π-Jack: Physical-World Adversarial Attack on Monocular Depth Estimation with Perspective Hijacking

Tianyue Zheng[1], Jingzhi Hu[2], Rui Tan[2], Yinqian Zhang[1], Ying He[2], Jun Luo[2] *
[1]*Southern University of Science and Technology,*
[2]*Nanyang Technological University*

Abstract

Monocular depth estimation (MDE) plays a crucial role in modern autonomous driving (AD) by facilitating 3-D scene understanding and interaction. While vulnerabilities in deep neural networks (e.g., adversarial perturbations) have been exploited to compromise MDE, existing attacks face challenges in target accessibility and stealthiness. To address these limitations, we introduce π-Jack, a novel physical-world attack on MDE via *perspective hijacking*. It is based on an observation that MDE relies heavily on perspective cues to infer depth, yet these cues can be manipulated by strategically placing common 3-D objects in AD scenes. With an optimization-based approach, π-Jack "hijacks" the perspective information and alters the target pixels' depths perceived by the MDE model in a black-box manner. We also show via experiments that π-Jack is effective across various MDE models and scenarios, confirming generalizability of perspective hijacking. Our extensive evaluations demonstrate that π-Jack is effective across different target and attack vectors, and increases the mean depth error by over 14 meters. Moreover, in our end-to-end AD simulation, π-Jack results in compromised lane change, sudden braking, and life-threatening collisions.

1 Introduction

Monocular depth estimation (MDE) is a technique for estimating pixel-wise distances from a single RGB image, allowing low-cost and energy-efficient monocular cameras to sense depth and replace expensive radar and lidar sensors. Consequently, autonomous vehicles (AVs) can use MDE to facilitate downstream tasks with depth information, such as semantic segmentation [38], 3-D object detection [41], visual SLAM [69], and visual relocalization [66]. Due to its benefits and capabilities, MDE has garnered interest from both academia [21, 67, 68] and industry, including leading companies such as Tesla [1, 59], Waymo [85, 86], and Toyota [25, 84]. In particular, Tesla regards MDE as a pivotal component in their AV-AI ecosystem [1], and has already integrated MDE into their production-grade vehicles [59].

With the widespread adoption of MDE in the wake of all-vision autonomous driving (AD) solutions [1, 25, 59, 84, 85, 86], understanding its security vulnerabilities becomes increasingly important. In fact, any deficiencies in MDE could lead to AVs generating low-quality models of their surroundings (the absence of depth sensors in all-vision AD solutions renders it nearly impossible to correct such deficiencies), thereby affecting the effectiveness of their decision-making and trajectory planning. The situation could worsen significantly if an adversary targets MDE for attack, aiming to artificially deviate the inference results of MDE. Should such an attack succeed, AVs would highly likely to undertake risky maneuvers, such as improper lane changes or braking actions that may escalate into possible hazardous vehicular collisions [65]. Therefore, understanding possible attacks on MDE becomes important due to the substantial real-world consequences posed by them.

Conventional attacks on MDE are performed in the digital space [70, 74], which primarily manipulate depth perception by introducing adversarial perturbations at the *pixel level* [22, 46]. These perturbations are designed through gradient-based methods to iteratively modify pixel values in captured images to maximize depth estimation error. However, applying such attacks in real-world scenarios faces three challenges. First, it is difficult for attackers to tamper with the images during data transfer or storage due to air gap and cryptographic protection. Second, the intricate adversarial perturbations are not robust to changes in real-world lighting conditions due to shadow and weather as well as geometric variations related to view angles and object surfaces. Third, even if the pixel-level perturbations are feasible, the unnatural appearance of the perturbations can alert the AV system and the human driver, thereby triggering a fail-safe mechanism such as falling back to manual driving or emergency safe stopping.

More recent physical MDE attacks [9, 71] still act on the *pixel level*, by generating adversarial perturbations digitally and then transferring these perturbations into the physical

*This work was started when Tianyue Zheng was a PhD candidate at NTU. Open-source materials to be found at https://github.com/pi-Jack/pi-Jack

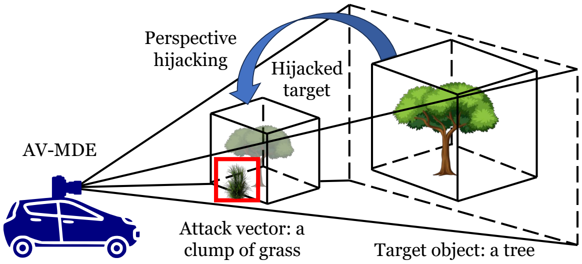


Figure 1: High-level idea of π-Jack: by strategically placing an attack vector (a clump of grass), MDE for a target object (a tree) can be hijacked.

world. They attempt to compensate for the drawbacks of digital attacks by employing methods such as style transfer [9] and Expectation over Transformation (EoT) [2, 32], as these techniques are designed to make the stickers less conspicuous and more robust to real-world conditions. However, still operating at the pixel level, these attacks remain largely impractical. For example, attaching style-transferred road markings to vehicle rears [9] still looks unnatural. Additionally, accessing the targets can be difficult due to restrictions on applying stickers on certain surfaces. Finally, the effectiveness of physical attacks is worsened by the presence of invalid colors that cannot be accurately reproduced during printing [35].

The persistent issue of adversarial perturbations (regardless of digital and physical) is their narrow focus at *pixel level* only; this calls for novel attack strategies conceived at a broader level. Inspired by the work [34] that tricks lane detection of AVs by putting markings on the road, we envision an attack operating at the *scene level*. However, designing such attacks faces two challenges. First, we need to perform an analysis of MDE's vulnerabilities and identify a scene-level depth-altering attack not achievable by previous proposals. For instance, previous physical attacks on classification tasks by using lasers [14] and projected patterns [20] are unfeasible, as they suffer from similar issues of being conspicuous and susceptible to real-world lighting conditions due to their reliance on additional light sources. Second, to achieve more practical and potent attacks, it is desirable to extend 2-D attacks (e.g., on a road sign's surface) to the 3-D physical scene. Crafting attack vectors in the 3-D scene, however, is challenging due to the difficulties in reconstructing the 3-D scene from a 2-D image and interacting with it.

A reasonable approach to achieving *scene-level* attack on MDE involves *strategically placing 3-D objects* commonly found on the road, such as traffic signs, barrier poles, and clumps of grass, as attack vectors. These objects are readily obtainable and accessible to the attacker. Moreover, these objects are inconspicuous because traffic inspectors and drivers consider them ordinary and benign. Furthermore, the texture and placement of these objects are natural and involve no delicate adversarial perturbations, making them inherently robust to real-world lighting condition changes and geometric variations. These attack vectors manipulate visual cues on which deep learning implicitly relies for MDE. Among these cues, perspective information (i.e., the spatial attributes of geometry and texture that change with distance) is crucial because it reveals the spatial relationships among the objects.

