GhostLite: Data Minimization with Applications to Real-Time LiDAR Attacks

Richard Capraru[1,2], Emil C. Lupu[3], Jian-Gang Wang[2] and Boon Hee Soong[1]

***Abstract*— LiDAR-based object detection plays a crucial role in autonomous driving, yet remains vulnerable to ghost object attacks, where adversarially crafted point clouds trigger false detections. Traditional attack methods often require a large number of points and substantial computational resources, making them inapplicable in real-world scenarios. Although prior research has explored reducing the number of required attack points, real-world feasibility remains a challenge due to the high computational cost and attack generation time. While LiDAR object detection can be compromised given enough adversarially crafted points, we believe there are some critical points, which we call reduced attack budget, which are sufficient to attack the detector. Motivated by this consideration, we propose a novel geometric- and heuristic-based approach capable of generating effective ghost object attacks by leveraging the target's contour characteristics. By strategically selecting a minimal set of adversarial points, as few as 20 points, our approach maintains high detection confidence, reduces attack execution time and attack budget by up to 4 and 73 times, respectively, in this paper. Our method has been verified on both simulated and real-world datasets (KITTI). The experimental results demonstrate that our optimized attack significantly reduces computational overhead and consequently improves the attack's real-time feasibility.**

***Index Terms*— Autonomous Vehicles, Lidar Object Detection, Security For Intelligent Vehicles**

I. Introduction

Autonomous vehicles (AVs) rely on LiDAR-based object detection for accurate environmental perception. Although LiDAR object detection works well under normal conditions, adversarial attacks, particularly ghost object attacks, are a significant security risk. By injecting carefully crafted points in the perceived 3D point cloud, the detection models are deceived into detecting non-existent objects. Such attacks can cause AVs to react to false hazards, leading to unnecessary braking and loss of control.

Recent studies have amply demonstrated the vulnerability of 3D object detectors to LiDAR spoofing attacks [1]–[5]. Petit et al. [2] introduced an attack that misled detectors into perceiving objects at locations beyond the attacker's actual position. Shin et al. [3] extended this approach by manipulating 10 points in a point cloud at a closer distance to the spoofer. Later, Cao et al. [4], Sun et al. [5], and Jin et al. [6] extended these attacks, injecting up to 100, 200, and 4,200 points, respectively. In summary, prior work shows that it is possible to inject a large number of points into the point cloud and demonstrates this in laboratory conditions. However, injecting a large number of points in a real environment remains challenging.

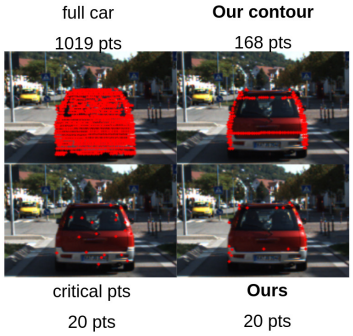


Fig. 1: Visualization of point selection for ghost car attacks. Top left: Full attack budget (all points). Bottom left: RainReaper's [7] reduced budget using critical points. Top right: Our method identifies key contour points. Bottom right: We select a minimal set of critical points, achieving comparable attack effectiveness with fewer points and faster generation.

One fundamental challenge is the need to synchronize the injected points with the LiDAR sensor scanning process in real-time. A typical automotive LiDAR operates at 10Hz to 20Hz i.e. a 3D scan is completed in 50–100ms. In order to successfully create a ghost object, the crafted points must be injected at some specific spatial locations within each scan window. However, when the target vehicle moves at varying speeds (e.g., 50km/h to 120km/h on highways), the relative position between the vehicle and attacker will change, making it extremely difficult to maintain consistent deception (an example for this case is shown in Fig. 3).

Another major challenge is the computational overhead of generating adversarial point clouds in real-time. Previous methods, such as [6] and [7], optimize the attack budget by reducing the number of injected points. However, multiple iterations are needed to craft adversarial points, and current methods require several hundred milliseconds to generate an adversarial point cloud. This makes them impractical in real-world applications.

In this paper, we introduce an optimized real-time ghost object attack (Fig. 1) that significantly reduces both the point budget of the attack and the attack execution time. Our method (illustrated in Fig. 2) strategically selects as few as 20 adversarial points, reducing the attack budget from thousands, whilst maintaining a good attack success rate. Moreover, our approach speeds up attack execution by up to four times. This shows for the first time that ghost object attacks are feasible in practical settings and can become a danger on our roads. We summarize our main contributions:

- **New Method.** We propose a novel geometric and heuristic approach that enables real-time attacks while

[1] School of Electrical and Electronic Engineering, NTU, Singapore 639798
[2] Institute for Infocomm Research, A*STAR, Singapore 138632
[3] Department of Computer Science, Imperial College London, UK SW7 2AZ

