Physical Hijacking Attacks against Object Trackers

Raymond Muller
Purdue University
mullerr@purdue.edu

Yanmao Man
University of Arizona
yman@arizona.edu

Z. Berkay Celik
Purdue University
zcelik@purdue.edu

Ming Li
University of Arizona
lim@arizona.edu

Ryan Gerdes
Virginia Tech
rgerdes@vt.edu

ABSTRACT

Modern autonomous systems rely on both object detection and object tracking in their visual perception pipelines. Although many recent works have attacked the object detection component of autonomous vehicles, these attacks do not work on full pipelines that integrate object tracking to enhance the object detector's accuracy. Meanwhile, existing attacks against object tracking either lack real-world applicability or do not work against a powerful class of object trackers, Siamese trackers. In this paper, we present AttrackZone, a new physically-realizable tracker hijacking attack against Siamese trackers that systematically determines valid regions in an environment that can be used for physical perturbations. AttrackZone exploits the heatmap generation process of Siamese Region Proposal Networks in order to take control of an object's bounding box, resulting in physical consequences including vehicle collisions and masked intrusion of pedestrians into unauthorized areas. Evaluations in both the digital and physical domain show that AttrackZone achieves its attack goals 92% of the time, requiring only 0.3-3 seconds on average.

CCS CONCEPTS

- **Computing methodologies** → **Machine learning algorithms**;
- **Security and privacy** → *Systems security*.

KEYWORDS

adversarial machine learning; neural networks; object tracking; autonomous driving; video surveillance

ACM Reference Format:
Raymond Muller, Yanmao Man, Z. Berkay Celik, Ming Li, and Ryan Gerdes. 2022. Physical Hijacking Attacks against Object Trackers. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), Nov. 7–11, 2022, Los Angeles, CA, USA*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3548606.3559390

1 INTRODUCTION

Object detection and object tracking are both an important part of the pipelines of different autonomous systems in various domains, such as autonomous driving [27], pedestrian detection [14], and mobile robot navigation [2]. Extensive work has been studied in attacking the object detection pipeline of autonomous vehicles, such as using stickers and patches attached to objects within a scene to fool object detection into ignoring or misclassifying critical entities (e.g., stop signs [11, 54]). However, object detection is only half of the visual perception pipeline in autonomous systems; object *tracking* is used to build the trajectories of objects in order to enhance the accuracy of object detection.

Unlike object detection, which involves determining the classification and bounding boxes of objects in a *single* frame, object tracking involves computing the bounding box of the *same* object over multiple frames. By keeping a record of the same object's movement over time, object trackers are able to estimate velocities and trajectories for objects, which allows for a more consistent and accurate bounding box calculation compared to pure object detection [22]. This accuracy enhancement also provides robustness against object detection attacks. With a tracker deployed, attacks purely targeting object detection must succeed at least 98% of the time over 60 consecutive frames, which is infeasible for current attacks against object detection [22]. There are many different kinds of trackers, but Siamese tracking remains the most prevalent due to their inherent balance between accuracy and efficiency, which makes them suitable for real-time applications [37].

Because object tracking does not involve classifying objects, and instead focuses on calculating their bounding boxes, attacks against it focus on *tracker hijacking*, the technique of deviating bounding boxes calculated by an object tracker in a controlled direction. There are two types of tracker hijacking attacks: move-in and move-out. In the move-in attack, the bounding box of an object (e.g., vehicle, pedestrian, tumbleweed, animal) not already in the target system's path is moved into the path to disrupt its operation. For example, if the target system is a vehicle, it will stop or deviate from its path to avoid the hijacked object. If the target system is a security camera equipped with pedestrian tracking, it will sound a false alarm. Similarly, a move-out attack takes the bounding box of an object in the target system's path and moves it outside of the path. For a vehicular target system, this will cause a collision as the system determines an obstacle has been cleared from the path. For surveillance, a move-out attack can mask the entry of an intruder.

Because object detection attacks are not feasible against object tracking, tracker hijacking attacks take advantage of elements unique to the tracking domain that do not exist in object detection. For example, one previous work focused on attacking Simple Online Real-Time (SORT) trackers [22], which use a straightforward velocity estimation model to track objects. Unfortunately, this attack exploited a design flaw specific to the SORT algorithm, where tracking results are given a lifespan such that incorrect tracking results

CCS '22, November 7–11, 2022, Los Angeles, CA, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9450-5/22/11.
https://doi.org/10.1145/3548606.3559390

from a broken tracker continue to be reported for a given number of frames. The SORT attacks are not transferable because this design flaw does not exist in more complex tracking algorithms, such as Siamese trackers. Another work has targeted Siamese tracking [21], utilizing adversarial machine learning to create noise patterns that alter tracking results. However, this attack only works in the digital domain, as it cannot distinguish between areas that are impossible to attack in the physical domain, such as the sun, and valid attack areas (e.g., walls). This severely limits its applicability, requiring an attacker to directly modify the camera output.

In this paper, we introduce AttrackZone, the first work which can physically launch tracker hijacking attacks against Siamese trackers. Because the physical world dynamically and unpredictably changes, when conducting the attack, it is necessary to ensure physically perturbable space exists beyond the potentially small constraints of the targeted object. Therefore, we leverage 3D point cloud data to construct *attack zones*, areas in an environment that can be used in a physical domain tracker-hijacking attack. Once these areas are determined, we exploit the Siamese heatmap proposal algorithm and generate noise patterns to shift the tracker's focus to desired areas in the scene. We ensure that our attack is minimally noticeable while evading established defenses (such as Kalman Filtering) and remaining adaptive to the environment. We then project the noise patterns onto the valid attack zones to achieve tracker hijacking without compromising a victim vehicle's cameras.

We evaluate the efficacy of AttrackZone against base [26] and Distractor-aware [56] Siamese trackers in autonomous driving and video surveillance. We demonstrate the attack success rate against the trackers' original test datasets, simulated data in CARLA, and trackers deployed in real-world environments. We take a model-agnostic approach and examine both the attack inter-transferability between different models of Siamese trackers and the attack intra-transferability between the same models with different parameters and training data. We inspect and compare two different approaches, offline and online, that are used depending on whether or not the victim route is known. We also examine the attack parameters, evaluating the time and projectable space needed for a successful attack and how strong the projections must be.

We measure the raw success rate based on whether the original goal of move-in or move-out is achieved for a given attack. Across all evaluated Siamese tracking models, on the trackers' original test datasets, the attack success rate is on average 95.5%, while the average success rate is 91.1% on the simulated dataset. For real world attacks, we achieve an average success rate between 80.8% and 89% depending on whether the attack is conducted online or offline. Attacks take on average between 0.3 and 3 seconds to achieve their goals, and only 63%-82% of the environment must be perturbable for the attack to succeed with a 30% margin for error. Lastly, our attack modifies each pixel on average by less than 5.86%. AttrackZone runs efficiently in real-time and evades current defenses against tracker hijacking attacks, such as Kalman Filters and noise cancellation.

Contributions. In this paper, we make the following contributions.

- We are the first to demonstrate a systematic process for finding valid attack zones in projector attacks.
- We present a novel attack in the physical domain against Siamese trackers, exploiting the design of heatmap generation to guide trackers into a specific area.
- We successfully launch attacks against three of the most prevalent Siamese tracking models, evaluating AttrackZone's performance in the real world against the original test datasets for the Siamese trackers and a diverse simulated dataset.
- AttrackZone code is available at https://github.com/purseclab/AttrackZone for public use and validation.

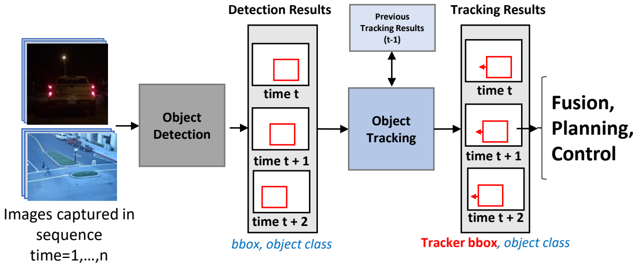

Figure 1: An overview of a full autonomous system pipeline.

2 BACKGROUND

A typical visual perception pipeline for autonomous systems consists of an object detector and an object tracker, as shown in Figure 1. Given a sequence of images from a camera's video feed, the system runs an *object detector* to obtain a bounding box and the object class for each object of interest [40]. These bounding boxes are then fed into an *object tracker*, which first computes its own bounding box based on its historical results then combines its results with the object detector's [5]. This process helps smooth out errors in bounding box predictions by providing *spatiotemporal* feature consistency across frames. Lastly, it passes this information to subsequent modules, such as the planning module of an autonomous vehicle that plans the vehicle's driving paths [52].

2.1 Object Detection and Tracking

To reason about the surrounding environment, an autonomous system uses a real-time object detection algorithm to classify objects in its vicinity. These algorithms typically employ Convolutional Neural Networks (CNNs) to detect objects in real-time. For instance, YOLO [40] is a popular network architecture because of its high speed and performance compared to other object detection architectures such as R-CNN [41] and DPM [48]. YOLO and its variants split an image into a square grid of a pre-configured size. For each square in the grid, they assign probabilities that the square is of each given class. They then congregate the squares into individual bounding boxes and return them.

Object tracking algorithms, on the other hand, estimate an object's heading and velocity to compute object trajectories and correct potential errors in object detection [52]. The trackers predict the object behavior by taking into account the spatiotemporal consistency of the image frames, whereas object detectors independently reason on each input.

