On Data Fabrication in Collaborative Vehicular Perception: Attacks and Countermeasures

Qingzhao Zhang[1], Shuowei Jin[1], Jiachen Sun[1], Xumiao Zhang[1], Ruiyang Zhu[1],
Qi Alfred Chen[2], Z. Morley Mao[1,3]
[1]*University of Michigan,* [2]*University of California, Irvine,* [3]*Google*

Abstract

Collaborative perception, which greatly enhances the sensing capability of connected and autonomous vehicles (CAVs) by incorporating data from external resources, also brings forth potential security risks. CAVs' driving decisions rely on remote untrusted data, making them susceptible to attacks carried out by malicious participants in the collaborative perception system. However, security analysis and countermeasures for such threats are absent. To understand the impact of the vulnerability, we break the ground by proposing various real-time data fabrication attacks in which the attacker delivers crafted malicious data to victims in order to perturb their perception results, leading to hard brakes or increased collision risks. Our attacks demonstrate a high success rate of over 86% on high-fidelity simulated scenarios and are realizable in real-world experiments. To mitigate the vulnerability, we present a systematic anomaly detection approach that enables benign vehicles to jointly reveal malicious fabrication. It detects 91.5% of attacks with a false positive rate of 3% in simulated scenarios and significantly mitigates attack impacts in real-world scenarios.

1 Introduction

The perception system of connected and autonomous vehicles (CAVs) is safety-critical as its performance directly affects driving decisions [7, 8]. However, CAV's perception is confronted with the basic limitation that onboard sensors have limited sensing capabilities. For instance, LiDAR, the commonly adopted 3D sensor, cannot see through occlusions and may render low resolutions for far-away objects, leading to imperfect detection performance. Many recent efforts have proposed LiDAR-based collaborative perception algorithms [31, 71, 77, 84], where different nearby vehicles exchange perception information (e.g., raw sensor data or feature maps processed by neural networks) and perform object detection algorithms on the fused data. In terms of the accuracy of object detection, the approach significantly outperforms the traditional CAV collaboration [36, 37, 62] sharing simple GPS messages or object locations, as illustrated in related studies [71, 77]. CAV industry [2–4, 9, 14, 16, 79] also proposes solutions of collaborative perception and launch road testing across the globe.

Although collaborative perception is evolving quickly towards maturity, it introduces a severe vulnerability to vehicle safety because the safety-critical perception algorithms now rely on sensor data or feature maps from remote untrusted vehicles. With the control of a remote vehicle via physical access to either software or hardware, an attacker can fabricate the data to share, aiming to inject fake object detection results into the view of victim vehicles and even mislead them to trigger accidents. However, the impact of such a severe data integrity threat has not been comprehensively evaluated. Existing studies of CAV security [59, 83] either focus on other scopes (*e.g.,* physical sensor security [30, 67], network protocols [25, 78]) or assume a different threat model (*e.g.,* single-vehicle perception [54, 67], object-sharing collaboration [23, 85]), thus existing mitigation methods are not effectively designed for the new threat.

To bridge the gap, we propose a series of stealthy, targeted, and realistic attacks exploiting LiDAR-based collaborative perception in this study. Our proposed attacks can spoof or remove objects at specified locations in the victim's perception results, making all mainstream types of collaborative perception schemes vulnerable. For early-fusion systems which directly merge LiDAR point clouds, we propose black-box ray casting to reconstruct malicious but natural raw point clouds. We design offline adversarial object generation and run-time occlusion-aware point sampling to further optimize the distribution of modified points. For intermediate-fusion systems which merge feature maps as intermediate results of object detection models, we design a white-box adversarial attack to perturb the feature maps. For optimal efficiency, the adversarial attack initializes the perturbation vector via a black-box method and runs one-step backward propagation in each LiDAR cycle (*e.g.,* 100 ms). More importantly, we propose zero-delay attack scheduling to make attacks realizable in the real world. To be specific, in order to attack the perception of frame i, attackers prepare a fabrication plan

based on the knowledge of frame $i-1$ before the next frame comes. In this way, attackers earn one LiDAR cycle time to complete attack generation without introducing a noticeable delay in the fabricated data.

We evaluate the attack effectiveness on 211 traffic scenarios in a simulated dataset Adv-OPV2V and a real-world dataset Adv-MCity (including 8 scenarios collected from a real-vehicle testbed MCity [19]). On the simulated dataset, all attacks have a success rate of more than 86% regardless of fusion methods and model configurations. In our real-world experiments, we deploy three vehicles equipped with LiDAR/GPS sensors and the latest Baidu Apollo autonomous driving software [8]. Our attacks can be launched in real-time and trigger safety hazards such as collisions and emergent hard brakes. We also provide a comprehensive analysis of how the attack effectiveness is affected by various factors including attack methods, fusion schemes, and scenarios. Our findings will guide system designers to build robust collaborative perception schemes.

To mitigate the demonstrated attacks, we propose Collaborative Anomaly Detection (CAD), a system that detects data fabrication attacks by revealing geometry inconsistencies of the shared data from different vehicles. To achieve this, CAD requires each vehicle to generate and share an occupancy map, which is a 2D map labeling the 2D space into three classes, free, occupied, and unknown. On receiving occupancy maps from others, the vehicle validates the consistency of the maps, *i.e.,* there is no region classified as occupied and free at the same time. Then the vehicle carries out the second check by merging the occupancy maps into one and checking perception results against it. For instance, free regions should not overlap with detected bounding boxes; each on-road moving occupied region should have one bounding box overlap with it. In this way, abnormal detection results caused by either fabricated data or perception faults are revealed if the attacked region is observed by at least one benign CAV. We execute CAD on datasets Adv-OPV2V and Adv-MCity for evaluation. CAD detects 91.5% attacks with a false positive rate <3% and is general for different attack methods and perception systems.

As the first comprehensive security analysis of collaborative perception, we will open-source all the above attack-/defense practices as a benchmark tool to facilitate future research. Our contributions can be summarized as three-fold:

- We compile the benchmark datasets Adv-OPV2V and Adv-MCity for evaluating the security of collaborative perception. Especially, Adv-MCity is the *first* multi-vehicle collaboration dataset collected on real vehicles and real roads.
- We propose multiple data fabrication attacks, where one attacker, as a collaborative perception participant, can successfully spoof or remove objects at specified locations. We conduct an extensive study on the impact of such attacks.
- We develop CAD, a defense system of collaborative perception for detecting our proposed data fabrication attacks.

Table 1: Existing attacks on LiDAR collaborative perception.

Method	Targeted system	Requirements	Spoof	Remove	Targeted	Real-time
LiDAR spoofing [28–30, 67]	Single-vehicle	Laser emitters	●	●	●	●
Physical objects [69, 88]	Single-vehicle	Physical access to the target	○	●	●	●
False object messages [38]	Late-fusion	None	●	●	●	●
Multi-agent adversarial [70]	Int.-fusion	Local computation	●	●	○	○
Ours	**Early/Int.-fusion**	Local computation	●	●	●	●

Notes: *Int.* - intermediate-fusion.

Table 2: Effectiveness of defenses on our attacks.

Method	Threat model	Attack types		Leveraged information			Overall
		Spoof	Remove	Spatial	Temporal	Multi-vehicle	
Network integrity [17]	Cyber attacks	○	○	○	○	○	○
Trusted execution [41]	Malicious software	●	●	○	○	○	◐
Occlusion-aware [67]	Physical LiDAR spoofing	●	○	●	○	○	◐
Temporal check [54]	Physical LiDAR spoofing	●	●	◐	●	○	○
Multi-sensor check [54]	Physical LiDAR spoofing	●	●	○	○	○	◐
Spatial conflicts w/ ego [23]	Fake message in late-fusion	●	●	●	○	◐	◐
Ours (CAD)	Our attacks	●	●	●	○	●	●

CAD reveals abnormal perception results through the sharing of fine-grained occupancy maps.

2 Background and Related Work

Connected and autonomous vehicles (CAVs) are transforming the transportation systems by enabling automatic and intelligent vehicle driving control. CAVs are complicated cyber-physical systems equipped with sensors such as LiDAR, camera, and radar to perceive the surroundings, and software to make appropriate driving decisions. By the end of 2022, numerous companies including Waymo, Honda, Baidu, and Tesla [6, 8, 10, 13] have developed models of CAVs.

Collaborative perception has been proposed to enhance CAV perception [21, 46, 48, 51, 64] by sharing raw or processed sensor data among infrastructure or vehicles. Mainstream solutions focus on LiDAR sensors because of the rich 3D geometry features brought by LiDAR images. Collaborative perception has three major types according to the sharing data, as shown in Figure 1. CAVs in early-fusion sharing schemes [32, 47, 60, 84] directly exchange raw sensor data, whose format is usually universal and can be naively concatenated, at a cost of data transmission bandwidth; intermediate-fusion schemes [31,33,71,76,80] ask CAVs to transmit feature maps, the intermediate product of perception algorithms, offering a good tradeoff between network efficiency and perception accuracy; in late-fusion schemes [53, 65, 66] lightweight perception results such as object bounding boxes are shared.

Collaborative perception is advancing quickly towards real-world deployment. 3GPP standardized for Cellular Vehicle-to-Everything (C-V2X) techniques in 2017 [1], indicating the maturity of roadside communication. Since then, major technology companies such as Huawei, Intel, Bosch, Infineon, and Qualcomm [2–4, 9, 14] have strived to build various C-V2X solutions. Road trials have been launched across the globe in countries like Germany, France, the United States, and Japan. Ford [16] and Baidu Apollo [79] built real-world collaborative perception datasets.

Attacks on CAV perception. Several attacks can harm LiDAR perception systems as listed in Table 1. First, LiDARs on CAVs are vulnerable to physical attacks, such as GPS spoofing [50, 63], LiDAR spoofing [29, 39, 43, 50], and physical realizable adversarial objects [69, 82, 88]. These attacks

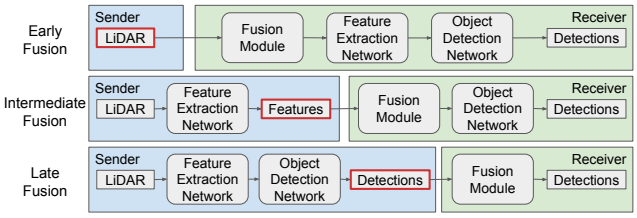

Figure 1: Taxonomy of cooperative perception systems.

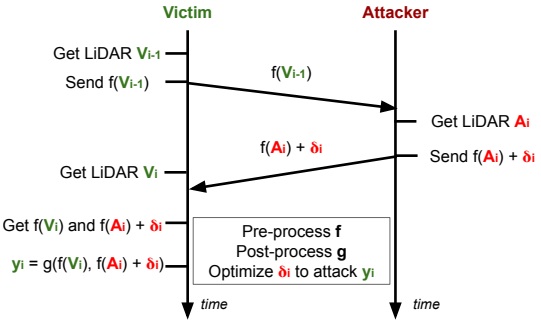

Figure 2: Temporal order of message exchanges.

are against one single autonomous vehicle. Late-fusion collaborative perception shares object locations [36–38, 62] thus the attacker can trivially modify these locations, which is the threat model of many existing studies [26, 27, 45, 56]. Tu *et al.* [70] is the first attack specific to intermediate-fusion collaborative perception, which is an untargeted adversarial attack creating inaccurate detection bounding boxes as many as possible by perturbing feature maps in intermediate-fusion systems. However, the attack is not realistic considering the constraints of real systems, as discussed in §3.3. We propose real-world realizable attacks that challenge both early-fusion and intermediate-fusion systems.

Defenses on CAV perception. As shown in Table 2, existing defense mechanisms are not designed for our proposed attacks thus cannot resolve them effectively. Several Vehicle-to-Everything (V2X) communication standards [15, 17, 18, 20, 42] define security practices of network protocols (*e.g.,* access control, message integrity). They cannot block the data fabrication attacks because the attackers can modify data before wrapping it into protocol messages where the protection is enforced. Trusted Execution Environments (TEEs) [41] can potentially safeguard perception algorithms via secure hardware, but its deployment is difficult and vulnerable to side-channel attacks. Against physical sensor attacks, various anomaly detection methods are proposed [22, 40, 54, 55, 61, 67]. For LiDAR systems especially, CARLO [67] detects abnormal point clouds that violate occlusion features and LIFE [54] detects temporal and sensor-fusion inconsistencies. Above defenses rely on physical rules but attackers in collaborative perception can simulate the physics to craft realistic but malicious data, as discussed in §5.1. For connected vehicle applications, many efforts model the benign behaviors of ego/remote vehicles and detect model outliers as anomalies [26, 27, 45, 56]. The models may involve various aspects including temporal consistency [27], physical constraints on message delivery or vehicle control [26, 45], cross-validation with local sensor [56], etc. However, existing works assume the systems to share simple GPS/OBU data, making it challenging to adapt them effectively for addressing anomalies in complicated LiDAR images or feature maps. We propose joint anomaly detection leveraging the sensing of spatial space from all connected vehicles, which enhances the spatial coverage of effective anomaly detection compared with the previous approaches.