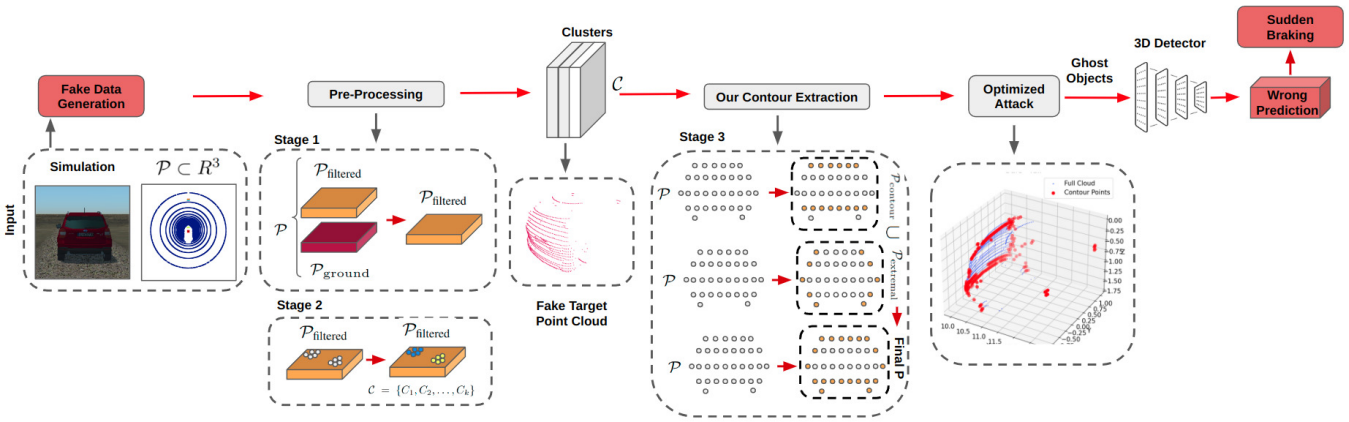


Fig. 2: **Overview of Our Approach.** We begin with a real or simulated point cloud $\mathcal{P} \subset \mathbb{R}^3$. **Stage 1: Ground Plane Removal.** We remove the ground points $\mathcal{P}_{\text{ground}}$, retaining the filtered point cloud $\mathcal{P}_{\text{filtered}}$, which contains only the relevant targets. **Stage 2: Grid-Based Clustering.** The filtered point cloud $\mathcal{P}_{\text{filtered}}$ is partitioned into clusters $\mathcal{C}$. **Stage 3: Down sampling.** Each cluster is processed using our Geometry-Inspired Contour Extraction method to obtain the contour points $\mathcal{P}_{\text{contour}}$ and extremal points $\mathcal{P}_{\text{extremal}}$. These are combined to form the final optimized point set $\mathcal{P}_{\text{inj}}$. This set is used to detect the victim target and the objects in the scene and to construct a reduced-budget ghost object by reusing points from existing targets. When injected into a 3D detector, the ghost object is falsely detected as a genuine target, potentially leading to dangerous outcomes such as sudden braking.

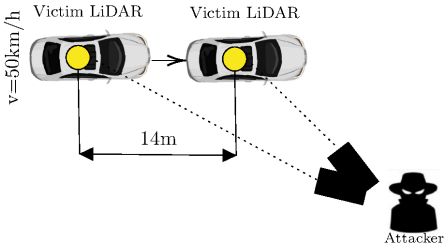


Fig. 3: Illustration of practical attacks. Within 1s, a car that moves at 50km/h advances 14 meters. In practice, the moving speed can vary and the attacker needs to know the precise location of a victim in order to execute the attack.

maintaining a high success rate against object detectors. This is currently infeasible with existing state-of-the-art (SOTA) attacks.

- **Detection Time and Data quantity optimization.** Our approach reduces attack generation time by up to four times and minimizes the required point cloud data by up to 73 times, significantly improving the feasibility.
- **New Knowledge.** To the best of our knowledge, we are the first to study and quantify the attacker's temporal constraints, showing their importance in achieving real-time adversarial attacks.
- **Better Understanding of SOTA Detectors.** By leveraging geometric characteristics, our method provides new insights into the detection mechanisms of black-box SOTA models.
- **Comprehensive Study.** We conduct an extensive evaluation of existing SOTA detectors by applying various down sampling strategies to test their robustness against spoofing attacks.

The structure of the paper is as follows: Section II provides an analysis of the temporal constraints that an attacker must satisfy to be successful. Section III outlines the problem formulation and methodology, followed by Section IV, which details our experimental setup, dataset, data processing, and evaluation metrics. The performance of our method is then verified using both real and simulated data. Finally, Section V presents our conclusions.

II. Ideal Attack Process

We begin our analysis by considering the idealized attack process under optimal conditions and the time available to perform each step of an attack. The total time required by an attack can be defined as follows:

$$\Delta T_{\text{total}} = T_{\text{inference}} + T_{\text{ghost-creation}} + T_{\text{attack-injection}} + T_{\text{calib}} \quad (1)$$

where: $T_{\text{inference}}$ is the time for an attacker to detect the target victim and other objects in a scene (elaborated in II-A), T_{calib} is the time for an attacker to align their laser system to the victim's angular position (elaborated in Section: II-E), $T_{\text{attack-injection}}$ is the time for an attacker to inject all the points (elaborated in Section: II-D) and $T_{\text{ghost-creation}}$ is the time required to identify critical points and generate ghost objects (elaborated in Section: II-B).

A. Target Detection

To launch a LiDAR attack, an attacker must accurately determine the location of the victim and that of other objects in a scene. This is straightforward in lab conditions, but is challenging in real-world scenarios. The attacker may use their own LiDAR to estimate the target's location. Then the time required consists of the detector's inference time, $T_{\text{inference}}$, and the LiDAR's full rotation time, T_{rotation}. For instance, with a scanning frequency of $f_s = 10$Hz and a transformer-based architecture, $T_{\text{inference}} = 0.4$s and $T_{\text{rotation}} = 0.1$s. Since $T_{\text{inference}}$ is significantly longer than T_{rotation}, the attacker could miss the opportunity to inject points into three consecutive frames.

In order to overcome the above problem, the attacker could use dead reckoning to predict the target's position. However, any rapid or unpredictable speed changes will deteriorate the accuracy of this method over time. So, $T_{\text{inference}}$ needs to be below 0.1 s—preferably even faster—to reliably detect the victim and inject the points accurately.