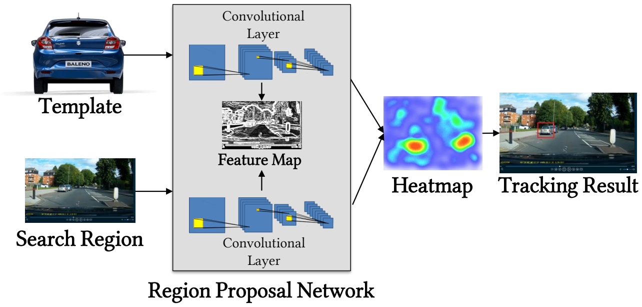

Figure 2: Illustration of Siamese-based object trackers.

While there are many different object tracking algorithms, the Siamese-based trackers have recently gained prevalence due to their efficiency in real-time inference [9, 37, 56], and are commonly employed in real-world systems, e.g., security cameras [53], robotic vehicles [15], and intelligent surveillance [1]. Siamese-based trackers extract image features via a region proposal network to track objects between frames. This enables them to yield higher accuracy than different types of trackers such as the Simple Online Realtime Tracking (SORT) [9], which uses velocity estimations between frames to predict tracker position.

2.1.1 Siamese Networks. A Siamese tracker [43] uses image features to track an object between frames rather than creating a velocity estimation (See Figure 2). The key feature of Siamese tracking is the *region proposal network*. A Siamese network is trained to classify each area in the image as either "background", to be ignored, or "target", to be tracked. A tracking algorithm on each frame then uses a CNN to find the regions that correspond to a target, generating a *heatmap* for the target region. The heatmap restricts the search space of the tracker to only target areas. Specifically, a bounding box regression is applied to locate the correct object, and update its tracker within that search space.

Siamese networks have different variants that differ in how they generate the heatmap. *Base Siamese Networks* (BSNs), such as SiamRPN [26], are trained only on target and background and purely segment an image based on whether or not it should be tracked. A recent improvement on BSNs is Distractor-aware Siamese Networks (DASNs), e.g., the DaSiamRPN [56] and DaSiamRPN+ [21] models, which are trained on data with labeled distractors. By learning to ignore areas of the image likely to cause false positives in BSNs, DASNs are able to generate more concise heatmaps focusing on an object of interest.

3 THREAT MODEL

3.1 Attack Goal: Physical Tracker Hijacking

We consider an adversary that aims to launch a remote *physical tracker hijacking* against Siamese-based object trackers with the goal of *moving-in* and *moving-out* the objects. Tracker hijacking is concerned purely with changing the bounding box of an object, and is not intended to affect an object's class. Figure 3 shows the attack goal and outcomes of tracker hijacking against two systems, object tracking in autonomous vehicles and pedestrian tracking in video surveillance systems.

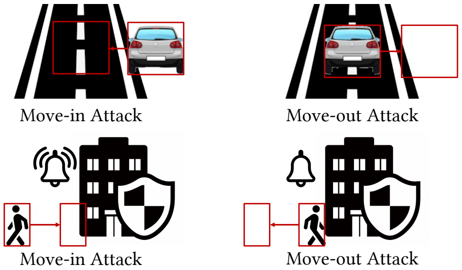

Figure 3: The attacker's goals and outcomes with move-in and move-out attacks against object trackers: (Top) autonomous vehicles (traffic disruption, collision) and (Bottom) video surveillance systems (false alarms, stealthy intrusion).

Move-in Attack. The adversary in this attack aims to move the tracker of an object (e.g., vehicle, pedestrian, tumbleweed, animal) not in the target system's path (e.g., vehicle, drone) into its path. The move-in attack for an autonomous vehicle stops the vehicle prematurely with the goal of disrupting the flow of traffic and leaving the vehicle vulnerable to an ambush. Similarly, for video surveillance, the attack triggers the camera alarm to fool the system into thinking someone has entered the building. This could distract operators or cause multiple false alarms to lull the system into ignoring accurate system warnings.

Move-out Attack. In this attack, the adversary's goal is to move the tracked object out of the tracker's path. Although the object is still there, the tracker determines that it has moved out of its way because its tracker has deviated. For autonomous vehicles, the attacker can cause the target system to collide with an object, which may be a pedestrian that may suffer fatal injuries. The move-out attack for video surveillance prevents a camera alarm system from seeing someone enter a building, causing a stealthy entrance into a typically secure facility.

3.2 Attack Model and Assumptions

We consider an attacker with two capabilities. First, we assume that the attacker knows the location of the system running the object tracker (where the camera or cameras are mounted). With this knowledge, the attacker uses a *surrogate* camera (e.g., stereo camera or camera with LiDAR) to obtain the view of the victim camera and its 3D point cloud for identification of attack zones (the surface of the valid objects, detailed below). An attacker can obtain this data in *real time* (online) or *offline*. An attacker can fly a drone-mounted low-cost/high-accessible camera and approximate the view of the victim's camera in real-time, ensuring a similar, adaptable view angle. As another example, if the route of a target autonomous vehicle is known ahead of time, an attacker can collect the camera feed offline by driving in the same route as the victim vehicle, or use online resources (e.g., USGS Explorer [45] with Google Earth [13] for LiDAR and image data). The performance of offline vs. online attacks is compared in Section 6.3.

Second, we assume the attacker has access to *any* open-source Siamese-based object tracker for generating the physical perturbations based on the surrogate camera feed. We show in Section 6 that attacks constructed on a specific Siamese network (e.g., BSNs) transfer to its variants (e.g., DaSiamRPN and DaSiamRPN+).

To conduct the tracker hijacking in the physical world, the attacker aims to find *stealthy* and *robust* physical perturbations on the surface of valid objects over time. Attacks in practical settings require targeting a moving object and generating perturbations which are (1) within the field of view of the victim's camera, (2) reflected on objects that distract the victim's camera feed (e.g., not on glass surfaces and sky), (3) effective within environmental instability (e.g., evolving lighting conditions), and (4) the least conspicuous to human observers.

Lastly, to launch the tracker hijacking, the attacker remotely projects the perturbations to the attack scene, e.g., by attaching a projector to an adversarial vehicle driving near the victim vehicle, or flying an adversarial drone with a mounted projector.

We do not assume the attacker has physical access to the target vehicle's camera or hardware. Additionally, the adversary does not exploit hardware vulnerabilities (e.g., LiDAR and GPS spoofing/jamming exploits [4]) or software vulnerabilities (e.g., remote code injections [17]).

4 EXISTING ATTACKS AND DESIGN CHALLENGES

Our primary goal and challenge is conducting the tracker hijacking attack in the physical world by altering the physical environment captured by the camera. With the threat model and design goals set, we show the reasons that existing attacks on object detection fail to work against object trackers, and the limitations of existing attacks on object trackers.

4.1 Generalizability of Attacks against Object Detectors to Trackers

Prior works have mainly studied adversarial attacks against object detectors that take a single frame at a time, e.g., printed adversarial stickers attached to a stop sign to fool the object detection results. These attacks focus on creating false objects [20, 31, 54], masking objects [3, 23], and changing a specific object's class [11, 29, 49].

Existing attacks against object detection fail to alter the object tracking results (and thus the actual visual perception pipeline of autonomous systems) for two main reasons. First, attacks against object detection do not consider the effects of adding object tracking to an autonomous system's visual pipeline to ensure spatio-temporal consistency across video frames. By keeping a record of the movement of an object, object trackers are able to achieve more accurate bounding box results than pure object detectors, meaning that bounding boxes computed by an object tracker have fundamentally different behaviors than bounding boxes computed by object detectors [22]. Second, object trackers naturally improve the robustness of object detection by taking consistent and stable results across multiple frames. Specifically, with an object tracker integrated into the autonomous system pipeline, an attacker must fool the object detector 98% of the time throughout 60 consecutive frames to successfully hijack a tracker [22]. This success rate is not feasible for current attacks against object detection [6, 11, 29, 54], making them unable to offer promise against object trackers.

4.2 Attacks against Object Trackers

Existing works that attack object trackers are very limited. A recent work has demonstrated tracker hijacking against SORT trackers by exploiting its reliance on velocity estimation as opposed to actual image features [22]. This makes the attack infeasible against many other trackers (including Siamese trackers) that do not use velocity estimation to track objects. Additionally, the attacks are generated in digital simulation (directly changing pixel values fed into the tracker), and restrict themselves to a patch placed on a single vehicle, limiting its attack surface area and success rate.

With the growing prevalence of Siamese-based trackers, a few attacks against them have been recently proposed. The RTAA attack aims to move the tracker away from its correct position [21], and the Cooling-Shrinking (C-S) attack removes the tracker from existence [50]. Both attacks are digital domain attacks, directly changing the input image pixels without physically launching the attack. They also generate noise without considering the physical constraints, leading to physically infeasible attacks, where imperturbable regions, such as the sky and sun, are perturbed.

Additionally, the consequences of existing tracker attacks on the autonomous system's decision-making pipeline are not well defined. The RTAA attack does not specify a desired end position for the tracker, causing it to move randomly and unrealistically around an image. This greatly reduces the attack's stealthiness and makes it easy to prevent, e.g., by applying a Kalman Filter even without knowledge of the attack. While the goal of the C-S attack is to hide an object from the Siamese tracker, it does not hide it from object detection algorithms, which can still assist the decision-making. This means that there are few physical consequences on the visual perception pipeline in an autonomous system, which simply loses the tracker of an object without moving it in or out.

4.3 Design Challenges and Potential Solutions

(C_1) **Physical Space Constraints.** The need for a physically feasible tracker hijacking attack arises from the challenges of conducting a purely image-based attack. With increasing security measures for autonomous systems, it is logistically easier to perturb the physical world to alter a system's perception instead of compromising the system itself. Yet, converting a 3D space in the physical world to a 2D space for perturbations is not trivial.

One approach to ensure a usable physical perturbation region is to commandeer a part of the environment and apply physical patches to objects. For example, for vehicle tracker hijacking, an attacker can paste a patch with adversarial noise onto the back of a car and drive it in front of the victim, ensuring that the car is always a usable attack region [11, 18]. However, commandeering surfaces sacrifices flexibility that may be crucial to ensuring physical feasibility in a dynamic environment. In the previous example, a vehicle interceding behind the attacker vehicle renders the patch obsolete.