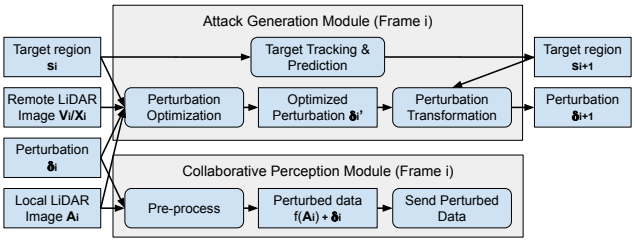

Figure 3: Framework of real-time targeted attacks on collaborative perception.

3 Problem Definition

We define the data fabrication problem in §3.1 and the threat model in §3.2. We emphasize the technical challenges for such new attacks compared with existing attacks in §3.3.

3.1 Formulation

In a scenario where multiple vehicles jointly execute collaborative perception, the attacker aims to spoof or remove road objects (*e.g.,* vehicles, pedestrians) from designated locations in the victim's perception results.

We formulate the problem of data fabrication as an optimization problem. We denote LiDAR data at frame $i \in \mathbb{N}$ from the attacker, the victim, and other benign vehicles by A_i, V_i, and $X_i^{(j)}$, $j \in \{0, 1, \ldots N\}$, respectively. LiDAR data with the same frame index will be merged on the victim side to generate perception results. From Figure 1, we denote preprocess before data sharing as f and post-process after data sharing as g. A normal collaborative perception for the victim on frame i can be described as:

$$y_i = g(f(V_i), f(A_i), f(X_i^0), f(X_i^1), ..., f(X_i^N)). \quad (1)$$

As the attacker can replace $f(A_i)$ by malicious data. For instance, the attacker can append a minor perturbation δ_i to craft malicious data as $f(A_i) + \delta_i$, which will change the original perception result from y_i to y_i':

$$y_i' = g(f(V_i), f(A_i) + \delta_i, f(X_i^0), f(X_i^1), ..., f(X_i^N)). \quad (2)$$

Given a fitness function I evaluating attack success and attack constraints C restricting the perturbation, the attacker solves:

$$\max_{\delta_i} I(y_i') \quad \text{s.t. } C(\delta_i). \quad (3)$$

3.2 Threat Model

We assume that CAVs execute collaborative perception in a Vehicle-to-Vehicle (V2V) scenario. Our results can be easily generalized to vehicle-to-infrastructure (V2I) settings by replacing one or more vehicles with edge computing devices.

We assume the attacker can physically control at least one vehicle participating in collaborative perception. This allows the attacker to gain privileges on the vehicle's software and hardware, enabling them to manipulate the sensors, tamper with the local execution of algorithms, and send arbitrary data through the network. In other words, attackers can directly alter the data to share, *i.e.*,, LiDAR point clouds, feature maps, and bounding boxes in early-fusion, intermediate-fusion, and late-fusion perception schemes, respectively.

We focus on early-fusion and intermediate-fusion collaboration schemes where attackers need to subtly craft complicated structured data. In terms of perception models, as the attackers locally install the perception model for joining the collaborative perception, we assume they have white-box access (i.e., model parameters). Some of our proposed attacks require no model access or only inference API.

Meanwhile, we assume the presence of benign vehicles which the attacker cannot invade. The assumption that the attacker would control all vehicles surrounding a victim vehicle on a busy road is deemed too impractical and financially prohibitive. We do not consider physical sensor attacks such as LiDAR spoofing [43] and GPS spoofing [72]. They are general threats to CAVs while we focus on new vulnerabilities brought by collaborative perception. Besides, the attacker cannot break the cryptographic protection thus cannot compromise the secure communication channels among vehicles.

3.3 Attack Constraints

In addition, the attacks must be realizable on real collaborative perception systems. Though Tu *et al.* [70] proposed a feature-perturbing attack against intermediate-fusion systems, it violates attack constraints as follows.

Sensor physics and definition ranges. We require the attacker to obey basic rules in terms of the data format, otherwise it is trivial to detect the anomalies. The attackers' LiDAR point clouds should have a reasonable distribution of point density and the angle of the lasers should comply with the LiDAR configuration. In addition, the point clouds must present reasonable occlusion effects, in order to bypass anomaly detection methods based on the occlusion features [67]. The attackers' shared intermediate features should be within the definition ranges, avoiding absurd values.

Targeted attacks. The attacker should be able to designate a target region for either spoofing or removal attacks, in order to support delicate creation of hazardous scenarios. Otherwise, the untargeted and uncontrollable attack impact as presented in Tu *et al.* [70] damages attack effectiveness and stealth.

Real-time temporal constraints. Collaborative perception is an asynchronous multi-agent system where each vehicle produces LiDAR images in cycles but is not synchronized in time. Figure 2 illustrates a typical order of events in collaboration perception. To attack the victim's perception at frame i (y_i), the optimization of δ_i has the following constraints:

• *Limited knowledge*. Optimization of δ_i must be finished before the victim's processed LiDAR data V_i is generated. Therefore, attack generation cannot leverage the victim's data on the same frame. Similarly, data from other benign vehicles at frame i may not be available either. The attacker can for sure rely on the shared data in previous frames from all vehicles, provided that the data transmission delay is much smaller than the LiDAR cycle. Tu *et al.* [70] assumes the availability of all data in the frame to attack thus it is impractical.

• *Real-time attack without observable delay*. The optimization of δ_i takes time, especially when the attack involves online adversarial machine learning. To make sure δ_i is produced and transmitted before the fusion stage of the victim, the attacker can either design fast real-time attacks or optimize the perturbation before frame i arrives.

4 Attack Methodology

We present realistic data fabrication attacks against various types of collaborative perception. We first introduce a general framework for real-time targeted attacks in §4.1 and elaborate on the details of ray casting attacks against early-fusion systems (§4.2) and adversarial attacks against intermediate-fusion systems (§4.3). Attackers can trivially send fake bounding boxes in late-fusion systems so we omit the discussion.

4.1 Zero-delay Attack Scheduling

As analyzed in §3, the attacks must be effective to trigger safety hazards while fast enough to satisfy real-time constraints. To satisfy both requirements, we propose an attack framework as shown in Figure 3, whose key idea is to parallelize attack generation and perception processes.

First of all, the attacker can identify the set of vehicles collaborated with the victim vehicle and align frame indices of their shared sensor data based on timestamps. The attack generation module is triggered on each LiDAR cycle. It first tracks the target region: (1) for object spoofing, the trajectory of the object to spoof is predefined; (2) for object removal, the attacker needs a simple object detection algorithm to localize the target object to remove. Then it optimizes the malicious perturbation that can be used to attack the victim's perception at the current frame. Note that the optimized perturbation is generated overtime and cannot be used to attack due to the real-time constraints (§3.3). We need to transform the perturbation into one that has a similar attack impact on the next frame. In this way, the perturbation is ready to apply when the next frame arrives, introducing no additional delay to the original collaborative perception pipeline. As the attack generation occurs one frame in advance, it affords the attacker up to one LiDAR cycle time to complete the optimization.

The optimization and transformation of the perturbation highly depend on the configuration of the collaborative system and will be discussed in later sections.

4.2 Black-box Ray Casting Attack

In early-fusion collaborative systems, CAVs share LiDAR point clouds. Thus, the attacker will perturb the location of

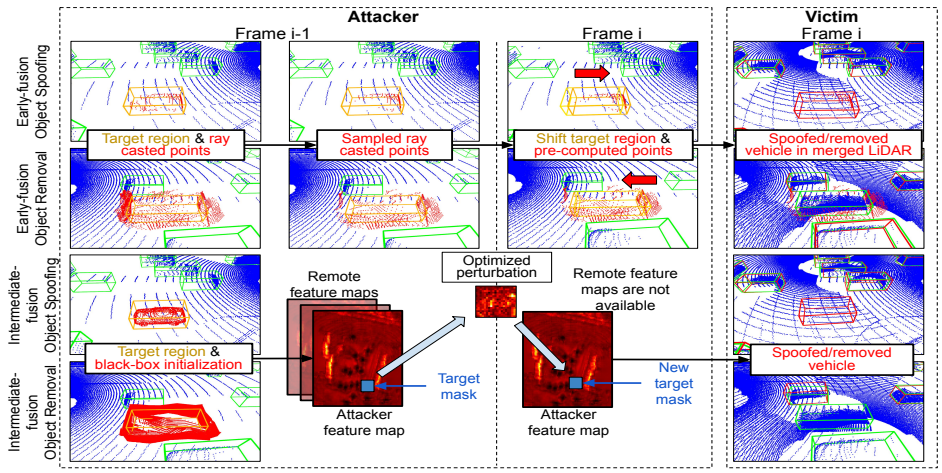

Figure 4: Demonstration of collaborative perception attacks. The orange box is the target region, the green boxes are ground-truth labels, and the red boxes are predicted objects.

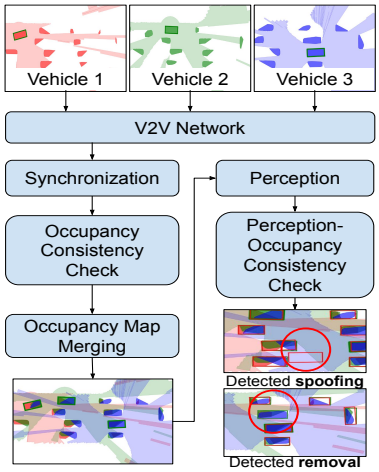

Figure 5: Workflow of CAD's anomaly detection.

LiDAR points directly but must obey the physical rules of LiDAR sensors as mentioned in §3.3. Note that a white-box adversarial attack [30] is not applicable because (1) most perception models involve non-differential pre-processing and (2) even if the gradient can be approximated, the heavy computation can hardly achieve real-time attacks.

Insights. First, we find that a higher point density on the object surface leads to more successful detection. Mainstream 3D object detection models learn spatial features from voxelized point groups (§2). It is therefore natural that a higher point density strengthens the learned feature toward object classes. Second, a higher coverage on object surfaces also contributes to better detection, as the shape features of objects become more explicit. This is also one of the key benefits of collaborative perception, as multi-view LiDAR data allows for a more comprehensive perception of objects. Given the two insights, the object spoofing attack aims to spoof denser LiDAR points of objects and cover a larger surface area of the object. The goal of the object removal attack is to obscure the surface of the original object as thoroughly as possible. We confirm the insights in our ablation study (§6.3.3).

Attack methods The attacker pretends that an object is spoofed or removed and reconstructs the LiDAR point cloud via ray casting techniques. The traced rays follow the physical laws of the original lasers so the reconstructed point cloud is realistic. The spoofing attack requires no model access while the removal attack requires the model's inference API. The attack is demonstrated in Figure 4 and Algorithm 1.

Preparation of 3D object model. The attacker first constructs a 3D model (*e.g.,* a triangle mesh) of the object they wish to fabricate. In later attack steps, we will place the 3D model in the target region and cast malicious points on its surfaces. For object spoofing, the model can represent a real object such as a car. For object removal, we optimize a universal adversarial shape offline as the model. We initialize a cuboid triangle mesh and use a black-box genetic algorithm to optimize the perturbation on mesh vertices. As shown

Algorithm 1: Black-box ray-casting attacks.

Input: A target region y_t, LiDAR image X, an 3D object model S (generated offline using an initial model S_0 and an attack dataset D).
Output: Fake LiDAR image X_a.

Function `BlackboxAttack` $(X,\ y_t,\ S)$ **:**
  $S_t \leftarrow$ `Transform`(S, y_t);
  $X_r \leftarrow$ `NonOcclusionRayCasting`(X, S_t);
  $X_a \leftarrow$ `PointSampling`(X_r, S_t);
Function `AdversarialShape` $(S_0,\ D)$ **:**
  $S \leftarrow S_0$; ▷ Only for object removal.
  for *Iteration* $1 \ldots K$ **do**
    for $X^{(i)}, y_t^{(i)} \in D$ **do**
      $X_a^{(i)} \leftarrow$ `BlackboxAttack`$(X^{(i)}, y_t^{(i)}, S)$;
      $Y^{(i)} \leftarrow$ `Perception`$(X_a^{(i)})$;
      Optimize S by maximizing
      $\sum_{y \in Y^{(i)}} \texttt{IoU}(y, y_t^{(i)}) \cdot log(y_\sigma)$;
        ▷ y_σ is the confidence score.
    end
  end

in Algorithm 1 (`AdversarialShape`), in each iteration, we launch the object removal attack on a dataset of attack cases and optimize the object model to maximize a fitness score representing the success of attacks (i.e., minimizing the confidence of detection proposals in the target region). A detailed explanation is in Appendix A.1.

Non-occlusion ray casting. We set up a ray casting scenario where the 3D model is placed at the designated location and the rays are lasers in the attacker's LiDAR image. Though the predefined object models have fixed sizes, we will dynamically adjust size, location, and orientation of them to fit the target region during the scenario creation (`Transform` in Algorithm 1), making the object models universal for various attack situations. The ray casting algorithm calculates the points of intersection between the rays and the 3D model. To

maximize point density on the target object, the ray casting is customized to ignore occlusion effects, ensuring that each ray is not blocked and goes through model surfaces to leave multiple intersection points.

Point sampling. We resolve the occlusion violations by sampling one intersection point per ray. Specifically, for each ray with one or more intersection points with the 3D model, its original LiDAR point is replaced by one of the intersection points. The selection of intersection points is through customizable weighted random sampling. In our implementation, intersection points closer to benign vehicles have a higher probability of being selected. In this way, spoofed fake points tend to have a higher density close to benign points, increasing the chance of obscuring the original point distribution. Also, the randomness ensures high coverage on object surfaces. More details are presented in Appendix A.2.