B. Spoofing Point Creation

We define $T_{\text{ghost-creation}}$ as the time required by an attacker to identify critical points and create ghost objects. To achieve this objective efficiently, an attacker may attempt to duplicate

points from existing objects to create spoofed targets that appear genuine or to leverage a reduced attack budget. Existing iterative methods for ghost object creation, [6] and Rain-Reaper [7] are computationally expensive, often requiring tens of minutes, which makes them impractical in real-time applications. For instance, in Rain-Reaper, the fitness function stage involves applying the inference of multiple detectors on a population of 100 samples per generation. This process takes approximately 62.7 seconds per generation, and determining the critical points typically requires multiple generations and additional computations.

We applied Rain-Reaper to measure the time that is needed to obtain the optimal combination of critical points for varying numbers of points. As shown in Figure 4, generating a successful attack with just two points takes at least 300 seconds. In practice, a successful attack generally requires at least 20 points, making such approaches infeasible for real-time execution. Using our methodology, we find that the detection steps (quantified using $T_{\text{inference}}$) can not only identify the primary victim target and other road objects in the scene, but also yield a reduced detection budget for each. These minimal point sets—extracted from other detected objects—can then be reused as optimised attack budgets to inject ghost objects, enabling faster and more feasible real-time attacks.

C. Synchronization with the LiDAR Scanning Pattern

To spoof points with the desired accuracy, an attacker needs to synchronize with the LiDAR's scanning pattern. The time to achieve synchronization should thus be estimated. For example, if the number of angular steps needed for synchronization (N_{steps}), is 100 and the processing frequency of the attacker's system ($f_{\text{processing}}$), is 1 kHz, an attacker will need a synchronization time, $T_{\text{sync}} = \frac{N_{\text{steps}}}{f_{\text{processing}}} = 0.1\text{s}$.

D. Injecting Spoofed Points

The attack involves injecting multiple spoofed points at specific positions in the LiDAR's field of view. The attack injection time, T_{attack}, can be calculated as follows.

$$T_{\text{attack-injection}} = N_p \cdot \Delta t_p. \tag{2}$$

where N_p is the number of spoofed points and Δt_p is the time required to inject one spoofed point. So, for $N_p = 1000$ and $\Delta t_p = 5.56\,\mu\text{s}$, $T_{\text{attack}} = 1000 \cdot 5.56 \times 10^{-6} = 5.56\,\text{ms}$.

E. Angular Calibration

The attacker can calculate and align their laser system to the victim's angular position. The required angular calibration time can be calculated as:

$$T_{\text{calib}} = \frac{\Delta\theta}{v_{\text{aim}}}. \tag{3}$$

where $\Delta\theta$ is the angular displacement (degrees) and v_{aim} is the aiming speed (degrees/second). $\Delta\theta$ can be defined as:

$$\Delta\theta = \arctan\left(\frac{v_{\text{rel}} \cdot T_{\text{detect}}}{d_{\text{target}}}\right) \tag{4}$$

where v_{rel} is the relative speed between the victim and the attacker, T_{detect} is the detection time and d_{target} is the distance between the attacker and the victim. For example, assuming $\Delta\theta = 5°$ and $v_{\text{aim}} = 3333°/\text{s}$ (using state-of-the-art Thorlabs GVS Series): $T_{\text{calib}} = \frac{5}{3333} = 0.0015\,\text{s}$.

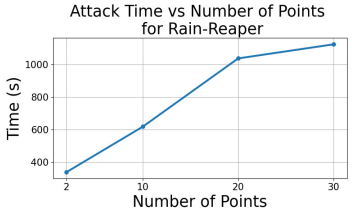


Fig. 4: Time (s) vs Number of Points when applying Rain-Reaper [7] to find the best combination of critical points.

F. Total attack time

Assuming that the victim uses a LiDAR operating at $f_s = 10\,\text{Hz}$, the attack must be performed in 0.1 seconds to spoof all frames. Let us consider that the synchronization with LiDAR pulses (which would only require 0.1s) is done before launching the attack, and the attacker's LiDAR provides the detection before the next pulse of the victim's LiDAR. The ΔT_{total} would be larger than 0.1s, making it impractical to launch an attack on all frames. The largest impact on the attack's feasibility is the inference time, $T_{\text{inference}}$, and the attack ghost creation time, $T_{\text{ghost-creation}}$, which motivates us to explore this further in the following sections.

III. Methodology

As ΔT_{total} must be under 0.1s, but given that $T_{\text{inference}} = 0.4\text{s}$ and $T_{\text{ghost-creation}} \geq 62.7\text{s}$, the total delay is $\Delta T_{\text{total}} \geq 63.107\text{s}$. This makes the attack impractical in real-world scenarios, with $T_{\text{inference}}$ and $T_{\text{ghost-creation}}$ being the primary limiting factors. Reducing the number of spoofed points and using fewer points for detection could not only shorten the attack time but also reduce delay and improve precision, as fewer points would need to be manipulated.

To reduce both $T_{\text{inference}}$ and $T_{\text{ghost-creation}}$ we propose a unified geometry-based methodology supporting two attacker-centric tasks:

- **(Task 1) Accelerated Detection**: This task aims to reduce $T_{\text{inference}}$ by downsampling the attacker's own LiDAR point cloud to reduce inference time while preserving both Intersection over Union (IoU) and detection confidence of legitimate targets.
- **(Task 2) Fake-Object Injection**: This task aims to reduce $T_{\text{ghost-creation}}$ and improve attack accuracy by selecting a small set of points (the reduced attack budget) to inject into the victim's LiDAR. This will cause a new bounding box to appear with sufficiently high IoU (relative to an attacker-defined target bounding box) *and* confidence. This should be achieved rapidly, without iterative or gradient-based point selection.

A. Problem Statement

Let $\mathcal{P} \subset \mathbb{R}^3$ be a LiDAR point cloud of size N. A detector $f_{\boldsymbol{\theta}}(\mathcal{P})$ outputs a set of 3D bounding boxes and associated confidences: $\mathcal{D}(\mathcal{P}) = \{(b_1, c_1), \ldots, (b_{|\mathcal{D}|}, c_{|\mathcal{D}|})\}$.