In this paper, we introduce π-Jack, a physical-world adversarial attack that harnesses the power of *perspective hijacking*. Its basic concept is depicted in Figure 1, where we strategically position an adversarial object (e.g., a clump of grass) to manipulate the perspective relationships of targets in the scene. This approach is designed to affect or "hijack" the perspective of the target object (e.g., a tree), thereby perturbing the depth estimation of the target object. We design a 3-D modeling and rendering pipeline to place the 3-D object and create composited yet photorealistic scenes. Additionally, we implement a systematic approach to optimize the position and pose of the placed 3-D object, maximizing its effectiveness of attack. This approach also adopts a novel Expectation over Motion and Illumination (EoM&I) technique to ensure the attack's robustness to AV motion. Lastly, we conduct extensive evaluations including tests in real-world scenarios to assess π-Jack's performance. In summary, this research provides the following key contributions:

- We develop a physical-world attack on MDE termed *perspective hijacking*. To the best of our knowledge, we are the first to explicitly manipulate visual cues and mislead AV-MDE system in an effective, robust, and inconspicuous manner.
- We offer a 3-D modeling and rendering pipeline allowing the composition of the attack vector into the driving scene. This pipeline allows π-Jack to test out photorealistic perspective hijacking while considering environment illumination and shadowing.
- We propose optimization strategies to systematically optimize the position and pose of the adversarial object under the constraints of sensitive perspective region and physical realization. We also develop an EoM&I technique to make π-Jack robust to vehicle motion and illumination variations.
- We provide comprehensive evaluations conducted in both composited and real-world scenes and on different MDE models to demonstrate the effectiveness and generalizability of π-Jack.

The rest of this paper is organized as follows. Section 2 introduces the background of our work. Section 3 explains the attack goal and threat model. Section 4 presents the attack design of π-Jack, while the security analysis is postponed to Section 7 after obtaining supports from evaluation results. Section 5 provides π-Jack's implementation details, experiment setup, and evaluation metrics. Section 6 reports the evaluation results, followed by security analysis and possible defense methods in Section 7. Related works are presented in Section 8. Finally, we conclude our paper in Section 9.

2 Background

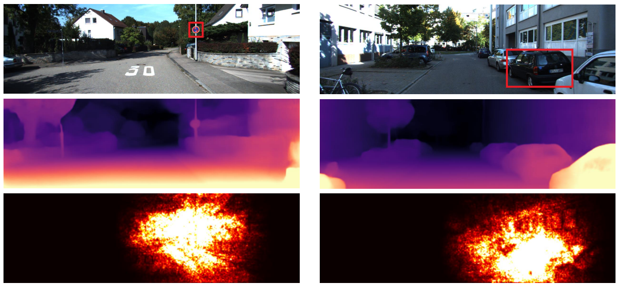
MDE is a challenging task because a single 2-D image can correspond to infinitely many 3-D scenes. However, humans can infer depth, even with one eye, based on their experience in navigating and interacting with the physical world. It has been experimentally shown that humans use visual cues such as perspective, familiar size, occlusion, shadows, and etc to make informed depth perception [57]. Drawing inspiration from this human capability, recent research has introduced deep learning-based MDE methods that uses the "experience" of sequential temporal frames as training signals [21, 67, 68]. This novel approach has been adopted by leading AV companies [1, 25, 59, 84, 85, 86], demonstrating its practical value and real-world applications.

To design an effective attack against these MDE methods, we shall identify the most vulnerable visual cues for MDE. Taking human visual perception as an analogy, it is reasonable to hypothesize that perspective cues (i.e., the spatial attributes of geometry and texture that indicate the relative locations among the objects in the scene) are essential for MDE and subject to manipulation, as there are various notable optical illusions that deceive human visual perception [23, 39]. These illusions involve parallel lines that seem to differ in length, orientation, or slope depending on the surrounding patterns, all of which distort the perspective cues. These illusions suggest that deep learning-based MDEs may also rely on perspective cues, and may be fooled by attacks with similar principles.

To have a preliminary validation of our hypothesis, we employ the widely-adopted MDE model MonoDepth2 [21] to generate a depth map with pixel values in the image representing depths, i.e., distances from the camera. We then use the saliency map [53] to understand what the depth estimation model focuses on when performing MDE. Specifically, the saliency map S is constructed by computing the gradient sum of the pixels inside the specified target region T on the depth map D, with respect to the input I as $S(u,v) = \sum_{(u_t,v_t)\in T} \frac{\partial D(u_t,v_t)}{\partial I(u,v)}$, where (u,v) represents pixel coordinates. A higher gradient means that a small change in that pixel will cause a large change in the target depth, and conversely, a lower gradient means a smaller change.

We present two example driving scenes in Figures 2(a) and 2(b), respectively. Each figure comprises three rows: an image, its estimated depth map, and its saliency map. We select a road sign and a car (two important objects in AV) as the target regions for saliency analysis in the scenes, as indicated by the red boxes. The saliency maps use brighter colors to indicate the areas most crucial for determining the depth of the selected target regions. It can be observed that, for the road sign, the sidewalk and the distant trees lining the horizon are the most salient regions; for the car, the road and the building on the side are the most salient.

All the salient regions underscore the importance of perspective cues: the parallel lines of the road converging in the distance, together with the vertical window frames of the roadside building establish a "spatial grid" that aids in perspective analysis. Additionally, the low contrast/saturation and blurred texture of the trees along the horizon indicate their remoteness and provide a reference point for perspective. This reliance on perspective cues motivates us to design the *perspective hijacking* attack: if the 3-D object we place in the scene has a similar geometric structure or texture with the target but at a different distance, the MDE model might mistakenly associate or merge our strategically-placed object with the target, thus the target perspective will be hijacked and the MDE model will output a wrong depth.



(a) Driving scene 1.

(b) Driving scene 2.

Figure 2: MDE rely on perspective cues for depth inference.

3 Problem Formulation

In this section, we present the attack goal and threat model.

3.1 Attack Goal

π-Jack aims at using 3-D physical-world adversarial objects in AV scenes to perform perspective hijacking and mislead AV-MDE model into producing incorrect depth estimation for specified target regions in the scenes. The adversarial object placement should meet the following requirements:

- *Effectiveness.* The objects should be placed in a way that they can alter the estimated target depth by the largest possible margins.
- *Accessibility.* The placed objects should be in valid and reachable areas in the scene.
- *Inconspicuity.* The objects should blend in with the environment, avoiding unusual shapes or patterns.
- *Robustness.* The attack should be robust and not affected by real-world conditions (e.g., lighting).

Given that the objects we use are ordinary 3-D objects that are generally considered benign, the inconspicuity requirement can be met. Additionally, since π-Jack does not depend on "sticker-pasting" or adversarial perturbations, it will not be influenced by environment lighting much, thus meeting the robustness requirement. Furthermore, as π-Jack does not necessitate placing objects in high-alert areas like road signs, the accessibility requirement can also be fulfilled. Lastly, for the effectiveness requirement, we provide a comprehensive investigation of how depth estimation error can be maximized in Section 4.3 and evaluate the attack effectiveness in Section 6.