Attack transformation. To transform the attack into a future frame, we need to record the modified LiDAR points and corresponding ray angles. When the next frame is produced, the attacker removes points with the same ray angles, transforms recorded points to the new target region, and appends the transformed points. Since two frames have a minor time interval (100 ms), the transformation preserves physical laws.

Time constraint. The attack generation can start when the attacker's LiDAR image is produced. Though *point sampling* requires the locations of remote LiDARs, they can be predicted using simple linear velocity estimation. The ray casting should be done within one LiDAR cycle.

4.3 White-box Online Adversarial Attack

Intermediate-fusion systems require CAVs to exchange feature maps, the intermediate result of neural network processing. Such systems are immune to the black-box ray casting attacks (§4.2) because the presence of benign feature maps will drop the attack success rate significantly, as demonstrated later in our experiments (§6.3.3). Adversarial machine learning, on the other hand, is able to generate adversarial feature maps. The attack assumes that the attacker has white-box knowledge of perception models.

Insights. We optimize a perturbation on the attacker's feature map by performing a backward pass in each LiDAR cycle and reusing the perturbation over frames as an online attack, similar to Tu et. al. [69]. We introduce two new ideas to achieve realistic real-time targeted attacks.

First, we initialize the perturbation using results from black-box ray casting attacks, making the initial perturbation vector closer to the optimal choice. This step is crucial for achieving real-time attacks as it significantly reduces the number of optimization iterations required.

Second, to restrict attack impact to a specific region, we mask the feature map. This is based on the fact that convolution networks preserve the relationship between feature map indices and real-world locations [48]. Another conventional approach to enforce spatial constraints is to add a regularization term to the loss function (*e.g.,* penalize detection errors in non-attack regions). However, this requires multiple iterations to converge, making it unsuitable for real-time attacks.

Attack methods. The attacker optimizes a perturbation on their feature map over continuous frames. For each frame, the attacker spoof/remove objects in the point cloud first as initialization, then updates the latest perturbation map through an iteration of projected gradient descent (PGD). As the target region moves, the perturbation is re-indexed accordingly in each cycle. The key steps are demonstrated in Figure 4 and a detailed algorithm is presented in Appendix A.3.

Black-box initialization. The attacker starts by modifying the raw point cloud. Unlike the ray casting attack in §4.2, there is no restriction on this modification in terms of the physical laws. Therefore, the attacker tends to inject high-density high-coverage LiDAR points representing the 3D models mentioned in §4.2, which can be prepared offline.

Feature map masking. We make the assumption that each feature map index is associated with a voxel/pillar in the 3D real-world coordinate system. Given the target region, we extend the region by a fine-tuned parameter and extract corresponding feature indices. The masking operation ensures that only features with the selected indices are perturbed. If the index mapping is not explicit, it can be approximated by comparing the feature map before and after the black-box initialization and identifying the indices where the feature values have been altered.

Loss objective. The optimization objective is to increase/decrease the score of the bounding box proposal on the labeled attack region, for spoofing/removing objects. We define the objective function as Equation 4, where Z' denotes the set of bounding box proposals after the perturbation, z'_σ is the score associated with the proposal z', and z_t represents the target region to attack. The objective function maximizes/minimizes the confidence score of proposals overlapping with the target.

$$\begin{aligned} l_{spoof}(Z') &= \sum_{z' \in Z'} \text{IoU}(z', z_t) \cdot \log(1 - z'_\sigma) \\ l_{remove}(Z') &= -\sum_{z' \in Z'} \text{IoU}(z', z_t) \cdot \log(1 - z'_\sigma) \end{aligned} \tag{4}$$

Constraints on perturbation. We clip the perturbation by restricting feature values to their normal range, which is measured on a set of non-attack test cases. As feature values do not explicitly deliver spatial semantics that can be used for anomaly detection, there is no need to restrict feature perturbation to minor thresholds.

Attack transformation. Given the centers of target regions between two consecutive frames, one can get corresponding feature map indices (x_0, y_0) and (x_1, y_1) respectively. Then each index (i, j) in the feature map is mapped to $(i - x_0 + x_1, j - y_0 + y_1)$.

Time constraint. The PGD optimization needs feature maps shared from as many vehicles that cooperate with the victim. Assuming all benign vehicles continuously broadcast and process feature maps at a frequency equal to the LiDAR cycle

(T) and the transmission delay is below a threshold t_T, the optimization must be done within $T - 2t_T$.

5 Anomaly Detection

We propose CAD, a Collaborative Anomaly Detection system to mitigate the security threats presented in §4. We enumerate the design challenges in §5.1. In §5.2, we outline our system, followed by the details of key components in §5.3 and §5.4.

5.1 Challenges

As discussed in §2, existing defense mechanisms [23, 54, 67] mainly focus on finding temporal or spatial inconsistencies but they cannot handle attackers who can generating fake data that conform with physics laws. We propose a cross-agent consistency check where all benign vehicles exchange evidence of anomalies to reveal adversarial behaviors jointly. To ensure the effectiveness, robustness, and generality of the proposed method, we have to overcome the following challenges.

Affordable bandwidth and computation cost. Collaborative perception systems must finish a perception cycle within a hard deadline (*e.g.,* 100 ms [52]). Therefore, CAD should only share minimal, essential data to save bandwidth and distribute data processing on different vehicles to minimize latency. Our method only shares small-sized metadata.

Detection of stealthy attacks. As the attacks may inject malicious data into a specific small region in 3D space, fine-grained anomaly detection is required. For instance, spoofing a ghost vehicle affects a region of approximately 10 m^2 while the perception range is over 4,000 m^2. CAD uses fine-grained occupancy maps to precisely reveal abnormal regions.

Robustness to benign errors. LiDAR data captured by different vehicles have slight differences in timestamps [65, 71]. CAD leverages motion estimation and prediction to synchronize occupancy maps. Localization error is another potential source of faults. As nowadays vehicle localization achieves an accuracy of less than 0.1 m [11], CAD can tolerate minor errors with proper threshold parameters.

5.2 System Overview

CAD is a system deployed on CAVs against data fabrication during collaborative perception. As shown in Figure 5, besides the original perception pipeline, CAVs are required to perform anomaly detection tasks in parallel.

When a local LiDAR image is produced, each vehicle generates an occupancy map that labels on-road objects, free-to-drive regions, and invisible regions in the 2D space. Then the occupancy map is broadcast via a V2V wireless network. The occupancy map is represented in fine-grained polygons, balancing precision and transmission overhead. In addition, motion information of on-road objects is attached for synchronizing occupancy maps from different vehicles.

After collecting occupancy maps from other vehicles, each vehicle launches two consistency checks. *Occupancy consistency check* reveals inconsistencies of occupancy maps, *e.g.,* one region identified as free and occupied by two different vehicles indicates that one of the participants is faulty or malicious. Occupancy maps are then merged into one, with inconsistent regions marked as unknown. *Perception-occupancy consistency check* then ensures the results of collaborative perception are consistent with the merged occupancy map - bounding boxes should overlap with occupied regions instead of free regions; on-road occupied regions should be detected in at least one bounding box. Even though attackers can launch strong stealthy attacks and fake occupancy maps, the attack impact is always reflected by perception results and can be revealed as malicious by benign occupancy maps.

5.3 Occupancy Map

The occupancy map generation involves three steps: point segmentation, space segmentation, and motion estimation.

Point segmentation. First, we eliminate less useful background points that are not on the road using HD maps provided by autonomous driving systems [7, 8]. Then, we apply ground fitting algorithms (*e.g.,* RANSAC [35] to detect the ground plane and remove LiDAR points on it. By clustering the remaining points based on point density, we can identify all non-ground objects on the road, with each cluster representing a unique on-road object. The method has been proven to be effective in prior research [34, 74, 81].

Space segmentation. After identifying on-road objects, we generate a fine-grained representation of 2D space occupancy, which classifies the 2D space into three categories: *free*, *occupied*, and *unknown*. (1) Occupied regions are the convex hulls [24] of the object clusters. (2) Free regions represent the region surrounded by only ground points. We evenly divide the 2D space into equal sectors whose vertex is the LiDAR sensor location. The number of sectors can be adjusted for different levels of granularity. In each sector, we measure the distance from the LiDAR to the closest non-ground point and label the region within the distance as a free region. A basic implementation of free regions is described above, while we introduce an optimized implementation in Appendix B.1. (3) The remaining region is classified as unknown due to occlusion or the limited range of LiDAR sensors. Since the accuracy of segmentation and clustering drops as LiDAR points get sparser, in the implementation, we define a 2D space as unknown if its distance to the LiDAR sensor exceeds a threshold (*e.g.,* 50 m). Unlike conventional grid-based occupancy maps [44, 49, 60], our occupancy map divides regions using polygon representation. Our approach offers two advantages over grid representation: (1) polygons can more precisely depict arbitrary shapes; (2) by adjusting the outline smoothing factor, polygon representation provides greater flexibility to strike an optimal balance between precision and size.

Motion estimation. First, each CAV executes a multi-object tracking (MOT) process on object point clusters. Inspired by AB3DMOT [73], a baseline solution of MOT, we assign an affinity score to each object pair between two consecutive

frames. This affinity score indicates the level of similarity considering factors such as distance and point density. Using the scores, MOT algorithms can match the same object across frames. Second, given two point clusters that refer to the same object but on two consecutive frames, we use point cloud registration to derive a transformation matrix between them. Formally, if two object clusters with timestamp t' and t ($t - t' \approx T$ where T is LiDAR cycle time) are denoted as $X_{t'}$ and X_t respectively, the transformation matrix T_t satisfies $X_t = T_t \cdot X_{t'}$. We then standardize the matrix to *motion per time unit* - divide translation and rotation extracted from T_t by the time gap $t - t'$ and reconstruct the matrix as T_e. We define this operation as SCALE: $T_e = \text{SCALE}(T_t, \frac{1}{t-t'})$. T_e represents the latest motion of the specific object and is attached to the corresponding occupied region in the occupancy map. Also, the maps should be transformed into a global coordination system as a consensus of all CAVs.

5.4 Consistency Checks

The processes of consistency checks are triggered simultaneously with the data fusion, involving occupancy map synchronization, occupancy consistency checks, and perception-occupancy consistency checks.

Occupancy map synchronization. After receiving a set of occupancy maps with slightly different timestamps, each vehicle aims to synchronize all maps to the timestamp of the latest local LiDAR image. For each on-road occupied region in each occupancy map (except the local map), we first calculate its time gap to the target timestamp, denoted by Δ_t. We then transform the occupied region by applying the transformation $\text{SCALE}(T_e, \Delta_t)$, where T_e is the corresponding motion per time unit. After moving all occupied regions, we post-process the occupancy map by excluding new occupied regions from the original free regions to resolve conflicts. In this way, all occupancy maps can be directly merged as they have been synchronized spatially and temporally. Formally, we denote synchronized occupancy maps by $M^{(i)} = (S_O^{(i)}, S_F^{(i)})$ where $i \in \{0, 1, \ldots, N\}$ denotes vehicle IDs ($t = 0$ denotes the ego vehicle) and S_O/S_F denotes occupied/free regions.

Occupancy consistency check reveals inconsistencies among synchronized occupancy maps. A region is considered conflicted if it is identified as occupied by one vehicle and free by another. We can define conflicted regions as

$$\varepsilon_{occ} = \bigcup_{i,j \in 0 \ldots N} S_O^{(i)} \cap S_F^{(j)}. \tag{5}$$

Considering the inevitable imperfection of synchronization, in the implementation, CAD will ignore conflict regions whose area is below a threshold (*i.e.,* σ_{occ}). Alerts are raised indicating the uncertain risks on conflicted regions.

Next, each vehicle generates one consistent occupancy map by merging available occupancy maps and dropping conflicted regions. Particularly, the occupancy map produced by the ego vehicle is trusted and retained in the merged map, unless sensors of the ego vehicle is detected as compromised by existing detection of LiDAR spoofing [54, 67]. The new occupancy map $M' = (S'_O, S'_F)$ is generated as:

$$\begin{aligned} S'_O &= S_O^{(0)} \cup \Big(\bigcup_{i=1 \ldots N} S_O^{(i)} - \varepsilon_{occ} - S_F^{(0)} \Big) \\ S'_F &= S_F^{(0)} \cup \Big(\bigcup_{i=1 \ldots N} S_F^{(i)} - \varepsilon_{occ} - S_O^{(0)} \Big) \end{aligned} \tag{6}$$

Perception-occupancy consistency check aims to reveal inconsistencies between the perception results and the merged occupancy map based on two rules. First, free regions should have overlap with predicted object bounding boxes. According to LiDAR sensor physics, objects on the road, if observable, always leave LiDAR points above the ground and should be clustered as occupied regions. This rule can counter object spoofing attacks, as attackers may spoof fake objects in free regions perceived by benign vehicles. Second, occupied regions should be within predicted bounding boxes. Similarly, point clusters on roads are potential obstacles and should be detected to avoid a collision. It serves as a countermeasure against object removal attacks where attackers make real objects undetectable. By checking the two rules, alerts are raised on conflicted regions, similarly filtered by a threshold of area (*i.e.,* σ_{spoof} and σ_{remove}). Formally, if we denote predicted bounding boxes as Y, alerted regions include:

$$\varepsilon_{spoof} = \bigcup_{y \in Y} y \cap S'_F \quad \varepsilon_{remove} = \bigcup_{s'_O \in S'_O} s'_O - Y \tag{7}$$

5.5 Limitations

CAD is a mitigation other than elimination of our proposed attacks. First, CAD cannot work in certain extreme scenarios. The detection could be successful only when at least a benign CAV observes the attacked region. Otherwise, the attacked region is an occluded region for all benign CAVs thus no conflict will appear in Equation 7. Second, CAD detects but may not resolve the anomalies. Though the system may identify the possible attackers via majority voting, it is limited in effectiveness if benign CAVs do not dominate the road.