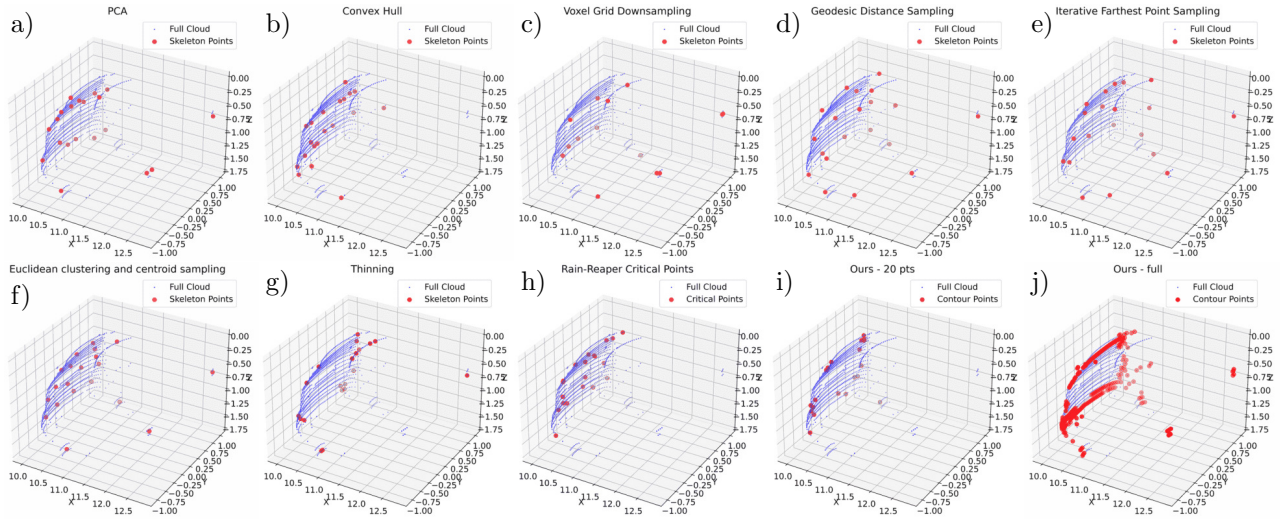


Fig. 5: Qualitative representation of the different sampling methods on the point cloud of a car positioned at 10 meters in front of the AV. The methods include: a) Principal Component Analysis; b) Convex Hull; c) Voxel Grid Downsampling; d) Geodesic Distance Sampling; e) Iterative Farthest Point Sampling; f) Euclidean clustering and centroid sampling; g) Thinning; h) Rain-Reaper [7] critical points; i) Our Method using 20 points only; j) Our Method using all contour points found.

1) Task 1: Accelerated Detection, balancing Inference Time vs. IoU and Confidence: We want to reduce the point count from N to $M \ll N$, thereby *lowering the inference time* $t(\mathcal{P}_{\text{reduced}})$ significantly but without degrading excessively IoU or confidence. Hence, we *jointly* require:

$$\text{IoU}(\hat{b}_i, b_i^{(\text{gt})}) \approx \text{IoU}(b_i, b_i^{(\text{gt})}), \quad \hat{c}_i \approx c_i, \tag{5}$$

for each legitimate detection i. Formally, we set thresholds ΔIoU_{max} and ΔConf_{max} such that:

$$|\,\text{IoU}(b_i, b_i^{(\text{gt})}) - \text{IoU}(\hat{b}_i, b_i^{(\text{gt})})| \leq \Delta\text{IoU}_{\max} \tag{6}$$

$$|c_i - \hat{c}_i| \leq \Delta\text{Conf}_{\max} \tag{7}$$

while ensuring $t(\mathcal{P}_{\text{reduced}}) \ll t(\mathcal{P})$. The attacker's optimization problem can then be defined as:

$$\min_{\substack{\mathcal{P}_{\text{reduced}} \subseteq \mathcal{P} \\ |\mathcal{P}_{\text{reduced}}| = M}} t(\mathcal{P}_{\text{reduced}})$$

$$\text{subject to} \begin{cases} |\text{IoU}_{\text{full}}^{(i)} - \text{IoU}_{red}^{(i)}| \leq \Delta\text{IoU}_{max}, \\ |c_i - \hat{c}_i| \leq \Delta\text{Conf}_{max}, \end{cases} \tag{8}$$

2) Task 2: Fake-Object Injection: Balancing Attack Budget, Time, and Desired IoU/Confidence: Now the attacker wishes to inject a ghost bounding box $(\widetilde{b}, \widetilde{c})$ in a victim's LiDAR $\mathcal{P}_{\text{victim}}$. This bounding box should appear with:

- **High IoU** w.r.t. an attacker-chosen box $b_{\text{fake}}^{(\text{gt})}$, i.e. $\text{IoU}(\widetilde{b}, b_{\text{fake}}^{(\text{gt})}) \geq \gamma$.
- **High Confidence** $\widetilde{c} \geq \beta$.

where γ and β are thresholds for bounding-box overlap and confidence, respectively. The attacker must add only a small set of points $\mathcal{P}_{\text{inj}}$ (the reduced attack budget) and do so quickly, avoiding time-consuming iterative or gradient-based point selection.

We therefore wish to minimize the injection time $T_{\text{ghost-creation}}$, as follows:

$$\min_{\substack{\mathcal{P}_{\text{inj}} \subseteq \mathcal{P} \\ |\mathcal{P}_{\text{inj}}| \leq \mathcal{B}_{\max}}} T_{\text{ghost-creation}} \quad \text{subject to} \quad \begin{cases} \text{IoU}(\widetilde{b}, b_{\text{fake}}^{(\text{gt})}) \geq \gamma, \\ \widetilde{c} \geq \beta, \\ |\mathcal{P}_{\text{inj}}| \leq \mathcal{B}_{\max}, \end{cases} \tag{9}$$

where $\mathcal{B}_{\max}$ is the budget (max number of injection points).