Formally, the attack objective of π-Jack can be expressed as follows. Let I be a scene image and $\theta(\cdot)$ be a depth estimation model such that $D = \theta(I)$ is the estimated depth map. Let O be a collection of ordinary 3-D objects, such as barrier poles, ladders, clumps of grass, and other similar items that can be placed into the scene. For each object $o \in O$, we define P_o as the set of coordinates and poses for o in the 3-D space corresponding to I. For each coordinate and pose $(x,y,z,\rho) \in P_o$, we obtain a modified image $I_{o,x,y,z,\rho}$ by inserting o at coordinate (x,y,z) with pose ρ. The corresponding estimated depth map is denoted by $D_{o,x,y,z,\rho} = \theta(I_{o,x,y,z,\rho})$. We then define a specific region T that contains the target to be attacked, where T can be obtained by either manual or automatic (e.g., SAM [37]) segmentation. The objective of π-Jack is to find an object $o^* \in O$ and its coordinate-pose $(x^*,y^*,z^*,\rho^*) \in P_{o^*}$ that maximize the sum of estimated depth differences between the original and compromised depth maps within the region T, i.e.,

$$o^*,x^*,y^*,z^*,\rho^* = \\ \arg\max_{o \in O,(x,y,z,\rho) \in P_o} \sum_{(u,v) \in T} \left| D(u,v) - D_{o,x,y,z,\rho}(u,v) \right|, \quad (1)$$

where $D(u,v)$ and $D_{o,x,y,z,\rho}(u,v)$ denote the depth values at pixel location (u,v) in the original and compromised depth maps, respectively.

3.2 Threat Model

π-Jack focuses on the black-box setting, where the attacker can only query the AV-MDE system with inputs and observe the output depth map, without any knowledge of the MDE model's internals (e.g., model architecture, parameters, and other specifics of the victim's AV implementation). This scenario is more realistic and difficult than most previous works on adversarial attacks to camera-based AD perception [5, 9, 15, 75]. It is worth noting that obtaining the output depth map has been previously demonstrated possible [17]. Moreover, the attacker is assumed to have access to the camera parameters (both intrinsics and extrinsics) of the victim AV, which are likely to be public information. As shown in Section 6.4, π-Jack can transfer the attack vector optimized for a surrogate MDE model to a previously unseen MDE model, with a minor trade-off between a slight drop in attack success rate and depth difference.

In the considered threat scenario, the attacker is capable of placing new 3-D objects on accessible locations within the AV scene, including sidewalks, lawns, parking lots, and some parts of the road shoulders. This threat assumption is realistic because objects placed in these locations would not look suspicious or break traffic rules. Additionally, the attacker is assumed to has access to the 3-D models of these objects, which can be obtained using photogrammetric 3-D reconstruction techniques with smartphone photos and related software [24]. These 3-D models provide the attacker with the ability to optimize object placement and orientation in the digital space without the need to physically modify objects in the real world.

4 π-Jack Attack Design

In this section, we introduce the attack strategy of π-Jack. As shown in Figure 3, the whole workflow consists of five steps: i) 3-D object selection, ii) setting the stage for 3-D object placement, iii) object placement and rendering, iv) expectation

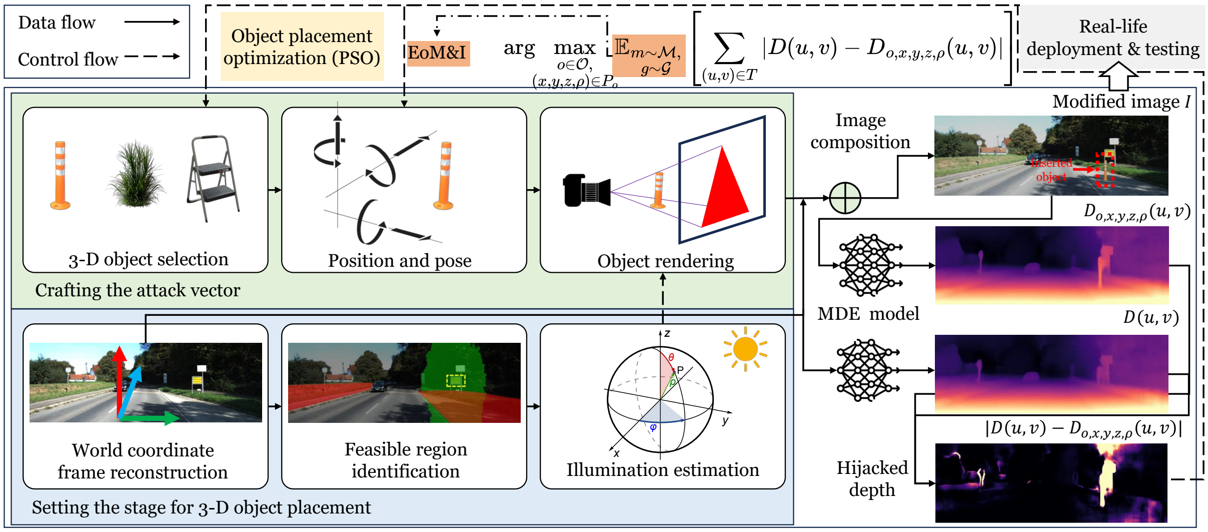

Data flow
Control flow
Object placement optimization (PSO)
EoM&I
$\arg\max_{o \in \mathcal{O}, (x,y,z,\rho) \in P_o} \mathbb{E}_{m \sim \mathcal{M}, g \sim \mathcal{G}} \left[\sum_{(u,v) \in T} |D(u,v) - D_{o,x,y,z,\rho}(u,v)| \right]$
Real-life deployment & testing
Modified image I
Image composition
Inserted object
3-D object selection
Position and pose
Object rendering
Crafting the attack vector
$D_{o,x,y,z,\rho}(u,v)$
MDE model
$D(u,v)$
$|D(u,v) - D_{o,x,y,z,\rho}(u,v)|$
World coordinate frame reconstruction
Feasible region identification
Illumination estimation
Hijacked depth
Setting the stage for 3-D object placement

Figure 3: The workflow of π-Jack's attack strategy.

over motion and illumination to improve the robustness of π-Jack attack, and finally v) real-life deployment and testing (as evaluated in Section 6.5).

4.1 3-D Object Selection

The initial step of π-Jack involves selecting a set of 3-D objects to be positioned within the scene. As discussed in Section 2, the key to successful perspective hijacking is to create a false connection between the strategically placed object and the target at a different distance, and thus manipulate the perception of the target's depth. To accomplish this, we suggest four criteria for selecting the 3-D object: i) the 3-D object should possess structures similar to the target, ii) the 3-D object should exhibit a texture akin to the target, iii) the 3-D object should have an extended shape, ensuring its influence is not limited to a localized area, and iv) the object should be ordinary and inconspicuous, so that their existence does not raise suspicion. Note that not all of the first three criteria need to be satisfied at the same time. Based on these criteria, we choose nine types of 3-D objects as potential attack vectors. Note that there are five variants for each type of object. The detailed properties of the objects are shown in Table 1.

4.2 Setting the Stage for 3-D Object Placement

In this section, we set the stage for 3-D object placement using information obtained from AV photos. To be specific, in order to reconstruct a realistic scene, it is crucial to determine the world model for placing the object and casting the shadow, and the direction of the light source (e.g., sunlight in the outdoor driving environments) for proper illumination. Furthermore, a feasible region must also be identified for placing the 3-D object, taking into consideration perspective sensitivity and adherence to traffic rules.