6 Evaluation

We introduce our dataset creation in §6.1 and the implementation details in §6.2. Then, we present a comprehensive evaluation of the proposed attacks and defenses in §6.3 and §6.4.

6.1 Data Collection

Adv-OPV2V. OPV2V [77] is a benchmark dataset for collaborative perception algorithms, with data collected from a combination of simulators, CARLA [5] and SUMO [12]. We generate Adv-OPV2V from OPV2V, as a benchmark for testing collaborative perception attacks and defenses. We select 300 scenarios for object spoofing and removal attacks respectively. Each scenario features 10 consecutive frames and 3 to 5 CAVs among which one attacker and one victim are designated. Each scenario also has predefined attack targets, such as a trajectory of a ghost vehicle for object spoofing or a trajectory of an existing vehicle for object removal. To ensure

the real-world impact of the attacks, we limit the distance between the victim and the target to less than 30 m.

Adv-MCity. We create a real-world multi-vehicle collaborative perception dataset using testbed MCity [19], which is a real-world mock city for testing CAV applications. On real roads, we deploy 3 Lincoln MKZ vehicles as CAVs, which are equipped with OxTS RT3000v3 GPS, Velodyne VLP-32C LiDAR, and Cohda MK6C OBU as a C-V2X receiver. We also deploy several other vehicles as perception targets. We create 8 attack scenarios that contain potential safety hazards, with 4 for object spoofing and 4 for object removal. We collect LiDAR, GPS, and C-V2X network traces from all CAVs to allow for emulation of collaborative perception.

6.2 Implementation

Collaborative perception models. For Adv-OPV2V, we utilize pre-trained models provided by OPV2V, which employ naive point cloud merging in early-fusion and attentive learning in intermediate-fusion. In early-fusion methods, the point clouds are naively concatenated together. In intermediate-fusion methods, the fusion is defined by the models. For Adv-MCity, we augment the OPV2V training data to approximate the LiDAR images collected in the testbed MCity and fine-tune the pre-trained models. During the training of models, an uniform noise of at most 0.2 m or 0.2° is injected to vehicle locations or rotations respectively, in order to better tolerate localization/synchronization errors in real scenarios, following the previous work [57, 71, 76].

Attacks are implemented in 4,874 lines of code (LOC) in Python. The adversarial shape generation is based on a classic genetic algorithm with a population size of 10 and for 5 generations. Adversarial attacks are based on Torch. We fine-tune the learning rate to 1 and optimize for a maximum of 25 iterations. The perturbation of feature maps is restricted in a 5 m×5 m square centered by the target location.

Anomaly detection is implemented in 1,629 LOC in Python, which uses polygon operations in shapely and implementation of RANSAC and DBSCAN from Open3D. The system parameters (*i.e.*, $\sigma_{occ}, \sigma_{spoof}, \sigma_{remove}$) are not fixed but evaluated through the receiver operating characteristic (ROC) curve.

In-vehicle execution environment. To demonstrate system deployment on real vehicles, we implement a collaborative perception framework based on Robot Operating System (ROS), consisting of 3,154 LOC in C++ responsible for V2V communication and basic sensor data processing. Our implementation of attacks and anomaly detection can be plugged into the framework as ROS nodes. For performance measurement, we use an in-vehicle machine with an Intel Xeon Silver 4110 CPU and an Nvidia RTX 2080 Ti GPU.

6.3 Evaluation of Attacks

We present our attack results in §6.3.1. We further analyze the impacting factors in attacks (§6.3.2) and present an ablation study (§6.3.3). We realize attacks in the testbed MCity, evaluate the overhead (§6.3.4) and conduct case studies (§6.3.5).

Table 3: Performance of attacks and defenses on Adv-OPV2V.

Attack setting: Method-Fusion-Goal	Attack results				Defense results		
	Succ.	IoU	Score	ΔAP	Succ.	TPR	FPR
[70]-Int.-Spoof	21.7%	0.01	0.06	-62.8%	100%	34.0%	10.3%
[70]-Int.-Remove	14.0%	0.47	0.34	-61.8%	100%	39.7%	7.6%
RC-Early-Spoof	86.0%	0.55	0.38	-0.4%	83.8%	80.9%	2.0%
RC-Early-Remove	87.3%	0.07	0.03	-0.5%	81.2%	38.0%	5.6%
Adv.-Int.-Spoof	90.0%	0.46	0.71	-2.0%	83.4%	80.1%	2.0%
Adv.-Int.-Remove	99.3%	0.02	0.01	-3.9%	83.6%	42.5%	2.2%
Naive-Late-Spoof	98.7%	0.96	0.99	0	80.8%	84.8%	2.7%
Naive-Late-Remove	0.3%	0.78	0.53	0	-	-	-

Notes: *Int.* - intermediate-fusion. *RC* - ray casting. *Adv.* - adversarial attack. *Succ.* - success rate.

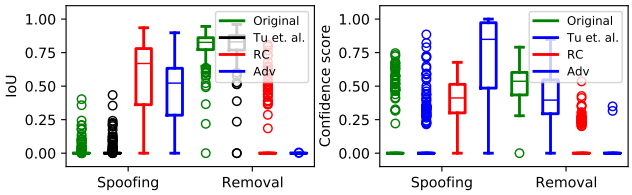

Figure 6: IoU/confidence on target region under the prior attack, our ray casting (RC) and adversarial (Adv) attacks.

6.3.1 Attack Results

To evaluate attack effectiveness, we launch each proposed attack on 300 attack scenarios in Adv-OPV2V against baseline perception models using PointPillars [48] as the backbone. Attack results are listed in Table 3. In each attack scenario, we identify the best predicted bounding box having the largest Intersection over Union (IoU) with the target region. A spoofing attack is considered successful if the IoU is greater than zero while a removal attack is considered successful if the IoU is zero. For late-fusion systems, object spoofing is trivial to reach almost 100% success rate while object removal is hard as long as one benign vehicle observes the object. Our proposed attacks against early/intermediate-fusion are generally successful with a success rate above 86%.

In addition, we illustrate the change of IoU and confidence score on target regions in Figure 6. We observe that attacks make a significant change in the two metrics. For spoofing attacks, the early-fusion ray casting attack achieves a larger IoU meaning more accurate spoofed bounding boxes while the intermediate-fusion adversarial attack pushes the confidence score to extremely high (> 0.8). The result indicates that attacker is easier to launch sophisticated attacks against early-fusion systems since attackers can directly manipulate the subtle spatial features - LiDAR points. The intermediate-fusion system enforces fewer constraints on malicious perturbation thus the upper bound of the attack impact is higher. The change of Average Precision (ΔAP) on non-attack regions is minor, which means all attacks focus on perturbing the tar-

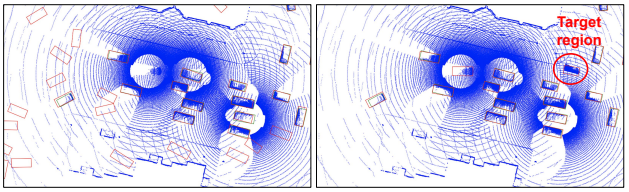

Figure 7: Different stealth of untargeted/targeted attacks.

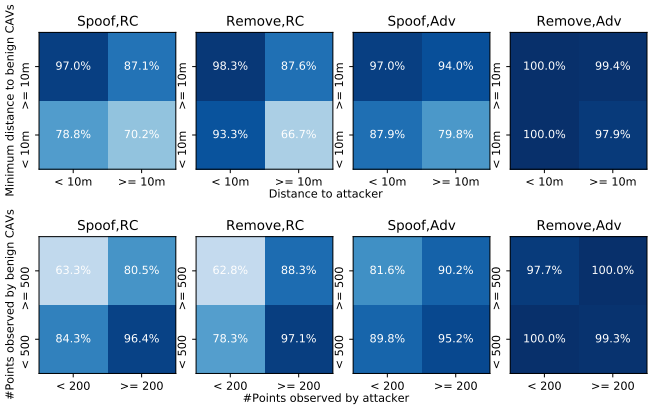

Figure 8: Attack success rate w.r.t. target visibility.

get region. However, ΔAP of intermediate-fusion attacks is higher because the perturbation on feature maps inevitably propagates to a larger region through convolution layers.

We also reproduced the prior attack proposed by Tu *et al.* [70] We use the loss function and attack parameters from the paper and the constraint of perturbation the same as our adversarial attack. We also allow the unrealistic attack constraints as discussed in §3.2. Attack results on Adv-OPV2V are shown in Table 3 and Figure 6. The prior attack is successful in its attack goal to inject as many false perception bounding boxes as possible, by injecting on average 56.2 FPs and 3.6 FNs in each LiDAR frame. The overwhelming FPs drop AP to nearly 1%. However, when attacking a certain region, it only yields 14%-22% success rate because the attack is untargeted fundamentally. The major problem of the untargeted approach is stealth of attacks. The untargeted attack generates a significant number of abnormal bounding boxes that are out of the road or heading away from the lane direction, as shown in Figure 7. The uncontrollable attack impact can be easily recognized by either humans or automatic anomaly detection.

6.3.2 Impacting Factors

Visibility of the target region. We hypothesize that the attack is more successful when the target region is clearly visible to the target but not benign CAVs. Intuitively, the target is more visible if it is closer to the LiDAR or there are more LiDAR points on it. To validate the hypothesis, we draw relationship between attack success rate and the two metrics in Figure 8. The result shows that the attack is more successful when the attacker is closer to the target while benign CAVs are further away, or the attacker has more LiDAR points in the target region while benign vehicles have fewer. The impact of the visibility is obvious in early-fusion systems but not intermediate-fusion systems. The difference is reasonable because for early-fusion schemes, more LiDAR rays interact with closer targets thus attackers can manipulate more LiDAR points without violating LiDAR sensor physics.

Benign errors. It is worth nothing that attacks should tolerate errors in real systems. To simulate the worst-case synchronization errors, we delay any LiDAR frame by 100ms at a probability of 0.5. To simulate localization errors, we incorporate uniform noise into vehicle locations (0 – 0.2 m) and orientations (0 – 0.2°), following existing works [11, 76]. Network errors can manifest as delays, corruptions, or dropped messages, all of which hinder the proper sharing of data. If the attacker's malicious data fails to reach the victim, the attack on that frame will certainly fail. Conversely, if benign vehicles' data cannot reach the attacker, less data is used for attack optimization, potentially leading to less successful attacks. To simulate such scenarios, we randomly drop 10% of data sharing during the attacks. The 10% error rate is regarded as the highest threshold of acceptable network connection by previous studies [68, 86].

From the results in Figure 9, synchronization and localization errors have very minor impact on the attacks. Network errors decrease the success rate by 10-20%, with a 10% reduction attributable to the fact that 10% of the attacker's messages fail to reach the victim. Even with the barely acceptable network connection, our attacks can achieve at least 60% success rate, showing the robustness against benign errors.

Model configuration. Our attacks are general for various collaborative perception models. In Figure 10, the attack success rate is stable if (1) replacing the backbone model by VoxelNet [87] in either early-fusion or intermediate-fusion methods; (2) changing the fusion network of intermediate-fusion system to V2VNet [71] or CoBEVT [75]. However, FPV-RCNN [80] involves a second-stage non-differential fusion on bounding box proposals (similar to late-fusion), making object removal hard.

Object types. We generalize our attacks from vehicle targets to pedestrians and cyclists. As OPV2V [77] only has vehicles originally, we augment OPV2V to include pedestrians and cyclists by modifying the simulation settings and re-training the models. As shown in Figure 11, the attacks are generally effective for different object types. Especially, removing pedestrians is easier than removing other object types because they usually comprise a small number of LiDAR points and have a low detection confidence.

Number of attackers. One attacker is strong enough to break collaborative perception. Adding another attacker can further increase the success rate of ray casting attacks and adversarial attacks by around 5% and 2%, respectively.

6.3.3 Ablation Study

For each attack we propose, we provide a set of variants by removing one or more components from the original design. Attack results are summarized in Figure 12 and the complete quantitative results are in Appendix D.

For ray casting attacks against early-fusion systems, we design the following variants. (1) *RC*. Baseline ray casting pretending the object to spoof/remove emerged/disappeared. Especially for object removal, *RC* uses the adversarial shape while *NoAS-RC* does not. (2) *Dense-A-RC*. Based on *Naive-*

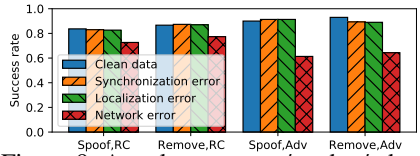

Figure 9: Attack success w/ and w/o benign errors.