Another approach would be to only display or project noise onto the object targeted for tracker hijacking. However, this severely constrains the amount of area that is usable for a tracker hijacking attack. As we show in Section 6.4.3, having a limited area to display or project noise onto negatively impacts the attack success rate. Thus, there is a need to distill the environmental constraints into

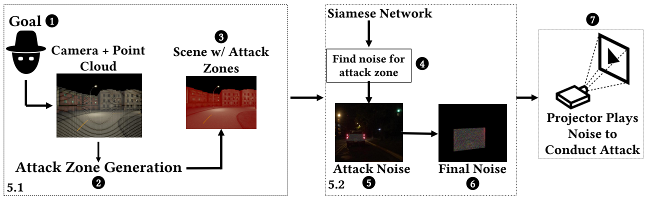

Figure 4: Illustration of AttrackZone's attack stages.

x usable region(s) for adversarial noise, while ensuring that these regions are sufficient to launch a tracker hijacking attack.

(C_2) **Stealthiness.** Perturbations should be crafted to be small, innocuous, or imperceptible to a casual observer. Yet, perturbations must still be visible to the camera and must not be so minute as to lose their effect. Therefore, the attacker must find minimal perturbations to achieve the attack goal.

(C_3) **Spatiotemporal Constraints.** The object trackers are spatiotemporally consistent, with each prediction relying on the previous tracking results. This places a time constraint on the noise generation and attack regions. Unlike in object detection attacks, where each individual image frame can be treated independently, object tracking attacks must successfully influence a group of frames over a given period, necessitating rapid adjustments to the attack to match changes in the surrounding environment. This plays an important role in the success of offline attacks vs. online attacks. Offline attacks start with all of the necessary information to ensure consistency when adapting to a changing environment, while online attacks do not, putting them at a disadvantage.

(C_4) **Evade Kalman Filtering.** A tracker hijacking attack that moves randomly and unrealistically around an image can be defeated by a Kalman Filter, which mitigates the effects of the hijacking. Because Kalman Filters are often used to correct tracking errors, with little additional overhead [7], the attack approach must be able to evade Kalman Filtering to be considered functional.

5 PHYSICAL OBJECT TRACKER HIJACKING

We present AttrackZone, a spatiotemporal attack for tracker hijacking, which identifies the attack zones within a physical scene (spatial) and generates physical perturbations across time (temporal). AttrackZone allows an attacker to launch remote physical attacks to disrupt regular operations of object trackers in autonomous systems. AttrackZone is general to all Siamese-based trackers used in diverse autonomous systems.

Figure 4 presents the stages of AttrackZone. The attacker first defines an attack goal, specifying a desired location for the bounding box of the targeted object to move to (❶). The attacker can either specify the end region or a sliding window over multiple frames to ensure a smooth tracker movement. After defining the attack goal, the attacker supplies the 3D point cloud of the surrounding environment and a camera feed (to provide a surrogate for the victim's view) to obtain an attack zone (❸), the valid regions within a scene used for physical manipulations (Section 5.1). The attack zones systematically determine the usable physical regions within the environment, addressing physical space constraints to project the perturbations and increase the attack success likelihood (C_1).

Once the attack zones are identified, the attacker supplies a trained Siamese network and generates perturbations for the attack zones (❹). The attacker uses the Siamese network for feedback, and minimizes a loss function (Section 5.2) to generate heatmap-altering perturbations until the optimal ones for tracker hijacking are found (❺). The loss function minimization ensures a minimal amount of perturbations that achieve the intended effect, addressing stealthiness (C_3). In addition, each perturbation generated is checked against a Kalman Filter. If the Kalman Filter prevents the tracker from going to the attacker-specified location, we discard the perturbations and generate a new pattern (C_4). If the victim's route is known ahead of time, steps ❹ and ❺ can be conducted in advance with no time constraint, generating perturbations offline to play during an attack, which directly addresses spatiotemporal constraints offline (C_2). Otherwise, the noise can be generated online at a slower but still adequate rate. We discuss the tradeoff between online and offline attacks in Section 6.3. Lastly, the attacker passes the perturbations to a projector, which casts onto the valid attack zones (❻-❼).

5.1 Attack Zone Identification

The first step of the AttrackZone attack is to identify a valid attack zone within the victim camera's view for generating physical adversarial perturbations on real objects.

5.1.1 Point Cloud Remapping. To find valid physical attack zones, we initially collect the 3D point cloud of the scene where the attack is conducted. The point cloud can be collected online or offline through light-based sensors, such as LiDAR and stereo cameras. These sensors yield more accurate point clouds for attack zones than non-light-based sensors (e.g., sonar) because they generate a semantically precise 3D representation of an environment, including the scene shape and its ground surface characteristics. For example, a LiDAR penetrates glass surfaces while sound waves bounce off any solid surface including glass.

As an alternative to light-based sensors, we have also examined the impact of generating the area of attack scene using *semantic segmentation*, such as through Mask R-CNN [16]. Semantic segmentation assigns a class to each pixel in an image, allowing one to separate it into different zones. However, we found that segmentation is susceptible to environmental conditions, e.g., uneven lighting [32], which yields imprecise 3D point clouds.

We determine the valid attack areas relative to the victim's camera (e.g., a camera installed behind the rear-view mirror or on the dashboard in a vehicle) after the 3D point cloud is obtained. Algorithm 1 details the steps for filtering out the 3D points not in the victim camera's Field of View (FoV). It is a worklist-style algorithm that takes the point cloud in coordinates with respect to the victim camera, the resolution of the attacker's surrogate camera, and the camera's FoV. We also include a maximum projection distance to account for the distance limit of the attacker's projector. The algorithm first computes the distance matrix from each LiDAR point to the origin (Line 2). It then calculates the horizontal and vertical FoV to identify each LiDAR point ($p_c = [x, y, z]$) with Equation 1 (Lines 3-4), where α is x for horizontal FoV and y is for vertical FoV.

$$\text{required\_fov} = |\text{atan2}(\alpha, z) \cdot 180/\pi| \tag{1}$$

Algorithm 1 Filtering 3D Points in an Attack Scene

Input: p_c: List of points in 3D point cloud, (C_w, C_h): Surrogate camera's width and height resolution, (C_{hfov}, C_{vfov}): Surrogate camera's horizontal and vertical FoV, d: max projection distance

Output: selected_points: 3D points visible to the camera.

```
 1: function FILTER_3D($p_c$, $C_w$, $C_h$, $C_{hfov}$, $C_{vfov}$)
 2:     distVictim = compute_distance($p_c$, origin)
 3:     $p_{hfov}$ = compute_horizontal_fov($p_c$)
 4:     $p_{vfov}$ = compute_vertical_fov($p_c$)
 5:     selected_points = [ ]
 6:     for $p \in p_c$ do
 7:         distanceOk = distVictim[$p_c$] < $d$
 8:         hfovOk = $p_{hfov}$[$p_c$] < $C_{hfov}/2$
 9:         vfovOk = $p_{vfov}$[$p_c$] < $C_{vfov}/2$
10:         if distanceOk & hfovOk & vfovOk then
11:             Add $p_c$ to selected_points
12:         end if
13:     end for
14:     return selected_points
15: end function
```

We discard points whose distance exceeds the specified range (Line 7). We also check the FoV of each point (computed at Lines 3-4) against the surrogate camera's FoV (Lines 8-10) to remove points that fall out of FoV range. The algorithm returns the list of points only visible to the camera. We then project the filtered 3D coordinates of points onto the 2D image coordinates to determine the attack zones in 2-dimensional space (where the image will be perturbed). To accomplish this, we compute the camera's intrinsic matrix c, a matrix based on camera parameters commonly used in plane-based camera re-sectioning and calibration [44]:

$$c = \begin{pmatrix} \frac{\iota_w}{2*\tan(\mathrm{FoV}*\frac{\pi}{360})} & 0 & \frac{\iota_w}{2} \\ 0 & \frac{\iota_w}{2*\tan(\mathrm{FoV}*\frac{\pi}{360})} & \frac{\iota_h}{2} \\ 0 & 0 & 1 \end{pmatrix} \tag{2}$$

Here ι_w and ι_h are the width and height of the camera resolution, and FoV is the camera's field of view. Should the viewing angles of the victim camera and the attacker surrogate camera be different, we use the FoV of the intersection of their two views to ensure that we only project to areas that the victim camera can see. Lastly, to obtain their 2D projection onto the image, we take the dot product of c (Equation 2) with filtered 3D points.

5.1.2 Attack Zone Generation. Once the valid 3D point clouds are projected onto the 2D image of the attack scene, we generate the attack zones, which define the valid regions within a scene used for physical manipulations. Because a 3D point cloud is not contiguous, we provide a threshold k to determine the number of pixels around a projected 3D point covered by that point. Specifically, k defines the maximum number of pixels away in the x or y dimension that is still considered to be part of the projected point. The k value depends on the maximum projectable distance and point cloud density. For lower point densities and higher projector distances, a higher k accounts for a more extensive spread of points on an adjacent surface.

We then generate an image mask for each pixel. If the pixel is covered by a 3D point, we set *true* for the pixel and *false* otherwise. Therefore, we ensure that every pixel with a value of *true* represents a valid point in the attack scene for the projection attack. With 2D zones mapped onto the image, the pixels within each zone can be readily perturbed to achieve the attack goal.

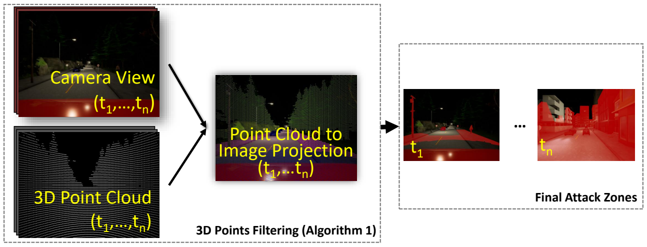

Figure 5: Illustration of attack zone generation steps.