B. Unified Geometry-Based Pipeline

Solving (8) or (9) with direct combinatorial or gradient-based methods is typically intractable as LiDAR point clouds are very large. Instead, we employ a pipeline of geometric and heuristic methods to identify critical subsets of points that preserve detection quality while reducing the number of points

1) Stage 1: Ground Plane Removal: The initial step involves removing ground points in order to focus on the objects of interest. This can be achieved through:

a) Height Thresholding: All points with $z_i < z_{\text{threshold}}$ are classified as ground and removed:

$$\mathcal{P}_{\text{filtered}} = \{\mathbf{p} \in \mathcal{P} : z(\mathbf{p}) \geq z_{\text{threshold}}\}. \tag{10}$$

b) RANSAC-Based Plane Segmentation: Fit a plane $\pi : ax + by + cz + d = 0$ to the point cloud using RANSAC, and remove inliers that satisfy $|ax_i + by_i + cz_i + d| \leq \epsilon$. Let $\mathcal{P}_{\text{filtered}} = \mathcal{P} \setminus \mathcal{P}_{\text{ground}}$, where $\mathcal{P}_{\text{ground}}$ is the set of inlier (ground) points.

2) Stage 2: Grid-Based Clustering: The filtered cloud $\mathcal{P}_{\text{filtered}}$ is next partitioned into clusters through voxel-based spatial grouping. Define a voxelization map $\mathcal{V} : \mathbb{R}^3 \to \mathbb{Z}^3$:

$$\mathcal{V}(\mathbf{p}_i) = \left\lfloor \frac{\mathbf{p}_i}{v} \right\rfloor, \tag{11}$$

where $v > 0$ is the voxel size, and the floor function is applied element-wise. Points mapping to the same voxel index $\mathbf{v}_j \in \mathbb{Z}^3$ are grouped into a cluster C_j. We apply a 3D connected-component labeling to merge adjacent voxels: $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$, here each C_j is a cluster of points that share proximity in voxel space.

3) Stage 3: Downsampling via Multiple Strategies: For each cluster C_j, we apply one of several down sampling strategies to select a subset $C_j^{\text{down}} \subseteq C_j$. The choice of strategy aims to retain points critical for object detection. The strategies include:

a) **Principal Component Analysis (PCA)**: Retain points aligned with principal eigenvectors of the covariance

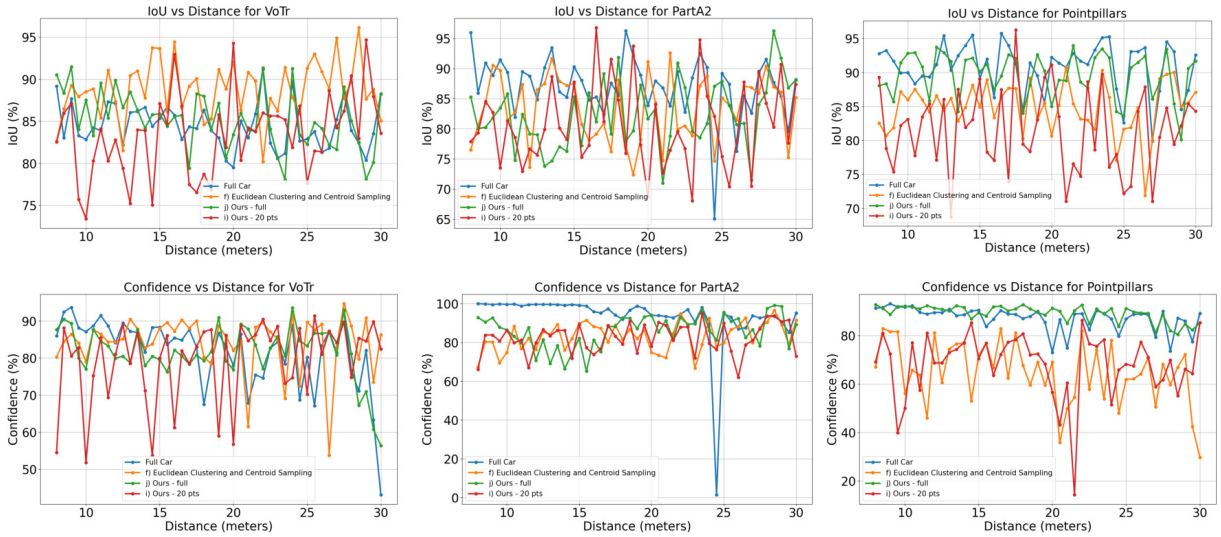


Fig. 6: IoU vs Distance and Confidence vs Distance comparison between Our method using full contour, Our method using only 20 points, the full car point cloud and Euclidean Clustering and Centroid Sampling for 3 detectors: Pointpillars [8], VoTr [9]; and PartA2 [10].

matrix Σ_j of C_j:

$$\Sigma_j = \frac{1}{|C_j|} \sum_{\mathbf{p} \in C_j} (\mathbf{p} - \mu_j)(\mathbf{p} - \mu_j)^\top, \tag{12}$$

where μ_j is the mean of points in C_j. Points are selected based on their projections onto the principal components.

b) **Convex Hull**: Retain points forming the convex hull $\mathrm{CH}(C_j)$ of the cluster: $C_j^{\text{down}} = \mathrm{CH}(C_j)$.

c) **Voxel Grid Downsampling**: Retain one representative point per voxel within each cluster.

d) **Geodesic Distance Sampling**: Select points based on geodesic distances to ensure spatial coverage within the cluster.