4.2.1 World Coordinate Frame Reconstruction

The goal of world coordinate reconstruction is to perform the perspective projection of the 3-D coordinates of (x,y,z) in the real world to its 2-D coordinates on the image plane (u,v). This allows us to place the selected object in the AV scene, render it, and evaluate how the π-Jack attack affects the MDE model. According to the pinhole camera model [56], the projection can be formulated as follows:

$$\begin{bmatrix} u & v & 1 \end{bmatrix}^\mathsf{T} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} x & y & z & 1 \end{bmatrix}^\mathsf{T}, \quad (2)$$

where $\mathbf{R} = \mathbf{R}_z(\psi)\mathbf{R}_y(\eta)\mathbf{R}_x(\phi)$ and $\mathbf{t} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^\mathsf{T}$ are the camera's extrinsic parameters that represent the rotation and translation with respect to the world coordinate system, with ψ, η, and ϕ being the Euler angles about the z, y, and x axes respectively, and (t_x, t_y, t_z) describing how the camera is translated along the three axes. $\mathbf{K}$ is the camera's intrinsic matrix that encodes the focal length f, the sensor size s, and the principal point (c_x, c_y):

$$\mathbf{K} = \begin{bmatrix} f/s_x & 0 & c_x \\ 0 & f/s_y & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$

Essentially, Eqn. (2) sets the coordinate frames of showing how a strategically placed 3-D object is perceived by the AV camera.

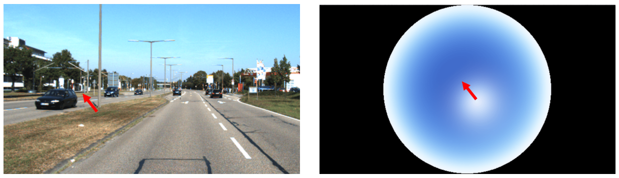
4.2.2 Illumination Estimation

Estimation of the outdoor illumination is crucial for setting the stage for 3-D object placement, because it has been demonstrated that illumination [14] and shadow [80] can be exploited to attack AV's deep learning models. The illumination condition can be readily achieved by analyzing the photo from the AV's camera. To be specific, for a clear sky, we model the sky's illumination according to [27], and the spectral radiance L_λ in the direction $\mathbf{l}$ of the skydome can be expressed as $g(\mathbf{l}, \lambda, \tau, \alpha, \mathbf{l}_s)$, where λ is the wavelength of the light, τ is atmospheric turbidity (the haziness or cloudiness), α is the ground albedo (a measure of the ground reflectivity), and $\mathbf{l}_s$ indicates the sun's position. For a diffuse overcast sky, we use the sky model $g(\mathbf{l}, \lambda, L_z)$ according to [12], where L_z is the zenith luminance.

From these radiance models, we further obtain digitized RGB values by rendering at a discrete number of wavelengths ranging from 360 to 700 nm. We refer to this conversion process as $g_{\text{RGB}}(\cdot)$, and express the digitized RGB color $g'_{\text{RGB}}(\mathbf{l})$ of a skydome direction $\mathbf{l}$ as $g'_{\text{RGB}}(\mathbf{l}) = \omega g_{\text{RGB}}(\mathbf{l})$, where ω is the exposure applied to the red, green, and blue channels. To

Table 1: Properties of the selected 3-D objects.

	Barrier pole	Flag	Grass clump	Ladder	Safety sign	Garbage bin	Traffic sign	Roadblock	Hydrant post
Structural similarity	Tree trunk, window	Tree trunk, window	Tree, bush	Scaffolding, fire escape	Vehicle, tree	Vehicle, building	Lamp post signs	Vehicle	Tree trunk, lamp post
Texture	Metallic	Glossy	Leafy	Wooden	Plastic	Glossy	Glossy	Coarse	Metallic
Extensibility	Good	Good	Fair	Good	Poor	Fair	Good	Fair	Fair
Typical height×width	$0.20\,\text{m}^2$	$0.92\,\text{m}^2$	$0.43\,\text{m}^2$	$1.20\,\text{m}^2$	$0.56\,\text{m}^2$	$1.4\,\text{m}^2$	$0.52\,\text{m}^2$	$0.73\,\text{m}^2$	$0.142\,\text{m}^2$
Stealthiness	Good	Fair	Good	Fair	Fair	Good	Fair	Fair	Fair



(a) Driving scene. (b) Estimated skydome illumination.

Figure 4: Example AV scene and estimated illumination.

generate a map of the skydome from this model, we further discretize the skydome into several directions, and render the RGB values.

Given a skydome panorama Q and the discretized indices q corresponding to sky pixels in Q, we estimate these parameters by minimizing the least-square error of skydome reconstruction using a convolutional neural network (CNN), as demonstrated in [26]. For a clear sky, we have:

$$\mathbf{l}_s^*, \omega^*, \tau^* = \arg\min_{\mathbf{l}_s,\omega,\tau} \sum_q \left(Q(q) - \omega g_{\mathrm{RGB}}\left(\mathbf{l}_q, \tau, \mathbf{l}_s\right)\right)^2 . \quad (3)$$

For an overcast sky, we have:

$$L_z^* = \arg\min_{L_z} \sum_q \left(Q(q) - \omega g_{\mathrm{RGB}}\left(\mathbf{l}_q, L_z\right)\right)^2 . \quad (4)$$

One example driving scene and its corresponding estimated skydome illumination (clear sky) are shown in Figures 4(a) and 4(b), respectively. The skydome of Figure 4(b) is oriented such that its up direction matches the forward road direction in Figure 4(a). One may readily observe that the reconstructed sunlight corresponds with the lighting in the driving scene. In particular, the shadow of the utility pole aligns perfectly with the sun's position (as indicated by the red arrows).

4.2.3 Feasible Region Identification

To place the 3-D object properly, we need to determine the feasible region where it can fit. The feasible region should meet two requirements: i) it has an impact on the depth estimation of the target, and ii) it should respect both physical laws and traffic rules. In other words, the region should be the intersection of the salient region (as mentioned in Section 2), and the valid region without violating physical laws and traffic rules. However, due to our black-box settings explained in Section 3.2, we cannot derive the saliency map by directly calculating the gradient. Following the idea of D-RISE [47], we generate saliency maps for black-box MDE models by deleting and in-painting a region of each input image frame I. We use a binary mask B to indicate the region to be deleted, and an in-painting function χ to fill in the deleted region with a realistic context. The saliency map S' is defined as the sum of absolute differences between the original and the modified output values for all pixels in the target object T:

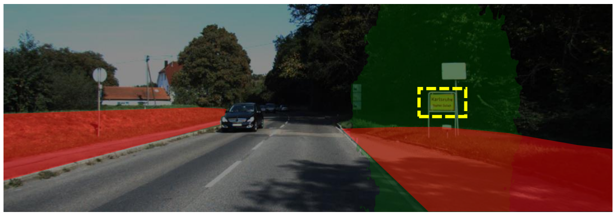


Figure 5: Feasible region identification.

$$S' = \sum_{(u_t,v_t)\in T} \left|\theta(I)(u_t,v_t) - \theta(\chi(I \odot B))(u_t,v_t)\right| , \quad (5)$$

where $\odot$ denotes element-wise multiplication. We further apply a Gaussian blur using the kernel $G(u,v)$ to smooth the saliency map and remove any discontinuities. Then, we threshold the smoothed map to obtain the salient region with saliency above the threshold $\mathcal{T}$. As a result, the processed saliency region $\mathcal{F}_{\mathrm{sal}}(u,v)$ can be expressed as:

$$\mathcal{F}_{\mathrm{sal}}(u,v) = \mathbb{I}\left[(G * S')(u,v) > \mathcal{T}\right] , \quad (6)$$

where $\mathbb{I}(\cdot)$ is the indicator function.