Figure 5 illustrates the attack zone generation process with the example of a target vehicle traveling down a roadway. The attacker takes the 3D point cloud of the environment and the camera's view and superimposes the point cloud onto the surrogate camera feed (covering the same viewing angle as the victim) through Algorithm 1. The attacker then converts the 3D point cloud into valid tracker hijacking attack zones highlighted in red color. We note that for each scene frame ($t_1, \ldots, t_n$), the valid attack zones are continuously updated to reflect the surrounding environment.

5.2 Attacking Object Trackers

We present the specifics of our physically launched attack against object tracking and detail our process for generating the noise fed into the projector. We focus on generating stealthy noise-based perturbations that would be hard for a human to notice.

5.2.1 Physical Adversarial Perturbation Generation. Algorithm 2 encapsulates the adversarial perturbation process. Siamese-based object trackers use a Region Proposal Network (RPN) to classify each pixel in an image as either a background (to be ignored) or a target (to be tracked). Over a given number of "proposals" (attempts), each pixel is given a *regression label* p_r (which corresponds to whether or not an object is contained within the pixel) in addition to its classification label p_c of either "target" or "background". Combined with a loss function, accuracy increases as more proposals are made.

Given an input frame I, a given number of proposals to generate N, binary classification loss L_c, bounding box regression loss L_r, and correct (benign) classification and regression labels p_c and p_r generated by the tracker in the previous frame, which together represent the most recent result of the tracking algorithm, the loss function for the region proposal network is given by:

$$L(I, N, \theta) = \sum_{n=1}^{N} [L_c(I_n, p_c, \theta) + \lambda * L_r(I_n, p_r, \theta)] \tag{3}$$

where I_n is the proposal for input frame I at proposal n, λ is a fixed weight used to smooth L_1 loss for regression, and θ is the region proposal network's parameters to be optimized [21].

The RPN makes a series of N-1 proposals, with each proposal aimed to minimize L_c and L_r. The p_c and p_r from the previous frame ensure spatiotemporal consistency across multiple frames over time, and are required for correct tracking results. We take advantage of this fact by creating a corresponding incorrect *pseudo classification label* p_c^* and *pseudo regression label* p_r^*, to make both the binary classification loss (L_c) and the bounding box regression loss (L_r) the same for both the correct and pseudo labels. These pseudo labels encode the final tracking result we would like to achieve; thus, by making the pseudo labels indistinguishable from the correct labels, we are able to control the tracking results.

We codify the attack goal purely as the desired position and size the attacker wants the tracker to have. Recall that we have two types of tracker hijacking attack: move-in or move-out. In the move-in attack, an object's tracker is moved from the side of the camera's view across the center of the view. In the move-out attack, an object's tracker is moved away from the center view of the camera to the side of the view. Both attacks work the same way, but they differ in where the final position is in relation to the victim vehicle's camera. If the target region and the object tracker's initial region are on opposite sides of the center of the camera's view, the attack is a move-in attack; otherwise, it is a move-out attack.

To encode the attack goals in terms of classification and regression, we set p_c^* to be the inverse of p_c, un-targeting the original target region and targeting the previously non-target region. We set p_r^* to be the desired location of the tracker bounding box. The "desired location" is controlled by the attacker with a target coordinate (x and y), as well as width and height. This allows us to customize the hijacking attack for different purposes. For example, in addition to conducting a move-out attack to cause a vehicle to crash, we can also craft a specific width and height similar to those of another vehicle making the same movement, as a protective measure against discovery.

As a further constraint, we ensure that attacks will not be detected by a standardized linear Kalman Filter implementation [46]. We do not assume that the victim has a Kalman Filter implemented; rather, we ensure that the attack results are similar regardless of whether or not a Kalman Filter is implemented. To do this, we run the Kalman Filter on the hijacked tracker and observe its effects. If the Intersection over Union (IoU) of the Kalman Filter results is less than 70%, we consider the noise ineffective, and discard that pattern before trying again. An attacker can also split a large movement into a series of smaller ones over time, to reduce the velocity of the hijacked tracker and ensure that a Kalman Filter will not correct for it.

With these constraints, we define the loss for tracker hijacking with θ that represents the network parameters of the tracker:

$$L_{adv}(I,N,\theta) = \sum_{n=1}^{N}[(L_c(I_n,p_c,\theta) - L_c(I_n,p_c^*,\theta)) + \lambda * (L_r(I_n,p_r,\theta) - L_r(I_n,p_r^*,\theta))] \tag{4}$$

The loss function is structured in terms of a *displacement* (difference) between the original (p_r) and target regions (p_r^*). This means that the deciding factor in the adversarial loss generation is the overall displacement of the bounding box, rather than the specific goal of move-in or move-out. A move-in attack that moves a bounding box left, from the side of the scene to the center, is functionally identical to a move-out attack that moves the bounding box by the same amount from the center to the left side of the scene. The move-in and move-out attack goals only differ in the situations in which they are deployed and their consequences. We discuss this further in Section 6.

Algorithm 2 Generating Tracker Hijacking Perturbations

Input: Input video v, target location t, attack area α.
Output: Adversarial perturbations to be projected.

```
function GENERATE_ADV_PERTURBATIONS($v$, $t$, $\alpha$)
    for frame in $v$ do
        for $m = 1$ to $M$ do
            $p_c, p_r$ = minIoU($m, m-1$)
            $p_c^*$ = inverse($p_c$)
            $p_r^* = t$
            gen_adv_loss($\alpha$)                ▷ Equation 4
            initialize_loss($m+1, m$)
        end for
    end for
    return Final noise pattern from perturbations
end function
```

Given an input frame I, we generate an adversarial perturbation over consecutive steps up to the total M perturbations:

$$I_{m+1} = I_m + \frac{e}{M} * \text{sign}(\frac{\delta L_{adv}}{\delta I_m}) \tag{5}$$

where I_m is the frame generated on the m^{th} adversarial perturbation and L_{adv} is the hijacking optimization loss loss of I_m.

We restrict the perturbation area to the attack area when perturbing the image. This ensures that the perturbations are projectable. The final adversarial perturbation that we use in our attack is I_M.

6 EVALUATION

We evaluate AttrackZone on three different Siamese trackers, DaSiamRPN [56], SiamRPN [26], and DaSiamRPN+ [56], against autonomous driving and video surveillance. Using these three models, we conduct (1) emulated attacks against test samples of trackers, (2) simulated attacks against recorded frames of traffic scenes in CARLA, and (3) real-world attacks using a real camera, projector, and vehicles in a controlled environment. In these experiments, we sought to validate the attack effectiveness (Section 6.3) and attack parameters (Sections 6.4.1-6.4.3). We performed our experiments on a laptop computer with a 2.6 GHz 2-core Intel i5 processor, an NVIDIA GTX 1650 GPU, and 8 GB RAM, using Python 3.8.10.

End-to-End Attack Impact. We evaluate the attack performance in autonomous driving and video surveillance because they are safety-critical applications with the consequences of physical injury and masked entry into restricted areas.

The impact of move-in or move-out attacks depends on the context of the attacked scenes. Specifically, we launch the move-in attacks for autonomous driving experiments when there is nothing in front of the victim vehicle, and we launch the move-out attack when there is a tracked object in front of the victim vehicle. While a move-in attack causes the vehicle to stop, a move-out attack causes the vehicle to crash. The move-in attacks are conducted for the video surveillance experiments when a benign person passes by, and a move-out attack on an entering person. In this case, a move-in attack causes a false alarm, and a move-out attack masks the entrance of a person into a restricted area. Although we evaluate both move-in and move-out attacks, we note that move-out attacks

Table 1: Details of evaluated Siamese object trackers.

	Distractor Aware?	Backbone	Acc. / Recall / EAO†
DaSiamRPN [56]	Y	AlexNet	0.56 / 0.34 / 0.326
SiamRPN [26]	N	AlexNet	0.49 / 0.46 / 0.244
DaSiamRPN+ [56]	Y	AlexNet-DA	0.59 / 0.28 / 0.383

† Accuracy, recall, and Expected Average Overlap (EAO) computed based on performance on VOT 2017/VOT 2018 benchmark samples [21].

Table 2: Attack success rate on the VOT dataset.

Source Tracker	DaSiamRPN	93.3%	100%	93.3%
	SiamRPN	93.3%	100%	93.3%
	DaSiamRPN+	93.3%	100%	93.3%
		DaSiamRPN	SiamRPN	DaSiamRPN+
		Destination Tracker		

have a greater impact in both evaluated domains. Combined with the fact that the conditions for move-out attacks are generally more likely than move-in in these settings, we conducted more move-out attacks in our experiments.

Following standard conditions suggested by projector-based attacks on object detectors [35], we ensure that our projector can project perturbations of the same intensity as the algorithm generates, as dictated by the lighting conditions in the environment.

6.1 Attack Scenarios and Setup

Siamase Trackers and Dataset. Table 1 details the properties, models, and performance of the state-of-the-art [37] Siamese trackers: DaSiamRPN, SiamRPN, and DaSiamRPN+. The trackers are based on the AlexNet CNN [25]. SiamRPN is a non-distractor-aware tracker, while DaSiamRPN and DaSiamRPN+ are distractor-aware trackers. This means that in addition to classifying targets and backgrounds, they also classify distractors, which are objects that look like targets but are not supposed to be tracked. DaSiamRPN+, in contrast to DaSiamRPN, includes an additional step that modifies the AlexNet backbone to be distractor aware. Although they achieve competitive results, DaSiamRPN+ yields the highest accuracy and Expected Average Overlap (EAO), although it has the lowest recall. These trackers are trained on the state-of-the-art Visual Object Tracking (VOT) 2017 or 2018 datasets [24]. The VOT is a multi-purpose visual dataset that includes samples from various object-tracking applications, such as autonomous driving, video surveillance, sports tracking, and facial tracking.