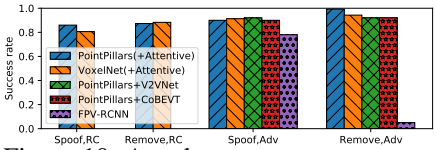

Figure 10: Attack success w.r.t. perception model configurations.

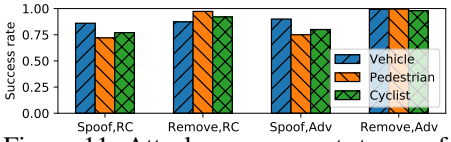

Figure 11: Attack success w.r.t. types of target objects.

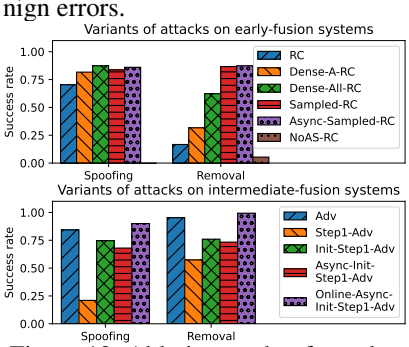

Figure 12: Ablation study of attacks.

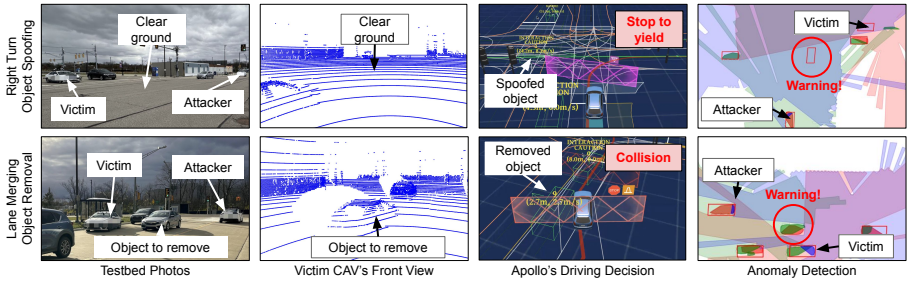

Figure 13: Real-world experiments of attacks and anomaly detection, involving early/intermediate-fusion attacks on two scenarios.

RC, make the spoofed points denser by placing the origin of rays only 5-meter away from the target during ray casting. (3) *Dense-All-RC*. More than *Dense-A-RC*, add multiple virtual LiDARs around the target to further increase point coverage. (4) *Sampled-RC*. Based on *RC*, do non-occlusion ray casting and point sampling as mentioned in §4.2. (5) *Async-Sampled-RC*. The proposed attack. Based on *Sampled-RC*, optimization is done one frame before the attack happens.

The attack results validate our assumptions and prove our design components are useful. *Point density* and *Point coverage* lead to stronger attacks. *Dense-A-RC*'s success rate is 12%/15% higher than *RC* for spoofing/removal. *Dense-All-RC*'s success rate is 6%/31% higher than *Dense-A-RC* for spoofing/removal. However, *Dense-A-RC* and *Dense-All-RC* are not stealthy attacks as spoofed points have abnormal density. Therefore, we propose *Sampled-RC*, whose success rate is 14%/50% higher than the naive ray casting while preserving LiDAR's physical laws. Finally, our asynchronous attack scheduling makes *Async-Sampled-RC* deployable in real-time systems, without a significant drop in success rate. In addition, the universal adversarial shape is crucial for object removal. Naively replacing object points with ground points only achieves a 5% success rate and the usage of adversarial shapes raises the number to 17%.

For adversarial attacks against intermediate-fusion systems, we design the following variants. (1) *Adv*. Basic implementation of PGD. It does not constrain the attacker's knowledge or number of optimization steps and disables black-box initialization. Instead of using perturbation masking, we add a regularization term to achieve a targeted attack. (2) *Step1-Adv*. Based on *Adv*, do optimization for only one iteration. Parameters are set the same as §6.2. (3) *Init-Step1-Adv*. Based on *Step1-Adv*, add black-box initialization. (4) *Async-Init-Step1-Adv*. Based on *Init-Step1-Adv*, optimization is done one frame before the attack happens. (5) *Online-Async-Init-Step1-Adv*. Our proposed attack (§4.3). Online attack optimizing one perturbation vector over consecutive frames.

We conclude the effectiveness of our key designs. *Adv* is a standard white-box adversarial attack with the minimum constraints and maximum resources, representing the empirical upper bound of attack impact. Limiting optimization iteration to only one per frame, though significantly lower computation cost, drops attack success rate by 63%/48% for spoofing/removal. To address the problem, we propose black-box initialization. The design is very useful, especially for object spoofing: *Init-Step1-Adv* achieves 53%/8% higher attack success rate than *Step1-Adv*. Finally, *Async-Init-Step1-Adv* integrates the zero-delay attack scheduling without dropping attack effectiveness and *Online-Async-Init-Step1-Adv* builds an online attack pipeline which further enhances the attacks.

6.3.4 Overhead

We measure the execution latency of our attack algorithms in the in-vehicle execution environment. For ray casting attacks, 3D object model preparation is done offline. The non-occlusion ray casting takes 54 ms on average. Our implementation of ray casting is CPU-only and can be further improved by hardware acceleration. The point sampling takes only <3 ms. Attack transformation introduces a negligible overhead of <1 ms. For adversarial attacks, the point cluster for black-box initialization is prepared offline thus the initialization simply appends pre-computed points to the LiDAR image, incurring a negligible overhead of <1 ms. The one-step PGD optimization is computationally intensive and requires GPU resources, taking 67 ms on average. The total attack generation is finished in 89 ms on average within one LiDAR cycle. The cost of attack transformation is negligiable.

6.3.5 Real-world Case Study

Attacks must be realizable. We test attack algorithms by emulating driving scenarios using dataset Adv-MCity. In this section, we focus on case studies on two scenarios, as shown in Figure 13. All scenarios are described in Appendix C.

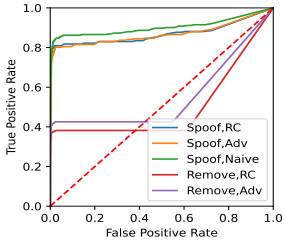

Figure 14: ROC curve of anomaly detection.

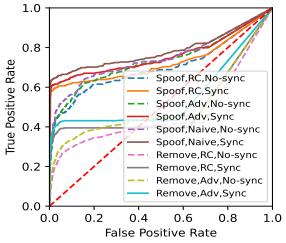

Figure 15: ROC curve w/ and w/o synchronization.

Object spoofing during right turn. The victim CAV is turning right at green while the attacker CAV stops on another road. The attacker's goal is to spoof one fake vehicle to stop the victim, forming a denial-of-service (DoS). First, we launch ray casting attack assuming CAVs use early-fusion collaborative perception. Since the victim is far away from the attacker (>30 m), it is hard to directly spoof an object in front of the victim. However, the attacker can leverage the traffic rule implemented in CAVs by spoofing a moving vehicle whose trajectory blocks the victim's path. In 5 seconds, the attack succeeds to spoof the vehicle in 76% of frames. Baidu Apollo indeed stops the vehicle to yield the spoofed vehicle. Second, if CAVs use an intermediate-fusion system, the adversarial attack can achieve a stronger attack by spoofing an obstacle right in front of the victim in 92% of frames.

Object removal during lane merging. The victim CAV is starting from a parking place to merge into the main road while another vehicle is going through from behind. Normally, the victim should yield the right of way. The attacker sits on another lane, aiming to remove the moving vehicle from the view of the victim. The ray casting attack succeeds in removing the vehicle in the first 45 frames but fails in the last 5 frames because the target is further. Nevertheless, it is too late when the victim perceives the target and Baidu Apollo reports a collision. Also, using the white-box adversarial attack against intermediate-fusion perception has a similar attack impact, removing the vehicle in 96% of frames.

6.4 Evaluation of Anomaly Detection

We evaluate effectiveness and efficiency in §6.4.1 and §6.4.3. We then compare CAD with existing defenses in §6.4.4. We demonstrate the real-world deployment in §6.4.5.

6.4.1 Defense Results

We apply CAD on attacked frames in Adv-OPV2V. Note that CAD is supposed to detect both attacks and perception faults, as long as the predicted bounding box has no overlap with ground-truth or the ground-truth bounding box is not detected. We consider adaptive attacks, where the attacker fakes his/her occupancy map to avoid conflicts with other occupancy maps or detected bounding boxes. Therefore, *occupancy consistency check* in §5 cannot defend adaptive attacks but serves as input validation before merging occupancy maps.

In Table 3, true positive rate (TPR) and false positive rate (FPR) are calculated on the whole LiDAR images, including the detection of both malicious attacks and benign perception faults. On the other hand, success rate measures the detection of only malicious attacks, which is the ratio of positive detection on the target region and the total number of attack scenarios. We also show the ROC curves in Figure 14. CAD is generally effective against various attack methods. If selecting thresholds σ_{spoof} and σ_{remove} to maximize AUC score, CAD achieves FPR <3% and TPR >80%/38% against spoofing/removal while detecting around 90% of anomalies caused by our attacks. From the split-down of alarms in Figure 16, low TPR against removal threats is mainly caused by undetected benign perception faults which are out of the range of occupancy maps. The "false alarms" are mostly the cases where predicted bounding boxes is not accurate (IoU < 0.5). Though considered as normal cases by our criteria, they have significant differences with accurate object detection.

We also apply CAD on the prior attack [70]. As the prior attack is untargeted and injects a few dozens of fake detection results in each LiDAR image, it is easy for CAD to reveal 100% of the attacked frames, as shown in Table 3. The low TPR (around 30%) is because the occupancy maps cannot cover lots of the far-away fake detection results.

Other adaptive attacks. The attacker may exploit *occupancy consistency check* to create as many conflicts as possible to minimize the coverage of the merged occupancy map and decrease TPR. However, if the occupancy conflict is with the victim's local map, the attacker is directly identified because the local data is trusted. This ensures a lower bound of TPR by using only local occupancy maps (71.7%/24.0%/70.4%/28.7%/78.1% against the attacks in Table 3). Also, occupancy conflicts obviously indicate the existence of attackers and are useful messages for other defense mechanisms such as reputation systems.

The attacker may also choose to launch attacks at locations out of the coverage of occupancy maps. However, our experiments on Adv-OPV2V show that benign occupancy maps cover 95.6% in 30 meters and 99.9% in 10 meters around the victim. It is very little chance for the attacker to spoof/remove objects stealthily at a safety-critical distance.

6.4.2 Impacting Factors

Distance to LiDAR sensors. As shown in Figure 17, over 80% false alarms are 60 meters away from any benign vehicles. Within the range of occupancy maps (50 meters in our configuration), CAD can stably make true detection.

Synchronization. With injected synchronization errors as introduced in §6.3.2, CAD's synchronization provides significant robustness. As shown in Figure 15, CAD is not effective without synchronization, having TPR 35%/15% against spoofing/removal when FPR is low (<5%). With synchronization, CAD achieves TPR 60%/40% against spoofing/removal, close to the detection rate on ideal synchronized data.

Localization errors. With the injected localization errors as stated in §6.3.2, we observe a minor decrease of accuracy

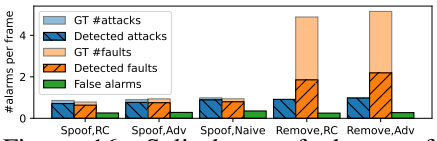

Figure 16: Split-down of alarms of anomaly detection.

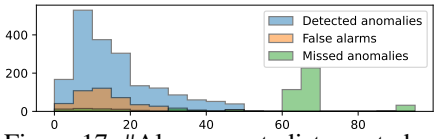

Figure 17: #Alarms w.r.t. distance to benign LiDARs.

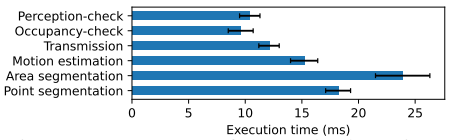

Figure 18: Latency of anomaly detection.

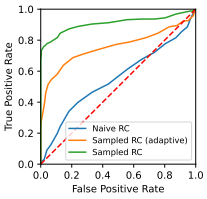

Figure 19: ROC curve of CARLO.

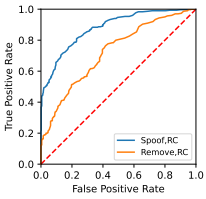

Figure 20: ROC curve of LIFE.

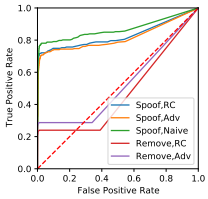

Figure 21: ROC curve of MDS.

(TPR -3.1%, FPR +0.2%), showing CAD's robustness.

Object types. As CAD uses the area of conflicted regions as the key metric, smaller object sizes result in higher FPR, e.g., minor conflicts caused by errors of occupancy maps may be falsely considered as anomalies. By choosing the best AUC score of the ROC curve, CAD detects pedestrian spoofing, pedestrian removal, cyclist spoofing, and cyclist removal in FPR/TPR of 78.4%/14.5%, 38.2%/13.9%, 81.7%/6.5%, and 29.4%/6.2%, respectively. When compared with the detection of fake vehicles, CAD yields around 12%/4% higher FPR on pedestrians/cyclists while maintains TPR stable.