Subsequently, we annotate the valid region $\mathcal{F}_{\mathrm{val}}(u,v)$ that satisfies both physical and traffic constraints (e.g., a 3-D object cannot be placed in the sky due to physical laws, nor can it obstruct the road against traffic regulations). The valid region can be obtained either by a human annotator or a semantic segmentation tool like [42]. Consequently, the feasible region can be expressed as follows:

$$\mathcal{F}(u,v) = \mathcal{F}_{\mathrm{sal}}(u,v) \cap \mathcal{F}_{\mathrm{val}}(u,v). \quad (7)$$

An example feasible region is shown in Figure 5. The target is the road sign highlighted by the dashed yellow box, the green color represents the thresholded salient region $\mathcal{F}_{\mathrm{sal}}$, and the red color indicates the valid region $\mathcal{F}_{\mathrm{val}}$. The feasible region $\mathcal{F}$ is the overlapping area containing parts from both the sidewalk and the lawn.

While the feasible region $\mathcal{F}(u,v)$ currently exists in the image space, our goal is to acquire a feasible region $\mathcal{F}(x,y,z)$ that dictates the 3-D object placement in world coordinates. To achieve this, we map the 2-D coordinate (u,v) back to the 3-D coordinate (x,y,z). This task poses an ill-posed problem since infinitely many 3-D points can project to the same 2-D point on the image plane. Nonetheless, as the 2-D plane containing the feasible region is known (typically the ground), we can compute the 3-D coordinates uniquely. This process is the inverse of the one described in Section 4.2.1 and can be performed by solving for (x,y,z) based on the equation:

$$s\begin{bmatrix} u & v & 1 \end{bmatrix}^{\mathsf{T}} = \mathbf{K}\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}\begin{bmatrix} x & y & z & 1 \end{bmatrix}^{\mathsf{T}}, \quad (8)$$

where s is a scale factor resulting from the fact that the 2-D coordinates (u,v) are homogeneous coordinates, which means that they can be scaled by any non-zero factor while still representing the same point on the image plane. For example, although $(u,v,1)$ and $(2u,2v,2)$ represent two distinct 3-D points in the world coordinate frame, they are equivalent homogeneous coordinates in the image. The scale factor s accounts for this ambiguity when converting from 2-D to 3-D coordinates. Take a feasible region on the ground as an example, to obtain the 3-D coordinate, we can solve for s by substituting $z = z_{\text{ground}}$ into Eqn. (8) and taking the dot product with the third row of $[\mathbf{R}\ \mathbf{t}]$, then substitute it back to get $(x, y, z_{\text{ground}})$.[1] the feasible region might reside in other planes as well.

4.3 Object Placement and Rendering

After reconstructing the world coordinate frame and identifying the feasible region, we can model the chosen 3-D object into the scene. In the modeling process, coordinate (x, y, z) in the world coordinate frame and pose ρ are obtained by the heuristic optimization method. We opt for heuristic optimization over gradient-based optimization (e.g., backpropagation) techniques because we cannot compute the gradients of black-box MDE models. We especially choose the particle swarm optimization (PSO) [36] in the family of heuristic algorithms due to its high performance. We compare PSO with other heuristic algorithms in Section **??**. PSO is inspired by the social behavior observed by a group of animals in nature, such as fish and birds, that move in the space and share information about their best position. Technically, PSO solves an optimization problem by having a population of candidate solutions, called particles that move around in the search space seeking to maximize the fitness function:

$$\zeta = \sum_{(u,v)\in T} \left| D(u,v) - D_{o,x,y,z,\rho}(u,v) \right|, \tag{9}$$

which measures the difference between the modified and original depth maps.

Each particle has a position p and velocity $\dot{p}$ in the search space P_o. Note that the position p in P_o should not be confused with the coordinate in the world coordinate frame. Rather, it is a collective term consisting of the current state of the coordinate (x, y, z) and the pose ρ of the placed 3-D object. Note that the coordinate (x, y, z) of the 3-D object is constrained by the feasible region $\mathcal{F}(x, y, z)$, as determined by Eqn. (7) and (8). As such, we represent the position and velocity of the i-th particle in the search space as $p_i = (x_i, y_i, z_i, \rho_i)$ and $\dot{p}_i = (\dot{x}_i, \dot{y}_i, \dot{z}_i, \dot{\rho}_i)$, respectively. Each particle keeps track of its best-known personal position p_i^{pbest} and the best-known position of the entire swarm p^{gbest}. The particles update their positions and velocities at each iteration based on their p_i^{pbest}, p^{gbest}, and some random factors for exploration. We provide the details of π-Jack's PSO algorithm in **Algorithm 1**.

After placement of the 3-D object, we use HDR (high dynamic range) lighting [55] with the estimated skydome illumination to cast realistic shadows of the 3-D object on the ground plane $z = z_{\text{ground}}$ or any other modeled plane in the scene. We further employ a path-tracing renderer (e.g., Cycles [31]) to render the object with appropriate shaders and materials. Then, we composite it into the original image. Finally, we perform color correction and noise reduction to make the object blend seamlessly into the background image.

Algorithm 1: PSO algorithm for π-Jack.

```
Input: Scene image $I$, depth estimation model $\theta(\cdot)$,
       collection of 3-D objects $O$, target region $T$ in $I$
Output: An object $o^* \in O$ and a coordinate and pose
       $(x^*, y^*, z^*, \rho^*) \in P_o$ that maximize fitness function $\zeta$
Def stopping criterion $\varepsilon$ and acceleration constants $c_1, c_2$
Initialize the current best fitness function $\zeta^* = -\infty$
for each object $o \in O$ do
    Initialize a population of particles with random positions
      $p_i = (x_i, y_i, z_i, \rho_i)$ and velocities $\dot{p}_i = (\dot{x}_i, \dot{y}_i, \dot{z}_i, \dot{\rho}_i)$ in
      the search space of $P_o$
    Evaluate $i$-th particle's fitness: $\zeta_i = \zeta(p_i)$
    Set the individual best position of each particle to its
      current position: $p_i^{\text{pbest}} = p_i$
    Set the global best position to the position of the best
      particle: $p^{\text{gbest}} = \arg\max_{p_i} \zeta(p_i)$
    while $|p^{\text{gbest}} - p_i| > \varepsilon$ do
        for each particle do
            Update the velocity of the particle : $p_i \leftarrow$
              $\omega[\dot{p}]_i + c_1 r_1 (p_i^{\text{pbest}} - p_i) + c_2 r_2 (p^{\text{gbest}} - p_i)$,
              where $r_1, r_2$ are random numbers in $[0, 1]$ and
              $\omega$ is the inertia weight
            Update the position of the particle using the
              velocity: $p_i \leftarrow p_i + \dot{p}_i$
            Evaluate the fitness of the particle: $\zeta_i = \zeta(p_i)$
            if $\zeta_i > \zeta_i^{\text{pbest}}$ then
                Update the personal best position of the
                  particle: $p_i^{\text{pbest}} = p_i$
            if $\zeta_i > \zeta_i^{\text{gbest}}$ then
                Update the global best position: $p^{\text{gbest}} = p_i$
    if the attack object o's best fitness is better than the
      current best fitness then
        Store the best object, position, and pose:
          $o^*, x^*, y^*, z^*, \rho^* = o, x^{\text{gbest}}, y^{\text{gbest}}, z^{\text{gbest}}, \rho^{\text{gbest}}$
return $p^* = (o^*, x^*, y^*, z^*, \rho^*)$
```

4.4 Robust Design and Analysis

In this section, we first introduce an optimization technique called EoM&I to make π-Jack robust, then perform a mechanistic analysis to understand why π-Jack is intrinsically explainable and robust.