Emulated Attacks. We use each Siamese model to generate hijacking perturbations (to move-in and move-out objects) against three trackers. We use the complete 15 test samples from the VOT dataset for autonomous driving (66.7%) and video surveillance (33.3%). Each video has a different resolution and aspect ratio that provides a wide range of conditions. In our experiments, because the VOT dataset does not provide the 3-D point cloud of a scene, we segment the images into valid and invalid attack zones based on their class. Cars, roads, sidewalks, buildings, and road signs in each scene are marked as valid attack zones, and manually reviewed to ensure the segmentation results are correct.

Simulated Attacks on CARLA. We used the DriveTruth automated autonomous driving dataset generator [33] to create a dataset for evaluating AttrackZone's attack performance. The dataset, created in the CARLA simulator, includes both image and LiDAR data collected in a realistic environment, It contains 15 20-second scenarios, recorded at 30 frames per second with various lighting and traffic conditions, as well as randomized distributions of pedestrians, cyclists, and vehicles with different appearances and positions. Of these scenarios, three were selected for move-in attacks, and twelve for move-out attacks.

Real-world Physical Attacks. We conduct real-world attacks against moving vehicles and pedestrians in controlled settings chosen for minimal traffic, such as on the highest level of a parking garage that is empty during daylight hours. Similarly, experiments on streets were conducted at night on residential roads with little traffic. All equipment was operated by individuals experienced in outdoor vehicle experiments. We used a real camera, projector, and vehicles, and generated practical scenarios for autonomous driving and video surveillance to evaluate both online and offline attacks. In the offline attacks, perturbations are generated in advance based on the victim vehicle/pedestrian route, at a stable rate of 30 frames per second (fps) to match that of the surrogate camera. In the online attacks, we generate the perturbations at run-time while the victim vehicle/pedestrian is moving based on the images fed from the attacker's surrogate camera. We use an average rate of 2.5 fps in online attacks, dependent on CPU and GPU speed, to generate the perturbations. We use an HD camera to record attack video footage in 1920x1080 resolution and an Optoma LV130 mini portable projector [38] to project the generated perturbations. This specific resolution is selected as it is widely supported by AD and surveillance camera manufacturers on the market [28].

6.2 Evaluation Metrics

We define *attack success rate* by the accomplishment of a given attack goal, move-in or move-out. In line with previous work [22], we consider a move-out attack successful when the tracker's intersection with its original region is 0. We consider a move-in attack successful when the bounding box overlaps the center of the x-axis of the image. Under these conditions, physical consequences will arise. We additionally present the successful attacks between DaSiamRPN/DaSiamRPN+ and SiamRPN to demonstrate *inter-transferability*, showing that the attack can be launched across Siamese models. Meanwhile, the successful attacks between DaSiamRPN and DaSiamRPN+ demonstrate *intra-transferability*, showing that the attack can be launched between the same Siamese model trained on a different dataset.

6.3 Attack Effectiveness

6.3.1 Emulated Attack Results. We assess the effectiveness of AttrackZone by directly applying the generated perturbations for autonomous driving and video surveillance samples in the VOT dataset. In both the autonomous vehicle and video surveillance subsets, 20% of the cases were move-in attacks, and the other 80% were move-out. Table 2 shows the attack success rate on three Siamese trackers and their intra and inter transferability.

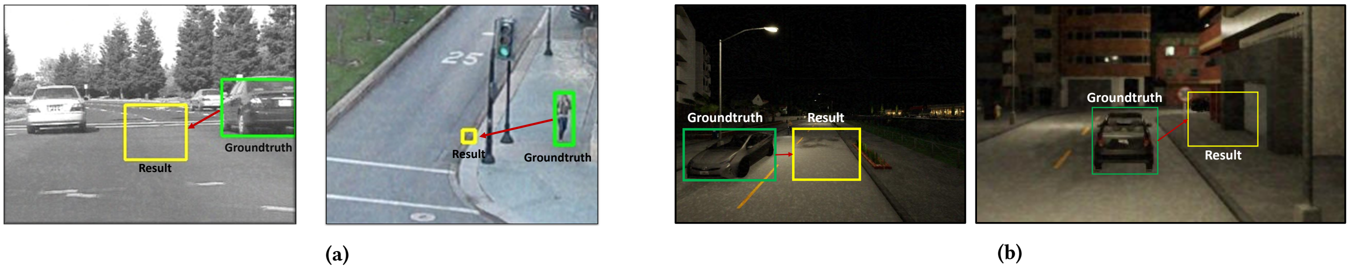

Figure 6: Illustration of example attacks on the autonomous driving and video surveillance systems. (a) Move-in attacks against VOT dataset, (b) Move-in (left) and move-out (right) attacks on the CARLA dataset. The roads, surrounding buildings (but not trees) and vehicles are considered valid attack surfaces.

Table 3: Attack success rate on the CARLA dataset.

		Destination Tracker		
		DaSiamRPN	SiamRPN	DaSiamRPN+
Source Tracker	DaSiamRPN	93.3%	93.3%	86.7%
	SiamRPN	93.3%	93.3%	86.7%
	DaSiamRPN+	93.3%	93.3%	86.7%

Table 4: Attack success rate in the physical world (values in bold are Offline and values in light are Online attacks).

		Destination Tracker		
		DaSiamRPN	SiamRPN	DaSiamRPN+
Source Tracker	DaSiamRPN	**88.9%** / 88.3%	**88.9%** / 88.3%	**100%** / 66.7%
	SiamRPN	**88.9%** / 72.7%	**77.8%** / 90.9%	**100%** / 100%
	DaSiamRPN+	**71.4%** / 60%	**85.7%** / 80%	**100%** / 80%

We found that the SiamRPN, a non-distractor-aware tracker, is the most susceptible to attacks, with 100% successful attacks. DaSiamRPN and DaSiamRPN+ are both equally susceptible, with 93% successful attacks. The destination model affects the attack’s performance, with the source model producing the same results. While the trackers behave slightly differently depending on the model used to generate the noise, the result of the tracker remains the same for each individual video. This suggests that with environmental factors remaining the same, noise generated on a particular Siamese tracker is universally effective against other Siamese tracking models. Figure 6a presents the impact of successful move-in attacks for the autonomous vehicle and video surveillance domains.

We found that the high success rates for all trackers are mainly due to the noise being applied directly to the video frames, preventing environmental factors such as lighting from affecting the results. This lack of interference from environmental factors also leads to consistent results. Additionally, the root cause of 6.7% of unsuccessful attacks is the rapidly fluctuating environment (due to a shaky camera), which prevents AttrackZone from generating hijacking noise consistently across multiple frames.

6.3.2 Carla Dataset Results. We evaluate the effectiveness of AttrackZone by conducting attacks on the DriveTruth-generated CARLA simulated dataset.

Attack Scenario Creation. We generate a set of scenarios likely to occur in a real-world tracker hijacking attack, and replicate them in CARLA. Within the simulation, a victim vehicle (of randomized make and model) drives around realistic residential and city areas while simultaneously recording video (via the front camera) and LiDAR data. Other traffic agents and pedestrians are randomized, providing a range of population densities for both vehicles and humans. All agents (vehicles, pedestrians, and cyclists) within the simulation obey traffic laws, road signs, and speed limits.

Table 3 shows the results of the simulated experiments. Figure 6b demonstrates two successful attacks on the CARLA dataset. DaSiamRPN+ proved more resistant to attacks than DaSiamRPN and SiamRPN, with 13.3% of attacks failing, suggesting that further training on distractors can improve robustness against attack (further discussed in Section 7). Over 50% of the failures were due to a lack of attack zones, discussed further in Section 6.4.3. Besides this factor, the destination model is the most important factor in success, with consistent results no matter the source model.

6.3.3 Real-world Experiments. We evaluate the AttrackZone success rate by examining the performance of trackers on the victim’s video as we use a projector to physically perturb the environment.

Attack Scenario Creation. A practical attack in the real world, unlike the emulated and simulated attacks, requires the following: (1) be effective across continuous video frames as the surrounding scene is changing, (2) generate the perturbations at the target camera, and (3) cause an end-to-end impact on driving and surveillance.

To achieve these requirements, we create a set of attack scenarios in a controlled environment. For autonomous driving experiments, we drive a GMC Sierra truck or a Hyundai Sonata sedan in front of the victim vehicle in residential environments with buildings, traffic signs, or other structures, which we leverage to cause the victim vehicle to rear-end the vehicle in front of it. For video surveillance experiments, we simulate a scenario where an attacker attempts to disguise entry into a building. We set up a security camera to watch a particular door in a campus building and have the attacker try to enter the door while projecting noise onto the opposite wall to mask unauthorized entry.

Unlike the emulated and simulated attacks, where the same set of videos is used across all experiments, the real-world experiments were conducted on three different video sets, one for each source

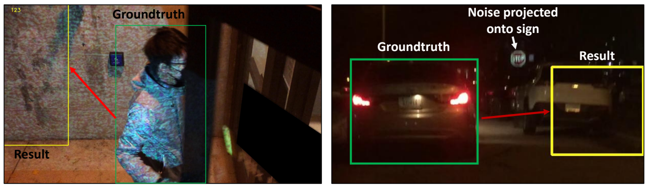

Figure 7: Offline move-out attack conducted on pedestrian (left), disguising his entry without setting of an alarm, and a vehicle (right), causing the victim vehicle to rear-end it.