Number of attackers. More attackers decrease the coverage of benign occupancy maps and cause more false negatives. Adding another attack decreases TPR by 5% on average.

6.4.3 Overhead

We measure the latency of the anomaly detection using the in-vehicle execution environment and recorded network trace. Segmentation/clustering algorithms are relatively expensive but they can be further boosted using hardware acceleration [28]. Occupancy map transmission is as fast as 10ms. Each map contains around 300-1000 polygon vertices and lightweight metadata of object motion, in a small size of around 10 KB. Consistency checks are simple polygon operations that can be finished in 15ms. The end-to-end anomaly detection takes 92ms, which means the CAV can be aware of abnormal bounding boxes before the LiDAR cycle ends.

6.4.4 Comparison with Other Defense Approaches

CARLO [67] is an anomaly detection algorithm detecting LiDAR spoofing attacks. Given the fact that detected bounding boxes should host solid objects, CARLO validates that the volume of "free space" (conical spaces between the LiDAR sensor and rendered points) in bounding boxes is under a threshold. In collaborative perception, we assume the victim CAV applies CARLO on each received LiDAR point cloud.

Results are shown in Figure 19, which detects our proposed ray casting attack (Sampled RC) with TPR 77.7% and FPR 3.9%. However, attackers can adjust the rule of point sampling in §4.2 to launch adaptive attacks. For instance, the attacker can restrict the number of points that can penetrate object surfaces to be <30% (Sampled RC adaptive). As a result, TPR decreases to 63.8% and FPR increases to 14.7% while the success rate only drops by 5.6%. If forbid ray penetration completely (Naive RC), CARLO is close to random guessing while the success rate of our attack is still above 70%. In contrast, CAD achieves higher TPR and, more importantly, is independent of attack methods.

LIFE [54] is a hybrid anomaly detection system against sensor attacks. First, it checks the temporal consistency of depth camera images based on machine learning methods. As discussed in §5.1, attackers have the capability to continuously launch attacks thus the check is fundamentally not useful. Besides, an object matching algorithm checks the consistency between objects detected in the camera and the LiDAR. In early-fusion systems, CAVs can launch object matching on remote LiDAR and local camera images. To reproduce LIFE, we use the same LiDAR segmentation as CAD and train a EfficientPS [58] model for camera image segmentation.

We draw the ROC curve of object matching in Figure 20. LIFE's object matching achieves around 80% TPR and 26% FPR against early-fusion ray casting attacks. LIFE suffers from a higher FPR because multiple machine learning processes introduce more errors - inaccurate detection from either camera or LiDAR, which is usual on far-away objects, may trigger a false alarm. Compared with LIFE, CAD has a higher detection rate with much lower computation/bandwidth consumption, thanks to the collaboration among CAVs.

MDS [23] is an anomaly detection framework assuming CAVs to share bounding boxes. Besides checks on message format and temporal consistency which are not relevant to our attacks, each CAV evaluates the consistency between the local occupancy map and final perception results, and also merges anomaly detection results from multiple CAVs by majority voting. However, the attackers can launch adaptive attacks to only send falsified data to the specific victim instead of all other CAVs, thus the majority voting is actually not helpful. Compared with CAD, the spatial check is restricted on the local occupancy map (without the sharing of occupancy maps) thus TPR is lower by 9-15%, as shown in Figure 21.

CAD has no conflicts with the above defenses. Users can deploy multiple defenses to strengthen sensor data integrity.

6.4.5 Real-world Case Study

We demonstrate CAD on the same attack scenarios discussed in §6.3.5, shown in Figure 13. In the scenario of the right turn, though the spoofed object is in the blind spot of the

victim (red), another benign vehicle (blue) observes that region and identifies the anomaly when it is 15 meters away from the victim. In the scenario of lane merging, the victim vehicle recognizes an object point cluster on the left but is not detected by the perception system, triggering a warning of object removal 2.1 seconds before a potential collision. In other scenarios in Appendix C, our anomaly detection can detect attacks at least 1.5 seconds before a collision or hard brake happens. The anomaly detection can be more robust when there are more benign vehicles on busy roads.

7 Conclusion

In this work, we pioneer to examine the threats posed by data fabrication on collaborative perception systems. We unleash novel attacks that successfully spoof or remove on-road objects in various types of collaborative perception schemes and demonstrate the attack impact on real traffic scenarios. To mitigate the threats, we introduce a cross-vehicle validation solution powered by fine-grained occupancy maps, which detects anomalies seconds before potential road hazards occur. Our attempts of both attacks and defenses serve as a benchmark to spur future research on collaborative perception security.

References

[1] 3GPP Release 14. https://www.3gpp.org/specifications-technologies/releases/release-14, 2017.

[2] Qualcomm C-V2X. https://www.qualcomm.com/news/releases/2017/09/qualcomm-announces-groundbreaking-cellular-v2x-solution-support-automotive, 2017.

[3] Huawei C-V2X. https://carrier.huawei.com/en/products/wireless-network-v3/Components/c-v2x, 2019.

[4] Infineon C-V2X. https://www.infineon.com/dgdl/Infineon-ISPN-Use-Case-Savari-Securing-V2X+communications-ABR-v01_00-EN.pdf?fileId=5546d462689a790c0168e1c1f5e35221, 2019.

[5] Carla: Open-source simulator for autonomous driving research. https://carla.org/, 2021.

[6] Autopilot | Tesla. https://www.tesla.com/autopilot, 2022.

[7] Autoware: Open-source software for self-driving vehicles. https://github.com/Autoware-AI, 2022.

[8] Baidu Apollo. http://apollo.auto, 2022.

[9] Bosch C-V2X. https://www.bosch-mobility-solutions.com/en/solutions/connectivity/v2x-connectivity-solutions-cv/, 2022.

[10] Honda Global | Automated Drive. https://global.honda/innovation/automated-drive/detail.html, 2022.

[11] OxTS - the inertial navigation (INS) experts since 1998. https://www.oxts.com, 2022.

[12] SUMO: Simulation of Urban Mobility. https://www.eclipse.org/sumo/, 2022.

[13] Waymo. https://waymo.com/, 2022.

[14] Automotive Edge Computing Consortium. https://aecc.org/, 2023.

[15] Details of 'dts/its-00167' work item. https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=46541, 2023.

[16] Ford AV Dataset. https://aecc.org/, 2023.

[17] Ieee sa - ieee 1609.2-2022. https://standards.ieee.org/ieee/1609.2/10258/, 2023.

[18] J3224_202208: V2x sensor-sharing for cooperative and automated driving. https://www.sae.org/standards/content/j3224_202208/, 2023.

[19] Mcity - University of Michigan. https://mcity.umich.edu/, 2023.

[20] Tr 103 562 - v2.1.1 - intelligent transport systems (its). https://www.etsi.org/deliver/etsi_tr/103500_103599/103562/02.01.01_60/tr_103562v020101p.pdf, 2023.

[21] Simegnew Yihunie Alaba and John E Ball. A survey on deep-learning-based lidar 3d object detection for autonomous driving. *Sensors*, 22(24):9577, 2022.

[22] Khattab M Ali Alheeti, Abdulkareem Alzahrani, and Duaa Al Dosary. Lidar spoofing attack detection in autonomous vehicles. In *2022 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–2. IEEE, 2022.

[23] Moreno Ambrosin, Lily L Yang, Xiruo Liu, Manoj R Sastry, and Ignacio J Alvarez. Design of a misbehavior detection system for objects based shared perception v2x applications. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1165–1172. IEEE, 2019.

[24] David Avis and David Bremner. How good are convex hull algorithms? In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 20–28, 1995.

[25] Mir Ali Rezazadeh Baee, Leonie Simpson, Ernest Foo, and Josef Pieprzyk. Broadcast authentication in latency-critical applications: On the efficiency of ieee 1609.2. *IEEE Transactions on Vehicular Technology*, 68(12):11577–11587, 2019.

[26] Srivalli Boddupalli, Ashwini Hegde, and Sandip Ray. Replace: Real-time security assurance in vehicular platoons against v2v attacks. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 1179–1185. IEEE, 2021.

[27] Srivalli Boddupalli and Sandip Ray. Redem: Real-time detection and mitigation of communication attacks in connected autonomous vehicle applications. In *Internet of Things. A Confluence of Many Disciplines: Second IFIP International Cross-Domain Conference, IFIPIoT 2019, Tampa, FL, USA, October 31–November 1, 2019, Revised Selected Papers 2*, pages 105–122. Springer, 2020.

[28] José Canilho, Mário Véstias, and Horácio Neto. Multi-core for k-means clustering on fpga. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4. IEEE, 2016.

[29] 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. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 176–194. IEEE, 2021.

[30] 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 *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 2267–2281, 2019.

[31] Qi Chen, Xu Ma, Sihai Tang, Jingda Guo, Qing Yang, and Song Fu. F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pages 88–100, 2019.

[32] Qi Chen, Sihai Tang, Qing Yang, and Song Fu. Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 514–524. IEEE, 2019.

[33] Jiaxun Cui, Hang Qiu, Dian Chen, Peter Stone, and Yuke Zhu. Coopernaut: End-to-end driving with cooperative perception for networked vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17252–17262, 2022.

[34] Bertrand Douillard, James Underwood, Noah Kuntz, Vsevolod Vlaskine, Alastair Quadros, Peter Morton, and Alon Frenkel. On the segmentation of 3d lidar point clouds. In *2011 IEEE International Conference on Robotics and Automation*, pages 2798–2805. IEEE, 2011.

[35] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[36] Jorge Godoy, Víctor Jiménez, Antonio Artuñedo, and Jorge Villagra. A grid-based framework for collective perception in autonomous vehicles. *Sensors*, 21(3):744, 2021.

[37] Hendrik-Jörn Günther, Björn Mennenga, Oliver Trauer, Raphael Riebl, and Lars Wolf. Realizing collective perception in a vehicle. In *2016 IEEE Vehicular Networking Conference (VNC)*, pages 1–8. IEEE, 2016.

[38] Mohamed Hadded, Pierre Merdrignac, Sacha Duhamel, and Oyunchimeg Shagdar. Security attacks impact for collective perception based roadside assistance: A study of a highway on-ramp merging case. In *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pages 1284–1289. IEEE, 2020.

[39] R Spencer Hallyburton, Yupei Liu, Yulong Cao, Z Morley Mao, and Miroslav Pajic. Security analysis of {Camera-LiDAR} fusion against {Black-Box} attacks on autonomous vehicles. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1903–1920, 2022.

[40] Zhongyuan Hau, Soteris Demetriou, Luis Muñoz-González, and Emil C Lupu. Shadow-catcher: Looking into shadows to detect ghost objects in autonomous vehicle 3d sensing. In *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I 26*, pages 691–711. Springer, 2021.

[41] Shengtuo Hu, Qi Alfred Chen, Jiwon Joung, Can Carlak, Yiheng Feng, Z Morley Mao, and Henry X Liu. Cvshield: Guarding sensor data in connected vehicle with trusted execution environment. In *Proceedings of the Second ACM Workshop on Automotive and Aerial Vehicle Security*, pages 1–4, 2020.

[42] Shengtuo Hu, Qingzhao Zhang, André Weimerskirch, and Z Morley Mao. Gatekeeper: A gateway-based broadcast authentication protocol for the in-vehicle ethernet. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 494–507, 2022.

[43] Zizhi Jin, Ji Xiaoyu, Yushi Cheng, Bo Yang, Chen Yan, and Wenyuan Xu. Pla-lidar: Physical laser attacks against lidar-based 3d object detection in autonomous vehicle. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 710–727. IEEE Computer Society, 2022.

[44] ByeoungDo Kim, Chang Mook Kang, Jaekyum Kim, Seung Hi Lee, Chung Choo Chung, and Jun Won Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 399–404. IEEE, 2017.

[45] Hyogon Kim and Taeho Kim. Vehicle-to-vehicle (v2v) message content plausibility check for platoons through low-power beaconing. *Sensors*, 19(24):5493, 2019.

[46] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3d proposal generation and object detection from view aggregation. *IROS*, 2018.

[47] Swarun Kumar, Lixin Shi, Nabeel Ahmed, Stephanie Gil, Dina Katabi, and Daniela Rus. Carspeak: a content-centric network for autonomous driving. *ACM SIGCOMM Computer Communication Review*, 42(4):259–270, 2012.

[48] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[49] Hao Li, Manabu Tsukada, Fawzi Nashashibi, and Michel Parent. Multivehicle cooperative local mapping: A methodology based on occupancy grid map merging. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2089–2100, 2014.

[50] Yiming Li, Congcong Wen, Felix Juefei-Xu, and Chen Feng. Fooling lidar perception via adversarial trajectory perturbation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7898–7907, 2021.

[51] You Li and Javier Ibanez-Guzman. Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems. *IEEE Signal Processing Magazine*, 37(4):50–61, 2020.

[52] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E Haque, Lingjia Tang, and Jason Mars. The architectural implications of autonomous driving: Constraints and acceleration. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 751–766, 2018.

[53] Hansi Liu, Pengfei Ren, Shubham Jain, Mohannad Murad, Marco Gruteser, and Fan Bai. Fusioneye: Perception sharing for connected vehicles and its bandwidth-accuracy trade-offs. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2019.

[54] Jinshan Liu and Jung-Min Park. "seeing is not always believing": Detecting perception error attacks against autonomous vehicles. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2209–2223, 2021.