4.4.1 Expectation over Motion and Illumination

The above optimization of the 3-D object placement assumes a static scene. Nevertheless, the photos are dynamic and evolve over time due to the vehicle's motion and illumination variations, both being important aspects dictating image rendering and reflecting the inherent dynamics of AD systems and real-world driving conditions. Therefore, we employ the ideas in [5, 49] and formally create the expectation over motion

[1] The ground is used only as an example here,

and illumination (EoM&I). To implement EoM&I, we first define a distribution of photos captured along the vehicle's moving trajectory $\mathcal{M}$ to model the domain shift of the image due to different viewing angles in consecutive frames as the vehicle moves. To ensure that the 3-D object we place stays at the same location across all frames, we use the Kanade-Lucas-Tomasi (KLT) algorithm [43] to track the motion of several anchor points (e.g., corners/edges with high visibility and distinctiveness) in the feasible region using optical flow, and get a solution of the camera's motion. We further use the solved trajectory of the camera to align our 3-D object at the same position in the world frame across consecutive photos.

We also consider a distribution of different illumination conditions $\mathcal{G}$ to account for the possible error of illumination estimation in Section 4.2.2, as well as the shadow and brightness variations as time elapses. These illumination conditions include different sun directions $\mathbf{l}_s$, atmospheric turbidity τ, exposure ω, and zenith illumination L_z. Collectively, these parameters encapsulate all the components necessary for the models [12, 27] described in Eqns. (3) and (4). Since the potential estimation error and parameter difference are usually small, the new illumination condition is only slightly different from the original estimated one. Based on the above analysis, we can reformulate the objective of π-Jack attack as follows:

$$\underset{\substack{o\in O,\\ (x,y,z,\rho)\in P_o}}{\arg\max}\ \mathbb{E}_{\substack{m\sim\mathcal{M},\\ g\sim\mathcal{G}}}\left[\sum_{(u,v)\in T}\left|D(u,v)-D_{o,x,y,z,\rho}(u,v)\right|\right]. \quad (10)$$

In practice, it is unfeasible to explore all possible photos along the trajectory in $\mathcal{M}$ and illumination conditions in $\mathcal{G}$ due to high computational cost. Thus, we approximate the expectation in Eqn. (10) by averaging five consecutive frames and five randomly sampled variations of illumination conditions, resulting in $5\times 5 = 25$ images in total.

4.4.2 Mechanistic Analysis

The optimization goal of Eqn. (10) is to maximize the perceived depth difference before and after placing the object. Nonetheless, it is critical to understand that Eqn. (10) does not inherently predict whether the target will appear closer or further away following the optimization process. This indeterminacy is a byproduct of the optimization's heuristic nature, which introduces randomness to the attack outcome. Given these constraints, we perform a retrospective examination of the attack's impact, as opposed to making a priori predictions on the target's position. Through a careful examination of post-attack observations (refer to Figures 6 and 19 for concrete examples), we proceed with a posterior mechanistic analysis. This analysis reveals two ways in which π-Jack manipulates depth perception.

First, π-Jack is able to *creates* deceptive perspectives by establishing textural similarity between the target and inserted objects, and by blending structured lines of objects with existing linear perspectives. These approaches forge visual links between the target and the introduced objects, misleadingly suggesting *proximity* of the target. Second, π-Jack is also able to *disrupts* established perspectives by concealing critical target boundaries essential for MDE systems' depth calculations, and by disrupting the scene's natural perspective lines with contrasting linear patterns. These tactics weaken MDE's reliance on genuine perspective cues, associating the target with *distant* objects within the scene.

5 Implementation, Setup, and Metrics

In this section, we provide details on π-Jack's implementation, and introduce the experiment setup and metrics in our study.

5.1 System Implementation and Setup

We employ Blender 3.6 for 3-D modeling and rendering, and automate the modeling and rendering process with the Blender 3.6 Python API. All deep learning algorithms, including the illumination estimation and MDE models, are built upon PyTorch 1.7.1. For illumination estimation, we set $\alpha = 0.3$ empirically for an urban driving environment. In feasible region identification, the standard deviation of the Gaussian kernel is set to 10 pixels, and the threshold $T = 5$. For 3-D object placement, we limit the angles with respect to the three axes to less than 10° to ensure that the object appears natural and inconspicuous. We employ Python 3.7 to implement and analyze the PSO algorithm and the feasible regions, setting both acceleration constants of the PSO algorithm c_1 and c_2 to 2. For EoM&I, the turbidity variation is within a range of ± 0.5, the exposure variation is within a range of ± 0.005; the variation of the sun position is within a range of $\pm 5^\circ$ (equivalent to the sun moving 40 minutes); the variation of the zenith illumination is 100 lux. The camera-solving process is implemented by the Libmv [82] library.

5.2 Experiment Setup

We select 500 experiment AV scenes from the KITTI dataset [19], encompassing a broad spectrum of road conditions such as highways, crosswalks, and pedestrian areas. These scenes also include a variety of background objects like buildings, trees, trucks, smaller vehicles, traffic signs, and traffic lights. In addition to this, we gather our own AV scenes, both with and without strategically positioned objects, to evaluate π-Jack in real-world scenarios. For the MDE model, we choose to evaluate π-Jack using MonoDepth2 [21], DepthHints [67], and ManyDepth [68]. We make this selection based on their practicality, as they employ a self-supervised training strategy similar to the ones used by Tesla Autopilot [1, 17, 59] and Waymo [86], eliminating the need for ground-truth depth while achieving comparable accuracy with supervised training. Furthermore, they are open-sourced and readily available.

5.3 Metrics

Mean depth error. For a specific target region T in the image, the mean depth error is defined as the average discrepancy between the perceived depth of the original scene and the modified scene after π-Jack attack at each pixel $(u,v) \in T$.

Intersection over union (IoU). For the downstream task of road segmentation, we use IoU to quantify the overlap between ground truth and the prediction pixel sets as the ratio of their intersecting and union areas.

Average precision (AP). For the downstream task of 3-D vehicle detection, AP measures the accuracy of detection by averaging the precision scores at different intersections over IoU thresholds. A lower AP means that the model misses some or detects wrong vehicles due to π-Jack attack.

