In *2021 IEEE Security and Privacy Workshops (SPW)*, 243–243. IEEE.

Haque, M.; Chauhan, A.; Liu, C.; and Yang, W. 2020. ILFO: Adversarial attack on adaptive neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14264–14273.

Hong, S.; Kaya, Y.; Modoranu, I.-V.; and Dumitraş, T. 2020. A panda? no, it's a sloth: Slowdown attacks on adaptive multi-exit neural network inference. *arXiv preprint arXiv:2010.02432*.

Jia, Y. J.; Lu, Y.; Shen, J.; Chen, Q. A.; Chen, H.; Zhong, Z.; and Wei, T. W. 2020. Fooling Detection Alone is Not Enough: Adversarial Attack against Multiple Object Tracking. In *International Conference on Learning Representations (ICLR'20)*.

Jocher, G. 2020. ultralytics/yolov5. https://github.com/ultralytics/yolov5.

Kane, M. 2021. Tesla Sold 2 Million Electric Cars: First Automaker To Reach Milestone. *insideevs.com*.

Kato, S.; Tokunaga, S.; Maruyama, Y.; Maeda, S.; Hirabayashi, M.; Kitsukawa, Y.; Monrroy, A.; Ando, T.; Fujii, Y.; and Azumi, T. 2018. Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, 287–296. IEEE.

Liu, H.; Wu, Y.; Yu, Z.; Vorobeychik, Y.; and Zhang, N. 2023. SlowLiDAR: Increasing the Latency of LiDAR-Based Detection Using Adversarial Examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5146–5155.

Luo, Y.; Wang, N.; Yu, B.; Liu, S.; and Chen, Q. A. 2022. Infrastructure-Aided Defense for Autonomous Driving Systems: Opportunities and Challenges. In *NDSS Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*.

Ma, C.; Wang, N.; Chen, Q. A.; and Shen, C. 2023a. SlowTrack: Increasing the Latency of Camera-based Perception in Autonomous Driving Using Adversarial Examples. arXiv:2312.09520.

Ma, C.; Wang, N.; Chen, Q. A.; and Shen, C. 2023b. WIP: Towards the Practicality of the Adversarial Attack on Object Tracking in Autonomous Driving. In *ISOC Symposium on Vehicle Security and Privacy (VehicleSec)*.

Milan, A.; Leal-Taixé, L.; Reid, I.; Roth, S.; and Schindler, K. 2016. MOT16: A Benchmark for Multi-Object Tracking. *arXiv preprint arXiv:1603.00831*.

Muller, R.; Man, Y.; Celik, Z. B.; Li, M.; and Gerdes, R. 2022. Physical Hijacking Attacks against Object Trackers. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2309–2322.

Rong, G.; Shin, B. H.; Tabatabaee, H.; Lu, Q.; Lemke, S.; Možeiko, M.; Boise, E.; Uhm, G.; Gerow, M.; Mehta, S.; et al. 2020. LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving. In *2020 IEEE 23rd International conference on intelligent transportation systems (ITSC)*, 1–6. IEEE.

Rybel, N. D. 2017. How to obtain standard deviation of replicate experiments with repeated measurements?

Sato, T.; Shen, J.; Wang, N.; Jia, Y.; Lin, X.; and Chen, Q. A. 2021a. Dirty Road Can Attack: Security of Deep Learning based Automated Lane Centering under Physical-World Attack. In *30th USENIX Security Symposium (USENIX Security 21)*, 3309–3326.

Sato, T.; Shen, J.; Wang, N.; Jia, Y. J.; Lin, X.; and Chen, Q. A. 2020a. Hold tight and never let go: Security of deep learning based automated lane centering under physical-world attack. *arXiv preprint arXiv:2009.06701*.

Sato, T.; Shen, J.; Wang, N.; Jia, Y. J.; Lin, X.; and Chen, Q. A. 2020b. Security of deep learning based lane keeping system under physical-world adversarial attack. *arXiv preprint arXiv:2003.01782*.

Sato, T.; Shen, J.; Wang, N.; Jia, Y. J.; Lin, X.; and Chen, Q. A. 2021b. WIP: Deployability improvement, stealthiness user study, and safety impact assessment on real vehicle for dirty road patch attack. In *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, volume 2021, 25.

Sato, T.; Yue, J.; Chen, N.; Wang, N.; and Chen, Q. A. 2023. Intriguing Properties of Diffusion Models: A Large-Scale Dataset for Evaluating Natural Attack Capability in Text-to-Image Generative Models. *arXiv preprint arXiv:2308.15692*.

Seita, D. 2018. BDD100k: A large-scale diverse driving video database. *The Berkeley Artificial Intelligence Research Blog. Version*, 511: 41.

Shapira, A.; Zolfi, A.; Demetrio, L.; Biggio, B.; and Shabtai, A. 2023. Phantom Sponges: Exploiting Non-Maximum Suppression to Attack Deep Object Detectors. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 4571–4580.

Shen, J.; Wang, N.; Wan, Z.; Luo, Y.; Sato, T.; Hu, Z.; Zhang, X.; Guo, S.; Zhong, Z.; Li, K.; et al. 2022. SoK: On the Semantic AI Security in Autonomous Driving. *arXiv preprint arXiv:2203.05314*.

Shumailov, I.; Zhao, Y.; Bates, D.; Papernot, N.; Mullins, R.; and Anderson, R. 2021. Sponge examples: Energy-latency attacks on neural networks. In *2021 IEEE European symposium on security and privacy (EuroS&P)*, 212–231. IEEE.

Tesla. 2022. Future of Driving. *tesla.com*.

Wan, Z.; Shen, J.; Chuang, J.; Xia, X.; Garcia, J.; Ma, J.; and Chen, Q. A. 2022. Too Afraid to Drive: Systematic Discovery of Semantic DoS Vulnerability in Autonomous Driving Planning under Physical-World Attacks. In *Network and Distributed System Security (NDSS) Symposium, 2022*.

Wang, D.; Li, C.; Wen, S.; Han, Q.-L.; Nepal, S.; Zhang, X.; and Xiang, Y. 2021. Daedalus: Breaking Nonmaximum Suppression in Object Detection via Adversarial Examples. *IEEE Transactions on Cybernetics*, 52(8): 7427–7440.

Wang, N.; Luo, Y.; Sato, T.; Xu, K.; and Chen, Q. A. 2022. Poster: On the System-Level Effectiveness of Physical Object-Hiding Adversarial Attack in Autonomous Driving. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 3479–3481.

Wang, N.; Luo, Y.; Sato, T.; Xu, K.; and Chen, Q. A. 2023. Does Physical Adversarial Example Really Matter to Autonomous Driving? Towards System-Level Effect of Adversarial Object Evasion Attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 4412–4423.

Zhang, X.; Wang, N.; Shen, H.; Ji, S.; Luo, X.; and Wang, T. 2020. Interpretable Deep Learning under Fire. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*.

Zhang, Y.; Sun, P.; Jiang, Y.; Yu, D.; Weng, F.; Yuan, Z.; Luo, P.; Liu, W.; and Wang, X. 2022. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. In *European Conference on Computer Vision*, 1–21. Springer.

Zhang, Y.; Wang, C.; Wang, X.; Zeng, W.; and Liu, W. 2021. FairMOT: On the Fairness of Detection and Re-Identification in Multiple Object Tracking. *International Journal of Computer Vision*, 129: 3069–3087.