e) **Iterative Farthest Point Sampling (FPS)**: Iteratively select points that are farthest from the current subset to maximize spatial diversity:

$$\mathcal{P}_{\text{FPS}} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{m_j}\}, \tag{13}$$

where $\mathbf{p}_{k+1} = \arg\max_{\mathbf{p} \in C_j \setminus \mathcal{P}_{\text{FPS}}} \min_{\mathbf{q} \in \mathcal{P}_{\text{FPS}}} \|\mathbf{p} - \mathbf{q}\|_2$.

f) **Euclidean clustering and centroid sampling**: Use the centroid of the cluster as the representative point:

$$\mathbf{c}_j = \frac{1}{|C_j|} \sum_{\mathbf{p} \in C_j} \mathbf{p}, \tag{14}$$

and set $C_j^{\text{down}} = \{\mathbf{c}_j\}$.

g) **Thinning**: Uniformly subsample points from the cluster to achieve the desired reduction.

Our Geometry-Inspired Contour Extraction

The concept of *critical points* was first introduced in [11], emphasizing the importance of specific input points in capturing global features for object classification. Building on this idea, Wicker et al. [12] demonstrated the vulnerability of object recognition and classification models, such as VoxNet [13], to occlusion attacks that exploit these critical points. More recently, the genetic algorithm-based approach Rain-Reaper demonstrated the ability to identify critical points for effective spoofing. Our analysis of Rain-Reaper revealed that contour points carry the most information about the target. Our proposed geometric- and heuristic-based method includes scan line extraction, functional contour selection and farthest point selection.

a) Scan Line Extraction: We slice $\mathcal{P}_{\text{filtered}}$ into horizontal layers by defining a set of bins based on z-coordinates:

$$S_i = \{\, \mathbf{p} \in \mathcal{P}_{\text{filtered}} : z_i \leq z(\mathbf{p}) < z_i + \Delta z \}, \tag{15}$$

where Δz is the bin size, and $z_i = z_{\min} + (i-1)\Delta z$. Each S_i is treated as a *scan line* capturing a horizontal slice.

b) Functional Contour Selection: We identify the lowest and highest *functional* scan lines ($S_{\text{lowest}}, S_{\text{highest}}$) that likely correspond to the base and top edges of objects. These are included in the set: $\mathcal{P}_{\text{contour}} = S_{\text{lowest}} \cup S_{\text{highest}}$.

c) Farthest Point Selection: Within each scan line S_i, we identify the "extremal" points along the x and y axes:

$$\begin{aligned} \mathbf{p}_{\min x} &= \arg\min_{\mathbf{p} \in S_i} x(\mathbf{p}), \quad \mathbf{p}_{\max x} = \arg\max_{\mathbf{p} \in S_i} x(\mathbf{p}), \\ \mathbf{p}_{\min y} &= \arg\min_{\mathbf{p} \in S_i} y(\mathbf{p}), \quad \mathbf{p}_{\max y} = \arg\max_{\mathbf{p} \in S_i} y(\mathbf{p}). \end{aligned} \tag{16}$$

d) Final Aggregation and Random Sampling: If the union of contour and extremal points, $\mathcal{P}_{\text{inj}} = \mathcal{P}_{\text{contour}} \cup \mathcal{P}_{\text{extremal}}$, does not reach the desired count M, we randomly sample additional points from $\mathcal{P}_{\text{filtered}}$ to fulfill M.

IV. Experiments and Evaluation

Through our experiments, our aim is to answer two key questions: *i) how well can we reduce detection inference time while maintaining high detection accuracy?* and *ii) how fast can we generate effective ghost object attacks?*

A. Data

1) Simulation Data: To explore how well our algorithm works on cars positioned at different distances and orientations, we created a simulated dataset using MAVS [15]. A Velodyne HDL-64E sensor was positioned 1.73 meters above the host vehicle (AV) to match the setup of the KITTI dataset. We simulated a car in front of the AV, aligned parallel, at distances from 8 to 30 meters with increments of 0.5 meters

TABLE I: The attack time and BEV IoU and Confidence of our methodologies on simulated data and real data [14]; P.P.: PointPillars; Conf.: Detector Confidence.

					Total Detection Time							
Data	**Detector**	**PCD**	**IoU**	**Conf.**	**Stage 1**	**Stage 2**	**Stage 3**	**Detection**	$T_{\text{inference}}$	**Time Improvement**	ΔT_{total}	**Attack Budget Reduction**
Simulated Data	**VoTr**	Original	87%	75%	N/A			0.278s	0.278s	N/A	0.285s	N/A
		Ours - 20	83%	78%	N/A	0.0004s	0.0006s	0.062s	0.063s	**4.41 x faster**	0.070s	**up to 73 x**
		Ours - Full	85%	82%	N/A	0.0005s	0.0008s	0.062s	0.063s			**up to 5.41 x**
	PartA2	Original	86%	93%	N/A			0.157s	0.157s	N/A	0.164s	N/A
		Ours - 20	78%	78%	N/A	0.0004s	0.0006s	0.069s	0.070s	**2.24 x faster**	0.077s	**up to 73 x**
		Ours - Full	79%	82%	N/A	0.0005s	0.0008s	0.069s	0.070s			**up to 5.41 x**
	P.P.	Original	88%	87%	N/A			0.092s	0.092s	N/A	0.099s	N/A
		Ours - 20	80%	61%	N/A	0.0004s	0.0006s	0.025s	0.026s	**3.53 x faster**	0.033s	**up to 73 x**
		Ours - Full	89%	85%	N/A	0.0005s	0.0008s	0.025s	0.026s			**up to 5.41 x**
Real Data	**VoTr**	Original	87%	96%	N/A			0.100s	0.100s	N/A	0.107s	N/A
		Ours - 20	76%	74%	0.003s	0.0008s	0.0006s	0.061s	0.065s	**1.53 x faster**	0.072s	**up to 51 x**
		Ours - Full	80%	91%	0.003s	0.0008s	0.0008s	0.061s	0.065s			**up to 6 x**
	PartA2	Original	89%	99%	N/A			0.089s	0.089s	N/A	0.096s	N/A
		Ours - 20	74%	71%	0.003s	0.0008s	0.0006s	0.068s	0.072s	**1.23 x faster**	0.079s	**up to 51 x**
		Ours - Full	78%	91%	0.003s	0.0008s	0.0008s	0.068s	0.072s			**up to 6 x**
	P.P.	Original	87%	94%	N/A			0.040s	0.040s	N/A	0.047s	N/A
		Ours - 20	77%	68%	0.003s	0.0008s	0.0006s	0.025s	0.029s	**1.37 x faster**	0.036s	**up to 51 x**
		Ours - Full	80%	87%	0.003s	0.0008s	0.0008s	0.025s	0.029s			**up to 6 x**