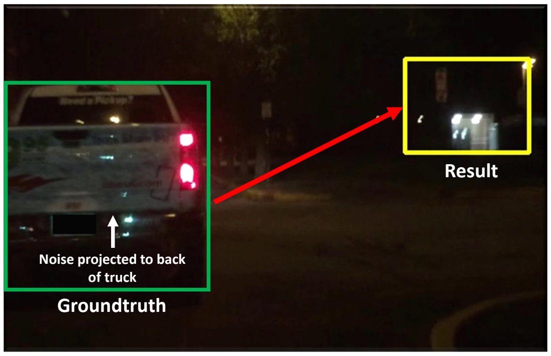

Figure 8: Online move-out attack conducted on a vehicle.

tracker. Because it is impossible to conduct different attacks simultaneously, experiments had to be split based on the model used to generate the hijacking noise. However, each set of videos was taken in the same places within 2-4 hours of each other to minimize environmental factors that may affect the results. Additionally, in the absence of a LiDAR sensor, we manually mapped valid attack zones for the environments we tested.

Offline Tracker Hijacking Attacks. We conduct 9 offline real-world attacks for each source tracker with different destination trackers to assess their inter- and intra-transferability. Table 4 shows the offline attack success rate in bold values. Our scenarios contain one pedestrian tracking and 8 vehicle tracking examples, two of which are shown in Figure 7.

We found that DaSiamRPN+ is the most susceptible to attacks, with success rates between 71.4% and 100%, due to more accurately honing in on the hijacking noise as a proper target. Unlike the emulated and simulated attacks, we observe a wide variance in success rate between different source and destination model pairs, likely due to differing environmental factors in the video datasets (separated by source model). 50% of all failures are attributed to the noise being washed out by ambient light sources, such as the tail lights of the leading vehicle. The rest of the failures are caused by the environment changing between offline attack zone/noise generation and the actual attack (e.g., a car passing by unexpectedly during the attack, changing the attack zone layout).

Online Tracker Hijacking Attacks. For the online real-world attacks, we conduct 6 vehicle tracking attacks for DaSiamRPN, 11 vehicle tracking attacks for SiamRPN, and 5 vehicle tracking attacks for DaSiamRPN+. Although there is a different number of videos, they represent the same areas over the same duration of time. For example, SiamRPN has shorter but more numerous videos compared to DaSiamRPN+. This split was done due to outside interruptions during the experiments (e.g., a jaywalker interrupting the drive). No pedestrian tracking attacks were conducted for online attacks. Because security cameras observe a static area that remains relatively unchanged, it is trivial to conduct an online attack compared to the vehicle tracking case, where the vehicle's camera may encounter many different environments in a short period of time.

Table 4 shows the online attack success rate in light numbers. Figure 8 shows the results of an online move-out attack conducted against a vehicle. We see a marked decrease in success rate compared to offline attacks. This is because online attacks, bounded by computational speed, played at a much slower rate, which led to the environment changing faster than the attack could account for. With the noise updating less frequently, the non-Distractor-aware SiamRPN was the most susceptible to attack, with an average of 86.41% successful attacks regardless of source or destination model. In fact, with SiamRPN as both the source and destination model, the online attack was more successful than the offline attack, indicating that SiamRPN has the most flexibility in desynchronization compared to other models. DaSiamRPN and DaSiamRPN+ were more successful in classifying the desynchronized noise as a distractor, averaging an 82.2% and 73.69% success rate, respectively.

6.4 Attack Parameters

6.4.1 Attack Duration. We measure the number of frames it takes for the move-in or move-out attack goals to be successful for each attack. Because all videos used experimentally run at 30 frames per second, "frames to success" gauge the amount of time an attacker needs to succeed. Figures 9a-9c show the mean, median, and standard deviation for the number of frames required to be successful across models for the emulated attacks, offline real-world attacks, and online real-world attacks, respectively. The y-axis plots the number of frames to succeed on a logarithmic scale, with the median bar represented as a black line. Each item on the x-axis is a source model, with box plots representing destination models.

For the emulated attacks, where conditions are uniform, the destination dataset is the main deciding factor in how long it takes to succeed. The SiamRPN model takes the shortest amount of time on average. However, there is a significant variation in the number of frames to success, indicating that environmental factors are critical in determining the time needed for a successful attack. For example, the base DaSiamRPN model can range from a few frames needed for a successful attack to hundreds of frames required, depending on the environment and source dataset. We also observe that outliers can far outpace the median and average, with some outliers over 100 times the average. As a general trend, we observe on average 10-100 frames are the norm for a successful attack in all cases, which amounts to 0.3 to 3 seconds of continuous attack.

Real-world attacks perform similarly to emulated attacks, typically taking 10-100 frames to complete. Yet, there is a much greater distribution due to the more significant variation in environmental factors. Considering the median frames-to-success, offline attacks are successful faster than online attacks due to the higher frame rate at which the noise is changed.

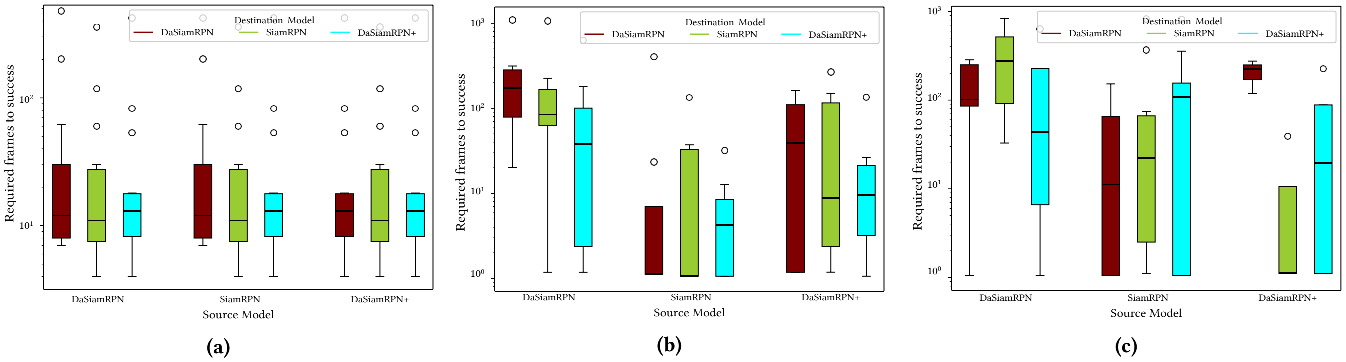

Figure 9: Distribution of the number of frames required for (a) emulated, (b) offline, and (c) online real-world attacks.

Table 5: Average attack zone utilization for successful cases for real world physical attacks (bold), the VOT dataset attack (italic), and the CARLA dataset attack (light).

	DaSiamRPN	SiamRPN	DaSiamRPN+
Average	**62.8%** / *83.3%* / 81.6%	**60%** / *83.5%* / 81.8%	**63.2%** / *83.3%* / 81.6%
Std Dev	**15.4%** / *19%* / 21.3%	**11.9%** / *18.73%* / 21%	**14.8%** / *19%* / 21.3%

6.4.2 Amount of Perturbations. For the emulated and simulated experiments, it is possible to compute the maximum per-pixel perturbation used to conduct a tracker hijacking attack. This is because we inject the noise pattern into the attack zones in these examples, directly altering the image. Maximum perturbations are not a useful metric for real-world physical attacks. Directly injecting the noise pattern could allow an attacker to completely transform an image in the digital domain. However, it is impossible to physically change the environment with a projector in real-world attacks. The maximum number of perturbations for the emulated attacks is stable across source models (14.32 ± 0.07 for DaSiamRPN, 14.35 ± 0.06 for SiamRPN, and 14.34 ± 0.06 for DaSiamRPN+), with a low standard deviation and few outliers. The average modification to a pixel's RGB value is under 5.86% (15) per channel. The simulated attacks require less perturbation to be successful (10.05 ± 0.22 for DaSiamRPN, 10.25 ± 0.72 for SiamRPN, and 10.5 ± 1 for DaSiamRPN+) compared to the emulated attacks, which have an average pixel modification of under 4.7% (12) perturbations per channel. This is due to the greater prevalence of brighter lighting in the VOT dataset. Brighter surfaces can wash out weak perturbations, requiring stronger hijacking noise for an attack to be successful.

6.4.3 Attack Zone Utilization. We define the attack zone utilization (AZU) as the percentage of the victim camera's image over which noise is applied. This is directly related to the percentage of the image that is a valid attack zone. We examine AZU in successful and unsuccessful attacks to infer the effect of AZU on success rate.

For real-world attacks, calculating the AZU is difficult because it is hard to quantify exactly where the noise has been applied due to environmental factors like light sources washing out projector images. Therefore, although Table 5 shows the raw AZU for successful attacks in the real world (in bold), we note that the actual AZU might be lower. On average, we observe that the AZU is 62% ± 14% across the three models.

Because our attack on the VOT dataset involves directly manipulating the images, calculating its AZU is much easier. The AZU for successful attacks is shown in Table 5 (in italics). The AZU on the entire image set across three models is on average 83.5% ± 18.7%. Similarly, AZU of attacks on CARLA dataset yields results (Table 5, in light) similar to the VOT attack. The overall average AZU is 81.7% ± 21.2%.

For real-world physical attacks, the AZU is similar between successful and unsuccessful cases, indicating that attack zone availability is not important for success in physical experiments. However, for the emulated attacks, we find that the AZU for the unsuccessful cases across models was 88.6% ± 7.4%, 5% higher than the average for both the DaSiamRPN and DaSiamRPN+. Intuitively, a more significant AZU should mean a higher success rate, as the attack is able to perturb more parts of the image, but this has proven not to be the case. Further, for the CARLA simulated experiments, failure cases had an AZU at least 30% lower than average. This suggests, within a margin of around 30% of the image's area, overall AZU does not matter as much as the environmental factors, such as the motion of the car and the quality of the image.