[55] Shinan Liu, Xiang Cheng, Hanchao Yang, Yuanchao Shu, Xiaoran Weng, Ping Guo, Kexiong Curtis Zeng, Gang Wang, and Yaling Yang. Stars can tell: A robust method to defend against gps spoofing attacks using off-the-shelf chipset. In *USENIX Security Symposium*, pages 3935–3952, 2021.

[56] Xiruo Liu, Lily Yang, Ignacio Alvarez, Kathiravetpillai Sivanesan, Arvind Merwaday, Fabian Oboril, Cornelius Buerkle, Manoj Sastry, and Leonardo Gomes Baltar. Miso-v: Misbehavior detection for collective perception services in vehicular communications. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 369–376. IEEE, 2021.

[57] Yifan Lu, Quanhao Li, Baoan Liu, Mehrdad Dianat, Chen Feng, Siheng Chen, and Yanfeng Wang. Robust collaborative 3d object detection in presence of pose errors. *arXiv preprint arXiv:2211.07214*, 2022.

[58] Rohit Mohan and Abhinav Valada. Efficientps: Efficient panoptic segmentation. *International Journal of Computer Vision*, 129(5):1551–1579, 2021.

[59] Minh Pham and Kaiqi Xiong. A survey on security attacks and defense techniques for connected and autonomous vehicles. *Computers & Security*, 109:102269, 2021.

[60] Hang Qiu, Pohan Huang, Namo Asavisanu, Xiaochen Liu, Konstantinos Psounis, and Ramesh Govindan. Autocast: Scalable infrastructure-less cooperative perception for distributed collaborative driving. *arXiv preprint arXiv:2112.14947*, 2021.

[61] Aanjhan Ranganathan, Hildur Ólafsdóttir, and Srdjan Capkun. Spree: A spoofing resistant gps receiver. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pages 348–360, 2016.

[62] Florian A Schiegg, Daniel Bischoff, Johannes R Krost, and Ignacio Llatser. Analytical performance evaluation of the collective perception service in ieee 802.11 p networks. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2020.

[63] 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 *Proceedings of the 29th USENIX Conference on Security Symposium*, pages 931–948, 2020.

[64] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019.

[65] Shuyao Shi, Jiahe Cui, Zhehao Jiang, Zhenyu Yan, Guoliang Xing, Jianwei Niu, and Zhenchao Ouyang. Vips: real-time perception fusion for infrastructure-assisted autonomous driving. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 133–146, 2022.

[66] Zhiying Song, Fuxi Wen, Hailiang Zhang, and Jun Li. An efficient and robust object-level cooperative perception framework for connected and automated driving. *arXiv preprint arXiv:2210.06289*, 2022.

[67] Jiachen Sun, Yulong Cao, Qi Alfred Chen, and Z Morley Mao. Towards robust {LiDAR-based} perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 877–894, 2020.

[68] Behrad Toghi, Md Saifuddin, Hossein Nourkhiz Mahjoub, Muhammad Ozair Mughal, Yaser P Fallah, Jayanthi Rao, and Sushanta Das. Multiple access in cellular v2x: Performance analysis in highly congested vehicular networks. In *2018 IEEE Vehicular Networking Conference (VNC)*, pages 1–8. IEEE, 2018.

[69] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. Physically realizable adversarial examples for lidar object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13716–13725, 2020.

[70] James Tu, Tsunhsuan Wang, Jingkang Wang, Sivabalan Manivasagam, Mengye Ren, and Raquel Urtasun. Adversarial attacks on multi-agent communication. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7768–7777, 2021.

[71] Tsun-Hsuan Wang, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, and Raquel Urtasun. V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. In *European Conference on Computer Vision*, pages 605–621. Springer, 2020.

[72] Jon S Warner and Roger G Johnston. Gps spoofing countermeasures. *Homeland Security Journal*, 25(2):19–27, 2003.

[73] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10359–10366. IEEE, 2020.

[74] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018.

[75] Runsheng Xu, Zhengzhong Tu, Hao Xiang, Wei Shao, Bolei Zhou, and Jiaqi Ma. Cobevt: Cooperative bird's eye view semantic segmentation with sparse transformers. *arXiv preprint arXiv:2207.02202*, 2022.

[76] Runsheng Xu, Hao Xiang, Zhengzhong Tu, Xin Xia, Ming-Hsuan Yang, and Jiaqi Ma. V2x-vit: Vehicle-to-everything cooperative perception with vision transformer. In *European Conference on Computer Vision*, pages 107–124. Springer, 2022.

[77] Runsheng Xu, Hao Xiang, Xin Xia, Xu Han, Jinlong Li, and Jiaqi Ma. Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2583–2589. IEEE, 2022.

[78] Takahito Yoshizawa and Bart Preneel. Survey of security aspect of v2x standards and related issues. In *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 1–5. IEEE, 2019.

[79] Haibao Yu, Yizhen Luo, Mao Shu, Yiyi Huo, Zebang Yang, Yifeng Shi, Zhenglong Guo, Hanyu Li, Xing Hu, Jirui Yuan, et al. Dair-v2x: A large-scale dataset for vehicle-infrastructure cooperative 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21361–21370, 2022.

[80] Yunshuang Yuan, Hao Cheng, and Monika Sester. Keypoints-based deep feature fusion for cooperative vehicle detection of autonomous driving. *IEEE Robotics and Automation Letters*, 7(2):3054–3061, 2022.

[81] Dimitris Zermas, Izzat Izzat, and Nikolaos Papanikolopoulos. Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5067–5073. IEEE, 2017.

[82] Qingzhao Zhang, Shengtuo Hu, Jiachen Sun, Qi Alfred Chen, and Z Morley Mao. On adversarial robustness of trajectory prediction for autonomous vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15159–15168, 2022.

[83] Rusheng Zhang, Zhengxia Zou, Shengyin Shen, and Henry X Liu. Design, implementation, and evaluation of a roadside cooperative perception system. *Transportation research record*, 2676(11):273–284, 2022.

[84] Xumiao Zhang, Anlan Zhang, Jiachen Sun, Xiao Zhu, Y Ethan Guo, Feng Qian, and Z Morley Mao. Emp: edge-assisted multi-vehicle perception. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 545–558, 2021.

[85] Chunheng Zhao, Jasprit Singh Gill, Pierluigi Pisu, and Gurcan Comert. Detection of false data injection attack in connected and automated vehicles via cloud-based sandboxing. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):9078–9088, 2021.

[86] Liang Zhao, Hongmei Chai, Yuan Han, Keping Yu, and Shahid Mumtaz. A collaborative v2x data correction method for road safety. *IEEE Transactions on Reliability*, 71(2):951–962, 2022.

[87] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

[88] 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 *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1945–1960, 2021.

A Attack Algorithms

We introduce more implementation details of attack algorithms proposed in §4.

A.1 Universal Adversarial Shape

As mentioned in §4.2, the universal adversarial shape is for early-fusion object removal attacks. It is not effective to pretend the object does not exist and simply replace object points with ground points, as proved by our ablation study in §6.3.3. As a solution, we find that spoofing an adversarial shape covering the original object to remove is surprisingly useful. As the generation of adversarial shapes involves time-consuming optimization, we aim to generate a universal adversarial shape that can be pre-computed offline and is general for different scenarios.

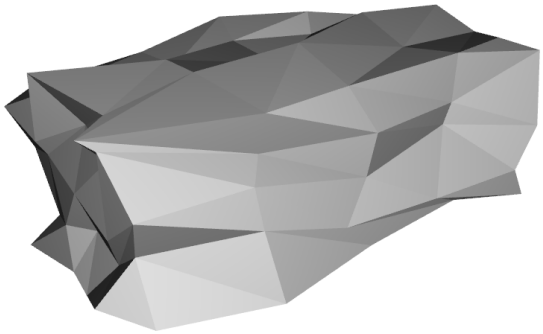

Figure 22: Example of universal adversarial shape.

As shown in Algorithm 2, the generation algorithm is based on black-box genetic optimization. In our implementation, we create the initial shape as a cuboid and randomly select 50 diverse object removal attack scenarios as X. For the genetic algorithm, the initial population is a set of random perturbations on the initial shape and other parameters are set as mentioned in §6.2. For each perturbed shape, we launch the attack defined in §4.2 on each attack case and calculate an average fitness score as Equation 8:

$$F(y) = \sum_{y \in Y} \text{IoU}(y, y_t) \cdot log(1 - y_\sigma), \tag{8}$$

where $y \in Y$ denotes predicted bounding boxes, y_σ denotes confidence scores, and y_t denotes the target region. The black-box optimization minimizes Intersection over Union (IoU) on the target region and confidence score simultaneously. Note that we will scale and transform the shape to a bounding box that has the same center as the target region but is 0.6 m (fine-tuned by experiments) larger on all dimensions, making sure the adversarial shape covers the original object edges. The remaining optimization steps are handled by the genetic algorithm itself. Figure 22 shows the universal adversarial shape we used for attacking Adv-OPV2V.

A.2 Point Sampling

The point sampling resolves occlusion conflicts after ray casting and meanwhile maximizes attack effectiveness, as shown in Algorithm 3. Each ray after non-occlusion ray casting has a set of intersection points with the target 3D model and the point sampling algorithm assigns each intersection point with a probability, which is larger when the points are closer to benign LiDARs, as mentioned in §4.2. The location of LiDARs is broadcast in collaborative perception, which is necessary to merge multi-source LiDAR images. Also, there is a parameter tuning how much rays can penetrate object surfaces. The fewer rays that can penetrate surfaces, the more natural the spoofed point clouds are, as discussed in §6.4.4.

Algorithm 2: Universal adversarial shape generation in details.

Input: An initial 3D shape S_0, a set of attack cases X.
Output: A universal adversarial shape S_{adv}.

Initialize a set of perturbation on vertices of S_{in}: P;
Initialize genetic algorithm instance: GENETIC(P);
for *Iteration* $1 \ldots K$ **do**
  for $p_i \in P$ **do**
    for $x_j \in X$ **do**
      $x'_j \leftarrow \text{REMOVALATTACK}(x_j, S_0 + p_i)$;
      $Y'_i \leftarrow \text{PERCEPTION}(x'_j)$;
      $I_i \leftarrow I_i - \sum_{y \in Y'_i} \text{IoU}(y, y_t^{(j)}) \cdot log(y_\sigma)$;
    end
  end
  $S_{adv} \leftarrow S_0 + \text{BESTSOLUTION}(P, I)$
  $P \leftarrow \text{GENETICUPDATE}(P, I)$;
end
Return S_{adv};

Algorithm 3: Point Sampling.

Input: A mapping from one ray to a set of points intersected with 3D model M, LiDAR poses of benign vehicles L, a parameter of the probability of penetration $0 \leq \sigma_p \leq 1$.
Output: A mapping from one ray to one intersection point M'.

for $r, X \in M$ **do**
  if $\text{RANDOM}() < \sigma_p$ **then**
    $D \leftarrow \text{COMPUTEDISTANCE}(X, L)$;
    $P \leftarrow \text{PROBABILITY}(D)$;
    $M'(r) \leftarrow \text{RANDOMCHOICE}(X, P)$;
  else
    $M'(r) \leftarrow \text{CLOSEST}(X)$;
  end
end
Return M';

A.3 Generation of Adversarial Feature Map

We use a standard PGD approach with target masking. We present the adversarial optimization in one iteration in Algorithm 4, excluding black-box initialization and attack transformation as introduced in §4.3. We first generate proposals using the perception model with the original masked perturbation on the attacker's feature map and then calculate the loss as defined in Equation 4. Finally, we do a backward gradient calculation and leverage the Adam optimizer to perform a one-time update on the perturbation.

B Defense Algorithms

We introduce more implementation details of anomaly detection algorithms proposed in §5.

Algorithm 4: Feature-level PGD optimization.

Input: Perturbation vector p, feature maps of benign CAVs X, the attacker's feature map x_a, target mask μ, learning rate r, and constraints of perturbation C_p.
Output: Updated perturbation vector p'.

$Z \leftarrow \text{MODEL}(x_a + p \cdot \mu, X)$;
$l \leftarrow \text{LOSS}(Z, \cdot)$;
$g \leftarrow \text{GRADIENT}(l, p)$;
$p' \leftarrow \text{CLIP}(y - r \cdot g, C_p)$;
Return p';

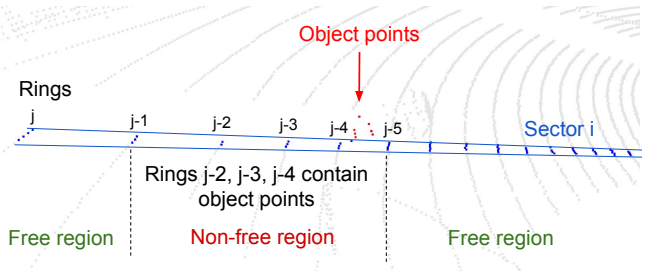

Figure 23: Example of free space identification.

Algorithm 5: Segmentation of free regions.

Input: Point cloud X, rings R, sectors E, object points $X_{obj} \subseteq X$, ground points $X_{grd} \subseteq X$.
Output: Polygons S_F as 2D free regions.