and at various orientations with respect to the AV (0^o, 10^o, 170^o, 180^o, 190^o, 350^o) producing a total of 276 samples. For our reported experiments, we mainly focused on the 0^o orientation, analysing 45 distances. The choice of distances and angles is motivated by safety considerations (e.g. braking distance) introduced in [7].

2) Real Experimental Data: We also evaluated our methodology on the well-established KITTI dataset [14] in order to test our approach on real data.

B. Target LIDAR Detector Models

We have explored different detector architectures such as PointPillars [8], VoTr [9] Part A2 [10], and 3D-SSD [16]. PointPillars is used to divide the points into bins in the BEV and PointNet [8] is adopted to extract features. The VoTr [9] is used to generate a voxel architecture with Transformers, Part A2 [10] processes raw point clouds, and 3D-SSD [16] groups points into some 3D bins. We have used SOTA detectors trained on the KITTI dataset [14]. We start our analysis by selecting a random sample from the KITTI dataset and recording its inference time using an NVIDIA GM200 [GeForce GTX TITAN X]. Next, we measure the change in inference time when randomly reducing the data and retaining only 1/2, 1/4, 1/8, 1/16, and 1/32 of the points. As shown in Figure 7, inference time decreased for Part A2, PointPillars, and VoTr models but remained unchanged for 3D-SSD. This trend was consistent when evaluating inference time across the entire KITTI dataset. Upon further analysis of the key factors influencing detection time, we identified voxelization as a significant contributor. Reducing the quantity of data effectively reduced the time required for this task, resulting in faster inference. Given these observations, we chose the PartA2, PointPillars, and VoTr models in this paper.

C. Evaluation Metrics

1) Task 1: We evaluate Task 1 based on two key metrics: **Target Detection** and **Target Detection Time Improvement. Target Detection** evaluates our methodology's ability to maintain a high detection accuracy using Bird's Eye View Intersection over Union (BEV IoU) and detector confidence

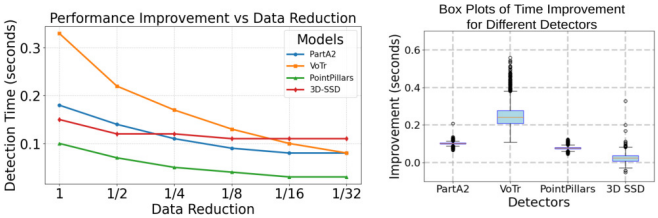


Fig. 7: Performance Improvement vs Data Reduction for one Kitti sample and Box Plots of Improvement for Different Detectors for over 7000 samples when point clouds are reduced to 1/8 of the total number of points.

levels. Specifically, we are interested in $\text{IoU}(\hat{b}_i, b_i^{(\text{gt})})$ and $\hat{c}_i$. **Target Detection Time Improvement** evaluates the improvement in detection time as $t(\mathcal{P})$ / $t(\mathcal{P}_{\text{reduced}})$.

2) Task 2: We evaluate attack quality using two key metrics: **Attack Detection** and **Attack Ghost Object Creation Time. Attack Detection** evaluates our ability to deceive a detector by appearing as a genuine target, measured using Bird's Eye View Intersection over Union (BEV IoU) and detector confidence levels. **Attack Ghost Object Creation Time** the time it takes to create the ghost object attacks, $T_{\text{ghost-creation}}$.

D. Results and Discussion

1) Choice of Down sampling method: To address Question 1, we first compare the detection accuracy and confidence of our down sampling methods against a set of 20 critical points selected using Rain-Reaper. This point budget—based on the original paper's setting [7]—represents the smallest number of points experimentally shown to be effective across multiple detectors and accross high and low-resolution LiDAR sensors. A visualization of the different down sampling methods is shown in Figure 5. Additionally, we evaluate performance against the full car at various distances (e.g., 8 to 30m), ensuring a fair comparison by considering only the car or the critical points in each sample.

We evaluated all methods using the $\text{IoU}(b_i, b_i^{(\text{gt})}) - \text{IoU}(\hat{b}_i, b_i^{(\text{gt})})$ and $c_i - \hat{c}_i$ metrics. Our results show that our proposed methods, along with Euclidean clustering, centroid sampling, and Rain-Reaper's critical points, maintain the closest performance to the full-budget attack. Figure 6 illustrates a comparison of the best-performing, time-optimized

methods across different distances and detectors. Notably, our full contour method achieves the best performance, and all methods achieve IoU values consistently exceeding 70%.

2) Attack Evaluation: To evaluate the efficiency and accuracy of our approaches, we conducted experiments on both simulated and real data.