6 Attack Evaluation

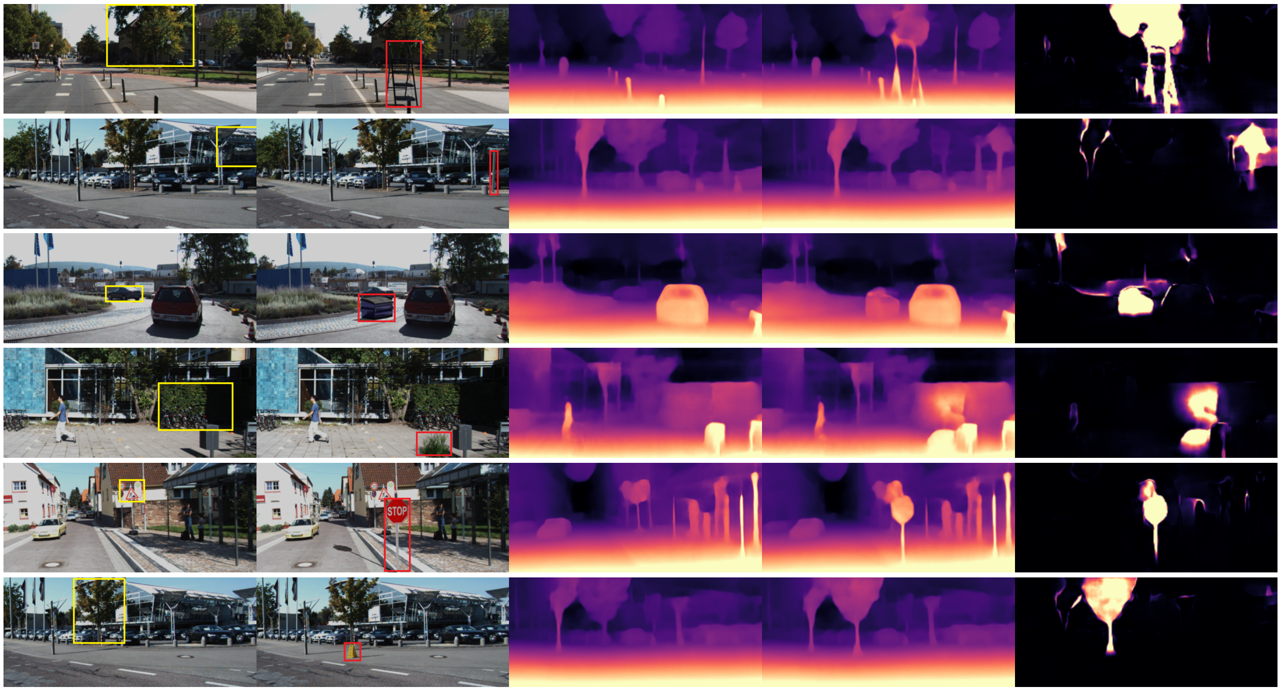
We begin by assessing the overall performance of π-Jack. Following this, we analyze the robustness of the attack under various practical factors, such as object size, and distance attack vectors. In addition, we conduct an ablation study to examine how π-Jack performs without the EoM&I techniques. Subsequently, we compare the PSO algorithm used by π-Jack with other optimization algorithms, and investigate the generalization capability of π-Jack to other models. Finally, we evaluate π-Jack with real-world scenes and attack vectors. We postpone extended discussions of impact of π-Jack on downstream tasks such as object detection and semantic segmentation in Appendix A.

6.1 Overall Performance

Figure 6 illustrates some examples of how π-Jack can manipulate depth estimation. In each example, the original scene is shown in Figure 6(a), with the target object highlighted by a yellow box. The modified scene after π-Jack attack is shown in Figure 6(b), with the attack vector highlighted by a red box. The depth maps of the original and modified scenes are shown in Figures 6(c) and 6(d), respectively. The absolute difference between the two depth maps is shown in Figure 6(e). It is important to understand that changes in depth across the entire picture are unintended byproducts of optimizing a localized objective function. Each row of Figure 6 demonstrates a different scenario and together these scenarios illustrate the effectiveness and diversity of π-Jack attack.

The first row shows how a ladder can trick the depth estimation system into believing that a faraway tree is almost as close as the ladder. The second row presents a case where a simple barrier pole can alter the perceived depth of a large portion of a building. The third row indicates that a garbage bin can affect the depth estimation of a moving vehicle, suggesting that downstream tasks such as 3-D vehicle detection are also vulnerable to π-Jack attack. The fourth row reveals that a clump of grass can deceive the MDE into thinking that a distant bush is much closer. It also shows that the attack vector and the target object do not need to touch in the image space for the π-Jack attack to succeed. The fifth row depicts how a "stop" sign can distort the depth of more distant traffic signs. In the sixth row, it can be observed that a small 3-D object such as a safety sign can successfully hijack the perspective of a much larger target object such as a remote tree.

We further evaluate π-Jack 's performance by measuring the mean depth error for various targets and attack vectors.



(a) Original scene. (b) Attacked scene. (c) Original depth estimation (d) Attacked depth estimation. (e) Depth difference.

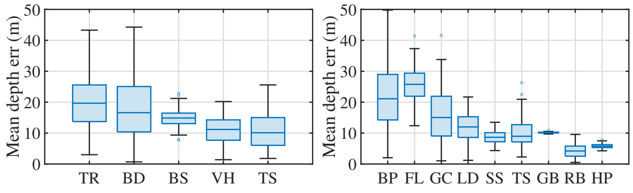
Figure 6: Example estimated depth maps before and after π-Jack attack.

The average depth error is 14.75 m, with detailed results reported in Figures 7(a) and 7(b). Figure 7(a) reveals that the medians for the mean depth errors of the target tree (TR), buildings (BD), bush (BS), vehicles (VH), and traffic signs (TS) are 19.68 m, 16.59 m, 14.87 m, 11.18 m, and 10.15 m, respectively. The reason behind the observation that trees are most vulnerable is that any pole-shaped 3-D object may hijack the depth estimation of the trunk of the tree significantly. Moreover, trees can also be attacked effectively because they have a texture that can be easily mimicked by plants such as a clump of grass. Buildings are also prone to π-Jack because they share many common structural features (e.g., parallel lines and rectangular patterns) with attack vectors. Although a bush has a similar texture to trees, its irregular shape and position make it less vulnerable to attack vectors in the feasible regions. Traffic signs and vehicles pose the most challenges for π-Jack due to their placement on perspective-rich roads, distance from feasible regions, and incongruent colors/textures with our attack vectors. Nonetheless, their median depth error exceeding 10 m represents a significant threat to AV-MDE.

Figure 7(b) shows the mean depth errors caused by different attack vectors. Barrier poles (BP), flags (FL), grass clumps (GC), ladders (LD), safety signs (SS), traffic signs (TS), garbage bins (GB), roadblocks (RB), and hydrant posts (HP) achieve medians for mean depth errors of 21.10 m, 25.77 m, 15.05 m, 12.00 m, 8.65 m, 8.973 m, 10.15 m, 4.21 m, and 5.76 m, respectively. The better performance of BP, FL, GC, and LD can be explained by Table 1. To be specific, poles and flags are the best because their superior extensibility allows them to connect and blend with distant targets; grass clump is the third best because its similar texture with distant trees and bushes enables it to blend in. Finally, ladders also perform well due to their fair extensibility and structural similarities with distant buildings.

6.2 Robustness Analysis

In this section, we analyze π-Jack's robustness to different real-life factors. These factors include the size of the 3-D object as well as the distance between the placed object and the AV camera. We postpone the robustness analysis of the vehicles displacement and sun's position to Section 6.3. We evaluate π-Jack on an Audi Q7 vehicle, utilizing a factory-installed front camera. The poses of the attack vectors are adjusted by placing foam and glue underneath.



(a) Target object.

(b) Attack vector.