$S_F \leftarrow \emptyset$;
$F \leftarrow \emptyset$ as the set of ground point groups;
for $e_i \in E$ **do**
  for $r_j \in R$ **do**
    $X_{i,j} \leftarrow \text{GETPOINTS}(e_i, r_j)$;
    if $X_{i,j} \cap X_{obj} = \emptyset$ *and* $X_{i,j} \subseteq X_{grd}$ **then**
      Push $X_{i,j}$ to F;
    end
  end
  while *Find* $r_j, r_{j+1}, \ldots, r_{j+k} \in F$ **do**
    Get P as a polygon in sector e_i and rings r_j, r_{j+k};
    if *Cannot ind* $x \in X_{obj}$ *in* P **then**
      Push P to S_F;
    end
  end
end
Return S_F;

B.1 Space Segmentation

We provide implementation details for identifying free regions to the best precision.

LiDAR sensors equip vertically placed laser transmitters. In each LiDAR cycle, the LiDAR sensor rotates itself thus the lasers scan the object surfaces. Therefore, points in the LiDAR image can be grouped according to which laser they belong to, called "rings". Inner rings are closer to the LiDAR sensor, and vice versa. Also, we can group LiDAR points by splitting the bird-eye view 2D space into sectors whose vertex is the location of the LiDAR sensor. In this way, each LiDAR point has a ring ID and a sector ID, as shown in Figure 23.

After grouping LiDAR points into rings and sectors, we execute Algorithm 5 to label free regions. First of all, we identify point groups that entirely consist of ground points. Then, we search in each sector i for consecutive rings from j to $j+k$ which are all ground point groups and generate a polygon covering a segment of sector j between the furthest point of ring i and the closest point of ring $j+k$. For now, we ensure that this region is not occluded by other objects, otherwise at least one point in these rings is an object point. Next, we double check no object point falls in the polygon, in case there are objects above the ground. If no object points are found, we can finally label the polygon as a free region.

Our free space segmentation is precise in two folds. (1) The grouping of LiDAR points follows the physical laws of LiDAR sensors thus each group has a similar number of points, making the segmentation stable even in far-away regions. (2) The identification of free regions is conservative by taking occlusion and floating objects into account.

Note that the size of sectors and rings is configurable for a good trade-off between precision and data size. In our implementation, we split the 2D space into 360 sectors, each for 1°. The number of rings depends on the number of lasers the LiDAR sensor has, for instance, LiDARs in Adv-OPV2V have 64 lasers while LiDARs in Dataset-MCity have 32 lasers.

C Testbed MCity

Figure 24 depicts the 8 attack scenarios we created in Testbed MCity. We show the topology of on-road vehicles as well as expected attack impacts. We create 4 traffic scenes: (1) a right turn on green in an intersection, (2) an unprotected left turn in an intersection, (3) an unprotected right turn in a T intersection, and (4) a lane merging from a parking place. Each scene contains one attacker CAV, one victim CAV, one benign CAV, and 1-2 regular vehicles. In each traffic scene, we craft one object spoofing attack case and one object removal attack case. Object spoofing intends to spoof a vehicle to the view of the victim. It aims to stop the victim from moving for as long time as possible, forming a denial of service. Object removal aims to remove an on-road vehicle, which is important for driving safety, from the victim's perception results. Therefore, object removal tends to trigger severe safety hazards such as collisions. We recorded 5-second LiDAR/GPS packets and network traces for each scenario to emulate attacks.

In experiments, we launch both black-box ray casting attacks (§4.2) and white-box adversarial attacks (§4.3) on 8 scenarios, assuming the CAVs may host either early-fusion or intermediate-fusion collaborative perception. All attacks impact (e.g., stop or collision) are successfully produced. Among the 400 frames of perception (5 seconds of 10 Hz LiDAR images for 8 scenarios), the ray casting attacks succeed in object spoofing/removal in 87%/79% of frames while the adversarial

attacks succeed in object spoofing/removal in 92%/95% of frames. Adversarial attacks in general have stronger attack effects as their success rate is not restricted by the distance between targets and the attacker.

We also launch an anomaly detection CAD on attacked data, achieving 85.3% TPR and 2.6% FPR for all scenarios in total. The alarms are useful to avoid safety hazards as they are delivered 1.5 seconds on average before collisions or brakes happen.

D Results of Attack Variants

In Table 4, We list the quantified attack impact of all attack variants mentioned in our ablation study 6.3.3, including evaluation metrics in §6.3.1: success rate, IoU, confidence score, and ΔAP. In addition, we present success rates on targets with various distances to the attacker, including close range (<10m), medium range (10-20m), and far-away (>20m). All results in Table 4 are run on OPV2V's pre-trained models, using PointPillars as the backbone.

Similar to §6.3.3, the following abbreviation words stand for various design choices.

- *RC*. Ray casting algorithm to reconstruct point clouds.
- *Adv*. Adversarial machine learning to optimize attacks.
- *Async*. Use the data in frame $i-1$ to attack frame i.
- *AS*. Universal adversarial shape generated offline, used in ray casting.
- *Dense-A*. During ray casting, increase the point density of the point cloud in the target region, by moving the origin of LiDAR rays closer to the target.
- *Dense-All*. During ray casting, increase the point density of the point cloud by pretending to have multiple LiDARs from different angles.
- *Sampled*. First perform non-occlusion ray casting and then sample points to resolve occlusion conflicts.
- *Step1*. During adversarial attacks, do one-step PGD instead of unlimited iterations in each LiDAR cycle.
- *Init*. During adversarial attacks, initialize the point cloud using a dense point cluster generated offline.

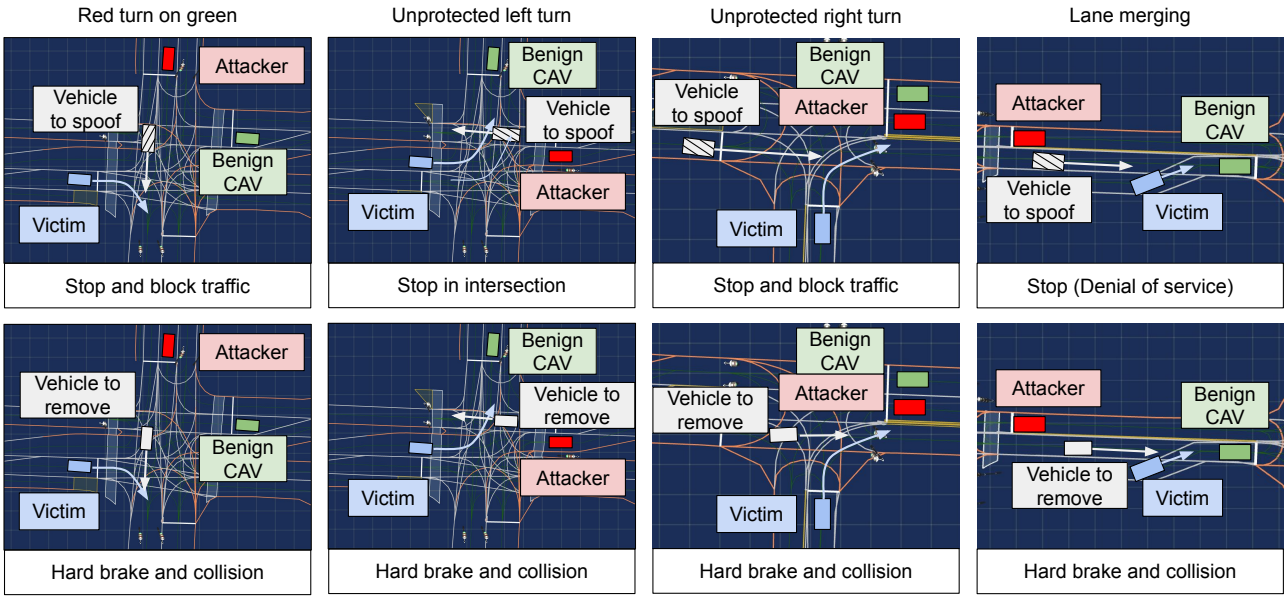

Figure 24: 8 attack scenarios in Dataset-MCity.

Table 4: Attack variants evaluated on Adv-OPV2V.

Fusion	Goal	Method	Success				IoU		Score		ΔAP
			Average	Close	Mid	Far	Before	After	Before	After	
Early	Spoof	RC	70.3%	81.0%	64.1%	67.3%	0.01	0.39	0.08	0.29	-0.39%
Early	Spoof	RC,Dense-A	81.7%	87.0%	78.6%	80.0%	0.01	0.46	0.08	0.35	-0.44%
Early	Spoof	RC,Dense-All	87.3%	91.0%	86.9%	81.8%	0.01	0.54	0.08	0.39	-0.48%
Early	Spoof	RC,Sampled	83.7%	91.0%	80.0%	80.0%	0.01	0.53	0.08	0.37	-0.39%
Early	Spoof	RC,Sampled,Async	86.0%	94.0%	84.1%	76.4%	0.01	0.54	0.08	0.38	-0.42%
Early	Remove	RC	5.3%	6.7%	7.7%	1.1%	0.80	0.73	0.53	0.45	-0.09%
Early	Remove	RC,Dense-A	6.3%	8.0%	7.7%	3.2%	0.80	0.70	0.53	0.44	-0.05%
Early	Remove	RC,Dense-All	9.7%	12.0%	10.8%	6.3%	0.80	0.66	0.53	0.41	-0.05%
Early	Remove	RC,AS	6.7%	6.7%	7.7%	5.3%	0.80	0.53	0.53	0.38	-0.08%
Early	Remove	RC,AS,Dense-A	21.7%	34.7%	20.8%	12.6%	0.80	0.50	0.53	0.31	-0.05%
Early	Remove	RC,AS,Dense-All	62.3%	80.0%	56.2%	56.8%	0.80	0.25	0.53	0.15	-0.09%
Early	Remove	RC,AS,Sampled	86.7%	97.3%	90.0%	73.7%	0.80	0.07	0.53	0.04	-0.48%
Early	Remove	RC,AS,Sampled,Async	87.3%	96.0%	93.1%	72.6%	0.80	0.07	0.53	0.03	-0.49%
Intermediate	Spoof	RC	36.3%	44.0%	29.7%	40.0%	0.01	0.13	0.11	0.18	-0.14%
Intermediate	Spoof	RC,Dense-A	36.7%	44.0%	29.0%	43.6%	0.01	0.14	0.11	0.18	-0.11%
Intermediate	Spoof	RC,Dense-All	42.7%	53.0%	35.2%	43.6%	0.01	0.18	0.11	0.21	-0.15%
Intermediate	Spoof	Adv	60.3%	65.0%	51.0%	76.4%	0.01	0.29	0.11	0.56	-3.60%
Intermediate	Spoof	Adv,Step1	21.0%	22.0%	15.2%	34.5%	0.01	0.02	0.11	0.12	-0.84%
Intermediate	Spoof	Adv,Step1,Async	26.3%	33.0%	17.2%	38.2%	0.01	0.05	0.11	0.14	-0.85%
Intermediate	Spoof	Adv,Step1,Async,Online	56.7%	61.0%	53.1%	58.2%	0.01	0.23	0.11	0.42	-5.36%
Intermediate	Spoof	Adv,Step1,Init	74.7%	78.0%	70.3%	80.0%	0.01	0.43	0.11	0.48	-0.73%
Intermediate	Spoof	Adv,Step1,Async,Init	68.0%	72.0%	64.1%	70.9%	0.01	0.33	0.11	0.41	-0.99%
Intermediate	Spoof	Adv,Step1,Async,Online,Init	90.0%	94.0%	88.3%	87.3%	0.01	0.46	0.11	0.70	-2.01%
Intermediate	Remove	RC,AS	3.7%	2.7%	4.6%	3.2%	0.81	0.74	0.63	0.56	-0.12%
Intermediate	Remove	RC,AS,Dense-A	6.0%	8.0%	5.4%	5.3%	0.81	0.69	0.63	0.52	-0.05%
Intermediate	Remove	RC,AS,Dense-All	9.7%	13.3%	12.3%	3.2%	0.81	0.70	0.63	0.51	-0.04%
Intermediate	Remove	Adv	78.7%	74.7%	73.1%	89.5%	0.81	0.12	0.63	0.07	-0.07%
Intermediate	Remove	Adv,Step1	57.3%	62.7%	60.8%	48.4%	0.81	0.24	0.63	0.18	-1.38%
Intermediate	Remove	Adv,Step1,Async	56.0%	52.0%	64.6%	47.4%	0.81	0.30	0.63	0.19	-1.45%
Intermediate	Remove	Adv,Step1,Async,Online	54.0%	60.0%	51.5%	52.6%	0.81	0.22	0.63	0.17	-1.38%
Intermediate	Remove	Adv,Step1,Init	55.0%	65.3%	56.2%	45.3%	0.81	0.28	0.63	0.17	-1.25%
Intermediate	Remove	Adv,Step1,Async,Init	57.7%	68.0%	56.2%	51.6%	0.81	0.27	0.63	0.17	-2.63%
Intermediate	Remove	Adv,Step1,Async,Online,Init	93.0%	89.3%	92.3%	96.8%	0.81	0.02	0.63	0.01	-3.90%
Late	Spoof	Naive	98.7%	99.0%	99.3%	96.4%	0.01	0.96	0.08	0.99	0
Late	Remove	Naive	0.3%	0.0%	0.8%	0.0%	0.80	0.77	0.59	0.53	0