7 DISCUSSION AND LIMITATIONS

7.1 Practical Considerations

Generality of Attack Zones and Noticeability. We demonstrated the viability of attack zones to conduct tracker hijacking attacks on object trackers. However, attack zones can be extended to other attacks that require projector surfaces. For example, AttrackZone can find valid areas for conducting the projector-based phantom attacks [35]. There are also other attacks against Siamese trackers, such as the C-S attack, an availability attack that removes trackers entirely [50]. This can be expanded to work via AttrackZone because it operates on heatmap manipulation. Additionally, adversarial perturbation methods for other tracker architectures have been proposed [22]. Although these different tracking algorithms have incompatible architectures and separate vulnerabilities, zone generation is algorithm agnostic. Therefore, it can be used to launch physical attacks on non-Siamese trackers, increasing the transferability of AttrackZone. We will examine other attacks that can be conducted via the attack zone generation process.

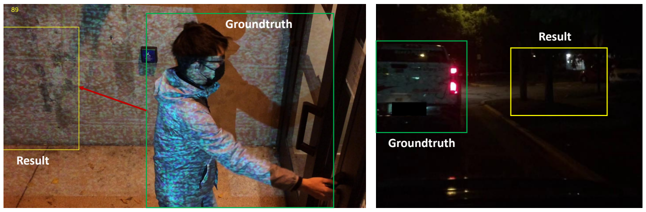

Figure 10: Applying noise cancellation defense against AttrackZone. Tracker remains hijacked.

Surrogate Views. Flying a drone close to the victim camera is the most convenient approach for creating a similar, adaptable viewpoint mimicking the victim's view. Drones flying close to victim vehicles have been suggested by previous works [35, 55]. However, other approaches for creating the surrogate view are possible, including discreetly attaching a camera to the victim's car, or transforming the camera's view with a dissimilar view angle via the camera's intrinsic matrix. If none of these methods are viable for an attacker, the attack can be conducted offline and the surrogate view sourced via public mapping data.

Multi-Object Tracking. Siamese trackers, including DaSiamRPN, only support single-object tracking [56]. However, real-world closed-source implementations extend Siamese tracking to multiple objects [9]. AttrackZone works the same way for multi-object tracking, with a single-object tracker for each object in the scene. We will investigate AttrackZone's performance on multi-object tracking.

Viewing Angles. Many existing perception attacks consider viewing angles by design (e.g., [3, 11, 54]) as their adversarial noise is placed on a stationary object. As the vehicle is driving forward, the viewing angle of the object is gradually changing. On the contrary, AttrackZone is not limited by such a constraint because a drone might fly close to the victim camera and change its orientation/position, compensating for the changes in viewing angles.

Limitations. Similar to other projector attacks in the physical world, AttrackZone is also limited by the brightness of the environment and the availability of projectable surfaces. For example, a much more powerful projector is required to conduct a projector attack during the daytime. Additionally, there might be very few visible surfaces in flat, open fields to project perturbation.

Because a projector competes with other light sources, a projector attack may be complicated by bright headlights or other sources of noise pollution while driving. Our real world experiments show that standard headlights and brake lights are insufficient to negate the threat of tracker hijacking via a projector. However, bright light sources, such as custom LED lights on cars or floodlights on city streets, might unusually affect the AttrackZone performance, either washing out projector noise or removing usually valid attack zones by making them too bright to use.

7.2 Countermeasures

Noise Cancellation. Previous works have applied noise cancellation to account for noise-based attacks applied directly to an image [21, 39]. However, because the noise pattern shown in the physical domain is different from one generated in the image domain (due to environmental differences arising from projector strength, lighting, and the absorption coefficients of physical objects), effective removal without degrading the tracker accuracy is not practical.

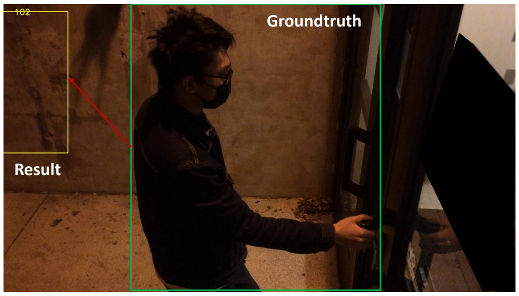

Figure 11: Noise cancellation resulting in unintentional tracker hijacking when no attack is conducted.

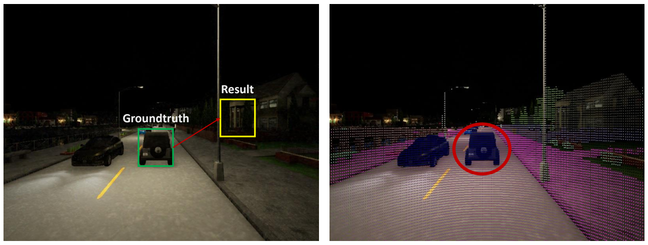

Figure 12: Effects of sensor fusion on attack results. Although the object tracker is hijacked (left), the original object still registers on the LiDAR (right, circled).

We applied a noise cancellation algorithm specifically tailored to defend against tracker hijacking [21] to our successful experimental results. Figure 10 shows the effect of noise cancellation on our attack. The results are not altered, and tracker hijacking remains viable under this defense. Noise cancellation can lead to *unintentional* tracker hijacking even when no noise is applied. This is shown in Figure 11. Although no attack was performed, the noise cancellation was capable of moving the tracker on its own, making it an undesirable method of defense.

Sensor Fusion. AttrackZone directly attacks the vision pipeline of autonomous systems, altering the perception of the camera. Two dangerous use cases for tracker hijacking in autonomous driving involve causing the vehicle perception to conclude that an object has moved into or out of the vehicle's path, causing it to stop prematurely or rear-end a vehicle. An additional non-camera sensor such as a LiDAR or radar can mitigate hijacking attacks. These sensors register the presence (or lack of presence) of obstacles in a vehicle's path. This helps minimize the chance of serious injury occurring due to a tracker hijacking attack. Figure 12 illustrates the effect of the attack under sensor fusion. Although the tracker is still hijacked, the LiDAR still sees the object in its original place. However, recent work has shown that sensor fusion algorithms are vulnerable to an adversary who is able to compromise only one of the fusion sources [30, 34, 42], which means using AttrackZone alone can potentially bypass some sensor fusion algorithms; let alone that existing LiDAR perception attacks [47, 55] can be used

Table 6: Comparison of AttrackZone against current tracker hijacking attacks.

	SORT [22]	RTAA [21]	AttrackZone
Full Autonomous System Pipeline	✓	✓	✓
Physical Domain Evaluation	◐†	✗	✓
Valid Patch/Projector Location	✓	✗	✓
Transferability between Trackers	✗	✗	✓

† The SORT Attack digitally manipulates the images, but the attack is constrained to surfaces that can be attacked through a patch, making it reasonable to assume that this attack could be transferred to the physical domain with minimal modifications.

in conjunction with AttrackZone by a powerful attacker. We focus on a camera-based object tracking system and leave the evaluation of AttrackZone on sensor fusion as future work.

Building Robust Trackers. The source of the vulnerability of Siamese trackers to tracker hijacking arises from the design of the heatmap, making building robust trackers an attractive option for defense. Unfortunately, previous attempts at creating robust Siamese trackers have relied on white-box knowledge of how an attack works and the assumption that the attack operates over the entire image [21]. In AttrackZone's case, because the attack operates on dynamically calculated portions of the image unless the victim has a LiDAR mapping on hand and knows where the attacker is projecting from, these assumptions do not hold. While having a LiDAR mapping is not an issue for vehicles that already operate on LiDAR, tracking attacker projections in real-time without prior knowledge would add unreasonable amounts of overhead for the victim. Nevertheless, previous works have found that training Siamese trackers on distractors greatly improved their performance [56]. Training Siamese trackers with noise patterns labeled as distractors will likely strengthen their resistance to AttrackZone, which will be a part of our future work.

8 RELATED WORK

Attacks against Object Trackers. Table 6 presents the capabilities and drawbacks of two existing, representative tracker hijacking attacks [21, 22]. Other attacks against object tracking either do not consider physical feasibility [8, 19, 51], or for the one that does [10], its attack goal is bounding box reshaping not hijacking. While all attacks target the full autonomous system pipeline (both object detection and object tracking), AttrackZone is unique in that it can hijack object trackers in the physical domain while transferring across multiple models.

Projector-based Attacks. Previous work in projector-based attacks has mainly focused on fooling object detection by projecting a pattern onto a surface, altering the lighting conditions to cause an object classifier to misclassify images [12, 36]. Unlike our work, these works attack object classification and use phase modulation on light rather than adversarial noise. Further work involves projecting depthless objects, like stop signs, onto hard surfaces visible to the camera [35]. In this case, an actual image of an object is projected, causing the AD system to misinterpret the projection as a real object and react accordingly. The vulnerability is not related to misclassification; instead, it is due to a lack of contextual understanding by the AD system as a whole. The tracker hijacking attack implemented by AttrackZone is less visible to humans due to projecting human-unintelligible noise instead of objects.

9 CONCLUSIONS

We present AttrackZone, a new physically-realizable tracker hijacking attack against Siamese object trackers. AttrackZone systematically determines valid surfaces within an environment used for physical perturbations. It then exploits the heatmap generation process of Siamese Region Proposal Networks to take control of an object's bounding box. We evaluated the efficacy of our attack on the original Siamese tracking test dataset, on a diverse simulated dataset generated in the CARLA simulator, and on real-world experiments against object tracking in autonomous vehicles and pedestrian tracking in video surveillance systems. We demonstrated attack inter-transferability across different types of Siamese trackers and intra-transferability across different model parameters. AttrackZone yields an average attack success rate of 92% in 0.3-3 seconds, and runs efficiently in real-world offline and online attacks.