For simulated data, we averaged the BEV IoU and confidence over 45 distances (8 to 30m) and recorded the detection time for each detector, along with the time required for each processing stage. Since the simulator provides a uniform ground, we can filter data by applying a simple height filter, to save the processing time of the stage 1. As shown in Table I, our method maintains high BEV IoU and confidence for all detectors we tested in our experiments, both for the full contour (where PointPillars achieves superior IoU or VoTr achieves better confidence) and our 20-point reduced method. Additionally, our approach improves detection speed by up to 4.41× while significantly reducing data requirements— e.g. a detection that would require a KITTI point cloud of 10k points only requires 20 points using our method - 500 times fewer points. In terms of the attack budget, an attack that would normally require injecting a car with 1,476 points requires 276 points using our full contour method or 20 points for our reduced method. This results in a 5.34× reduction in the attack budget for the full contour method and a 73× reduction for the 20-point approach. While we also observed a 171× reduction at very close range (within 5 meters), this lies beyond the scope of our primary evaluation, which focuses on distances of 8 meters and beyond.

For real-world validation, we have tested our method on 100 randomly selected cars from the KITTI dataset [14], ensuring the vehicles were within 8–30m of the the autonomous vehicle (AV). Without loss of generality, we focus on a region: 5m along the Y-axis and 100 m along the X-axis, with the AV's LiDAR positioned at (0,0). The experimental results have shown that our method maintains high BEV IoU and confidence across all detectors for both the full contour and the reduced method while achieving a 1.5-time speed improvement.

3) Limitations: Despite promising results, some limitations still remain. Clustering could be improved for scenes with multiple objects, and even if faster GPUs can reduce attack time, effectiveness is still constrained by LiDAR rotation speed. While rain-related feasibility is covered in prior work [7], this study does not assess other factors like fog or sunlight, which may impact real-world performance.

V. Conclusions and future work

We propose a geometric- and heuristic-based method to minimize both the attack budget and execution time of LiDAR attacks thereby demonstrating that they are feasiable in real-world situations. Our approach select a few critical adversary points, whilst maintaining high detection confidence and consequently improving attack execution speed by up to four times. This reduces the points required by state-of-the-art attacks from thousands of points to only a fraction (up to 73 times attack budget reduction) while preserving attack effectiveness. Evaluation results on both simulated and real-world (KITTI) datasets has shown that our method enables real-time adversarial attacks.

Future work could focus on enhancing attack adaptability, exploring multi-frame strategies, and developing methods that generalize across diverse LiDAR architectures. Edge detection and other vision techniques could improve fake object injection by selecting structurally meaningful edges rather than arbitrary points. It would also be valuable to analyze the sensitivity of attacks to point selection and spatial precision, and to extend evaluations to complex scenes with multiple moving objects. On the defense side, future work should explore countermeasures grounded in the physical invariance of rain, which remains an open challenge.

Acknowledgements

This research was supported by the NTU-Imperial Global Fellows Program, under the leadership and invaluable guidance of Assoc. Prof. Sierin Lim and Mrs. Seah Wan Er, the Imperial College London Department of Computing, and the A*STAR SINGA scholarship, which funded the research visit, with special thanks to Mrs. Eunice Heng for her exceptional support. The authors extend their deepest gratitude to Dr. Teoh Eam Khwang for his insightful suggestions during the writing of this paper. Assoc. Prof. B.H. Soong gratefully acknowledges the support of MOE Grant RG66/22. We also wish to thank the anonymous reviewers for their valuable feedback.

References

[1] Z. Hau, K. T. Co, S. Demetriou, and E. C. Lupu, "Object removal attacks on lidar-based 3d object detectors," *AutoSec (co-located with NDSS)*, 2021.

[2] J. Petit, B. Stottelaar, and M. Feiri, "Remote attacks on automated vehicles sensors : Experiments on camera and lidar," in *Black Hat Europe*, 2015.

[3] H. Shin, D. Kim, Y. Kwon, and Y. Kim, "Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications," in *CHES 2017*, 2017, pp. 445–467.

[4] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao, "Adversarial sensor attack on lidar-based perception in autonomous driving," in *Proceedings of the 2019 ACM SIGSAC*, ser. CCS '19, 2019, p. 2267–2281.

[5] J. Sun, Y. Cao, Q. A. Chen, and Z. M. Mao, "Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures," in *USENIX 2020*, 2020.

[6] Z. Jin, X. Ji, Y. Cheng, B. Yang, C. Yan, and W. Xu, "Pla-lidar: Physical laser attacks against lidar-based 3d object detection in autonomous vehicle," in *2023 IEEE Symposium on Security and Privacy (SP)*, 2023, pp. 1822–1839.

[7] R. Capraru, E. C. Lupu, S. Demetriou, J.-G. Wang, and B. H. Soong, "Rain-reaper: Unmasking lidar-based detector vulnerabilities in rain," in *2024 IROS*, 2024, pp. 3212–3217.

[8] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *2019 CVPR*, pp. 12 689–12 697.

[9] J. Mao, Y. Xue, M. Niu, H. Bai, J. Feng, X. Liang, H. Xu, and C. Xu, "Voxel transformer for 3d object detection," 2021.

[10] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network," *IEEE PAMI*, vol. 43, no. 08, pp. 2647–2664, aug 2021.

[11] C. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *2017 CVPR*, pp. 77–85.

[12] M. Wicker and M. Kwiatkowska, "Robustness of 3d deep learning in an adversarial setting," in *2019 CVPR*, jun 2019, pp. 11 759–11 767.

[13] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IROS*, pp. 922–928.

[14] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: the kitti dataset," *The International Journal of Robotics Research*, vol. 32, pp. 1231–1237, 09 2013.

[15] "Msu autonomous vehicle simulator," accessed: 2025-06-10. [Online]. Available: https://www.cavs.msstate.edu/capabilities/mavs.php

[16] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3dssd: Point-based 3d single stage object detector," in *2020 CVPR*, 2020, pp. 11 037–11 045.