ACKNOWLEDGMENTS

This work has been partially supported by the National Science Foundation (NSF) under grants CNS-2144645 and CNS-1801611, the Army Research Office (ARO) under grant W911NF2110320, and through a gift by Qualcomm. The views expressed are those of the authors only. We would like to thank Ruoyu Song for playing the role of the attacker in the real-world pedestrian move-out attacks.

REFERENCES

[1] Na An and Wei Qi Yan. Multitarget tracking using siamese neural networks. *ACM Transactions on Multimidia Computing Communications and Applications*, 2021.

[2] Federico Boniardi, Abhinav Valada, Wolfram Burgard, and Gian Diego Tipaldi. Autonomous indoor robot navigation using a sketch interface for drawing maps and routes. In *IEEE International Conference on Robotics and Automation*, 2016.

[3] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. Invisible for both Camera and LiDAR: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. *IEEE Symposium on Security and Privacy (S&P)*, 2021.

[4] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. In *ACM Conference on Computer and Communications Security (CCS)*, 2019.

[5] Subhash Challa, Mark R Morelande, Darko Mušicki, and Robin J Evans. *Fundamentals of object tracking*. Cambridge University Press, 2011.

[6] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Polo Chau. Shapeshifter: Robust physical adversarial attack on faster R-CNN object detector. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018.

[7] Xi Chen, Xiao Wang, and Jianhua Xuan. Tracking multiple moving objects using unscented kalman filtering techniques. In *International Conference on Engineering and Applied Science*, 2012.

[8] Xuesong Chen, Canmiao Fu, Feng Zheng, Yong Zhao, Hongsheng Li, Ping Luo, and Guo-Jun Qi. A unified multi-scenario attacking network for visual object tracking. In *AAAI Conference on Artificial Intelligence*, 2021.

[9] Aman Dhar. Object tracking for autonomous driving systems. Master's thesis, EECS Department, University of California, Berkeley, May 2020.

[10] Li Ding, Yongwei Wang, Kaiwen Yuan, Minyang Jiang, Ping Wang, Hua Huang, and Z Jane Wang. Towards universal physical attacks on single object tracking. In *AAAI Conference on Artificial Intelligence*, 2021.

[11] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] Abhiram Gnanasambandam, Alex M Sherman, and Stanley H Chan. Optical adversarial attack. In *IEEE/CVF International Conference on Computer Vision*, 2021.

[13] Google earth. https://earth.google.com/, 2022. [Online; Accessed 27-August-2022].

[14] Lie Guo, Linhui Li, Yibing Zhao, and Zongyan Zhao. Pedestrian tracking based on camshift with kalman prediction for autonomous vehicles. *International Journal of Advanced Robotic Systems*, 2016.

[15] Song Han, William Shen, and Zuozhen Liu. *Deep drone: Object detection and tracking for smart drones on embedded system*. PhD thesis, Stanford University, 2016.

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[17] Cabell Hodge, Konrad Hauck, Shivam Gupta, and Jesse C Bennett. Vehicle cybersecurity threats and mitigation approaches. Technical report, National Renewable Energy Lab.(NREL), OSTI, 2019.

[18] Shahar Hoory, Tzvika Shapira, Asaf Shabtai, and Yuval Elovici. Dynamic adversarial patch for evading object detection models. *arXiv preprint arXiv:2010.13070*, 2020.

[19] Saurabh Jha, Shengkun Cui, Subho Banerjee, James Cyriac, Timothy Tsai, Zbigniew Kalbarczyk, and Ravishankar K Iyer. Ml-driven malware that targets av safety. In *Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2020.

[20] Xiaoyu Ji, Yushi Cheng, Yuepeng Zhang, Kai Wang, Chen Yan, Wenyuan Xu, and Kevin Fu. Poltergeist: Acoustic adversarial machine learning against cameras and computer vision. In *IEEE Symposium on Security and Privacy (S&P)*, 2021.

[21] Shuai Jia, Chao Ma, Yibing Song, and Xiaokang Yang. Robust tracking against adversarial attacks. In *European Conference on Computer Vision*, 2020.

[22] Yunhan Jia Jia, Yantao Lu, Junjie Shen, Qi Alfred Chen, Hao Chen, Zhenyu Zhong, and Tao Wei Wei. Fooling detection alone is not enough: Adversarial attack against multiple object tracking. In *International Conference on Learning Representations (ICLR)*, 2020.

[23] Sebastian Köhler, Giulio Lovisotto, Simon Birnbach, Richard Baker, and Ivan Martinovic. They see me rollin': Inherent vulnerability of the rolling shutter in cmos image sensors. In *Annual Computer Security Applications Conference (ACSAC)*, 2021.

[24] Matej Kristan, Jiri Matas, Aleš Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin. A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012.

[26] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[27] Shaoshan Liu, Jie Tang, Zhe Zhang, and Jean-Luc Gaudiot. Computer architectures for autonomous driving. *IEEE Computer Magazine*, 2017.

[28] Logical inst. screen resolution. https://www.logicalincrements.com/articles/resolution, 2021. [Online; Accessed 27-August-2022].

[29] Jiajun Lu, Hussein Sibai, and Evan Fabry. Adversarial examples that fool detectors. *CoRR*, abs/1712.02494, 2017.

[30] Yuzhe Ma, Jon A Sharp, Ruizhe Wang, Earlence Fernandes, and Xiaojin Zhu. Sequential attacks on kalman filter-based forward collision warning systems. In *AAAI Conference on Artificial Intelligence*, 2021.

[31] Yanmao Man, Ming Li, and Ryan Gerdes. GhostImage: Remote perception attacks against camera-based image classification systems. In *International Symposium on Research in Attacks, Intrusions and Defenses*, 2020.

[32] Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[33] Raymond Muller, Yanmao Man, Z. Berkay Celik, Ming Li, and Ryan Gerdes. DriveTruth: Automated autonomous driving dataset generation for security applications. In *International Workshop on Automotive and Autonomous Vehicle Security (AutoSec) collocated with NDSS*, 2022.

[34] Shoei Nashimoto, Daisuke Suzuki, Takeshi Sugawara, and Kazuo Sakiyama. Sensor con-fusion: Defeating kalman filter in signal injection attack. In *Asia Conference on Computer and Communications Security*, 2018.

[35] Ben Nassi, Yisroel Mirsky, Dudi Nassi, Raz Ben-Netanel, Oleg Drokin, and Yuval Elovici. Phantom of the adas: Securing advanced driver-assistance systems from split-second phantom attacks. In *ACM Conference on Computer and Communications Security (CCS)*, 2020.

[36] Dinh-Luan Nguyen, Sunpreet S Arora, Yuhang Wu, and Hao Yang. Adversarial light projection attacks on face recognition systems: A feasibility study. In *IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020.

[37] Milan Ondrasovic and Peter Tarabek. Siamese visual object tracking: A survey. *IEEE Access*, 2021.

[38] Optoma. Optoma lv130 ultra-portable projector. https://www.optoma.com/vn/product/lv130/, 2021. [Online; Accessed 27-August-2022].

[39] Margarita Osadchy, Julio Hernandez-Castro, Stuart Gibson, Orr Dunkelman, and Daniel Perez-Cabo. No bot expects the deepcaptcha! introducing immutable adversarial examples, with applications to captcha generation. *IEEE Transactions on Information Forensics and Security*, 2017.

[40] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 2015.

[42] Junjie Shen, Jun Yeon Won, Zeyuan Chen, and Qi Alfred Chen. Drift with devil: Security of multi-sensor fusion based localization in high-level autonomous driving under gps spoofing. In *USENIX Security Symposium*, 2020.

[43] Bing Shuai, Andrew G. Berneshawi, Xinyu Li, Davide Modolo, and Joseph Tighe. Siammot: Siamese multi-object tracking. *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[44] P.F. Sturm and S.J. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999.

[45] United states geological survey explorer. https://earthexplorer.usgs.gov/, 2022. [Online; Accessed 27-August-2022].

[46] Open Source Computer Vision. Opencv: Kalmanfilter reference. https://docs.opencv.org/3.4.1/dd/d6a/classcv_1_1KalmanFilter.html, 2022. [Online; Accessed 27-August-2022].

[47] Wei Wang, Yao Yao, Xin Liu, Xiang Li, Pei Hao, and Ting Zhu. I can see the light: Attacks on autonomous vehicles using invisible lights. In *ACM Conference on Computer and Communications Security*, 2021.

[48] Xiaoyu Wang, Ming Yang, Shenghuo Zhu, and Yuanqing Lin. Regionlets for generic object detection. In *IEEE International Conference on Computer Vision*, 2013.

[49] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *IEEE international conference on computer vision*, 2017.

[50] Bin Yan, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Cooling-shrinking attack: Blinding the tracker with imperceptible noises. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[51] Xiyu Yan, Xuesong Chen, Yong Jiang, Shu-Tao Xia, Yong Zhao, and Feng Zheng. Hijacking tracker: A powerful adversarial attack on visual tracking. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020.

[52] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 2020.

[53] Qi Zang and Reinhard Klette. Object classification and tracking in video surveillance. In *Lecture Notes in Computer Science*, volume 2756, pages 198–205, 2003.

[54] Yue Zhao, Hong Zhu, Ruigang Liang, Qintao Shen, Shengzhi Zhang, and Kai Chen. Seeing isn't believing: Towards more robust adversarial attack against real world object detectors. In *ACM Conference on Computer and Communications Security (CCS)*, 2019.

[55] Yi Zhu, Chenglin Miao, Tianhang Zheng, Foad Hajiaghajani, Lu Su, and Chunming Qiao. Can we use arbitrary objects to attack lidar perception in autonomous driving? In *ACM Conference on Computer and Communications Security (CCS)*, 2021.

[56] Zheng Zhu, Qiang Wang, Li Bo, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *European Conference on Computer Vision*, 2018.