Figure 7: Overall π-Jack performance.

6.2.1 Effect of Object Size

We measure the planar size (height × width) of the attack vectors and investigate their impacts on depth estimation errors. The results are shown in Figure 8. Overall, object size is not strongly correlated with depth error. However, one may observe some intriguing phenomena: an object that appears smaller ($0.2\,\mathrm{m}^2$) to the camera may have better attack performance than an object that appears larger ($1.4\,\mathrm{m}^2$). This phenomenon can be attributed to the characteristics of the objects involved. For example, more extensible objects such as barrier poles, though visually smaller, demonstrated greater effectiveness in comparison to bulkier objects like garbage bins, which lack extensibility, as analyzed in Section 6.1.

6.2.2 Effect of Distance

We investigate the impact of the distance from the placed 3-D object to the camera and present the results in Figure 9. It becomes evident that when the distance to the camera falls within the range of 5 m to 20 m, the performance of the π-Jack attack remains largely insensitive to the distance, resulting in medians for mean depth estimation errors of aroufnd 20 m. However, as the distance continues to increase, the effect of the attack starts to decrease, and at a distance of 40 m from the camera, the medians for π-Jack's induced mean depth errors can be as low as 3 m. The diminishing effectiveness of π-Jack can be explained by the fact that as the distance increases, the attack vectors become smaller to the camera, thereby causing their structure and texture to become less clear and reducing their ability to hijack the perspective of the targets.

6.2.3 Effect of Object Angle

We further explore the effect of the object's azimuth angle. Given the dependency of the optimal angle on the target's position, which can significantly vary, we define the optimal angle as 0° and adjust the azimuth angle in proximity to this reference point. The robustness analysis presented in Figure 10 reveals that even within a relatively broad azimuth range of $[-3^\circ, 3^\circ]$, the median depth error consistently exceeds 10 m, thereby affirming the effectiveness of our approach.

6.2.4 Effect of Object Position

Object distance and azimuth angle in Section 6.2.2 and 6.2.3 are sufficient to position an object. We further examine the influence of object placement contexts, including roads (RD),

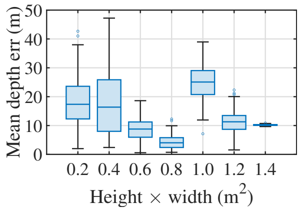


Figure 8: Impact of size.

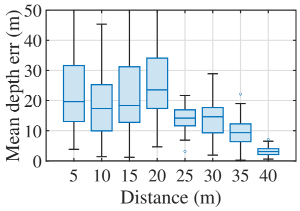


Figure 9: Impact of distance.

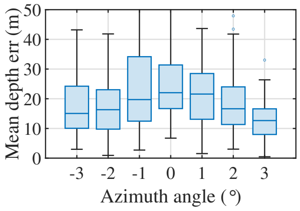


Figure 10: Impact of angle.

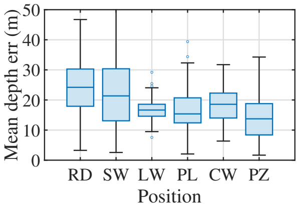


Figure 11: Impact of position.

sidewalks (SW), lawns (LW), parking lots (PL), crosswalks (CW), and plazas (PZ) in Figure 11. Our findings reveal that objects positioned on roads induce the greatest depth error. This can be attributed to the fact that roads are the most salient regions in camera views. However, while deploying attack vectors on roads might be most effective, such strategies are less feasible due to the potential for traffic disruption.

6.2.5 Effect of Illumination

We also study the impact of illumination, as shown in Figure 12. We evaluate the performance of π-Jack under environment illumination from 100lux to 1,000lux. The change of illumination is achieved by conducting experiments at different times of day. The results indicate that the depth errors remain relatively consistent, demonstrating limited sensitivity to changes in illumination.

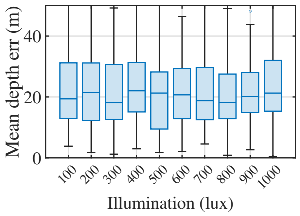


Figure 12: Impact of lux.

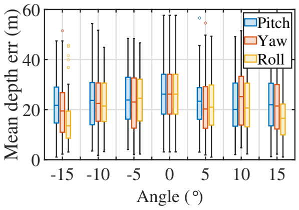


Figure 13: Impact of pose.

6.2.6 Effect of Object Pose

We also examine the effect of object pose variations (including pitch, yaw, and roll) in Figure 13. The results indicate that the system, π-Jack, sustains its high level of effectiveness across a broad range of $[-15°, 15°]$ for all three angles. Notably, changes in roll angle result in the most significant decrease in depth error, attributable to its pronounced change in the object's appearance in the camera's view.

6.2.7 Effect of Target Region Segmentation

We lastly compare the performance of manual target segmentation and SAM [37]. It turns out that the induced depth errors (on different targets and of different attack vectors) under these two segmentation methods are consistently smaller than 0.01m, demonstrating the equivalency of these two.

6.3 Ablation Study

In this section, we conduct ablation studies on the EoM and EoI processes, comparing the results with the original π-Jack pipeline. The targets and attack vectors are the same as in Section 6.1. Figures 14,15, and16 illustrate how EoM impacts π-Jack's performance during vehicle movement. EoM consistently achieves a median depth estimation above 10m, outperforming non-EoM cases by over 3m on average, showcasing π-Jack's effectiveness across various velocities. Notably, EoM manages frame differences of ±2 (at a 25m/s velocity, frame rate of 10) effectively, maintaining induced errors above 10m, while without EoM, errors drop below 4m. Similarly, EoM also handles lateral distances up to ±2 m effectively. Additionally, Figure 17 examines the effect of time of day on depth error estimation. Without EoI, depth error estimation decreases by 2m under varying illuminations. However, with EoI, π-Jack's performance remains stable across the day.

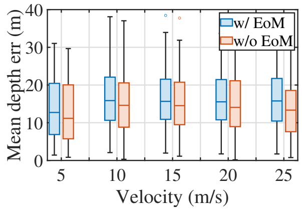


Figure 14: Velocity.

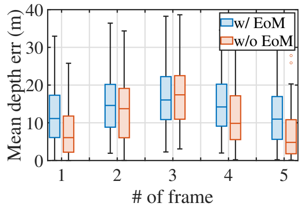


Figure 15: Number of frames.

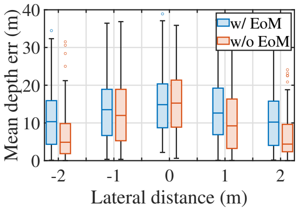


Figure 16: Lateral distance.

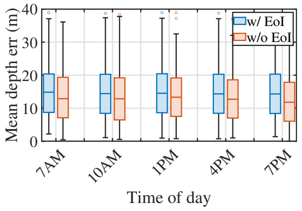
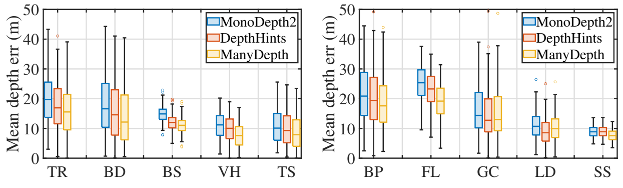


Figure 17: Time of day.

6.4 Generalization to Different MDE Models

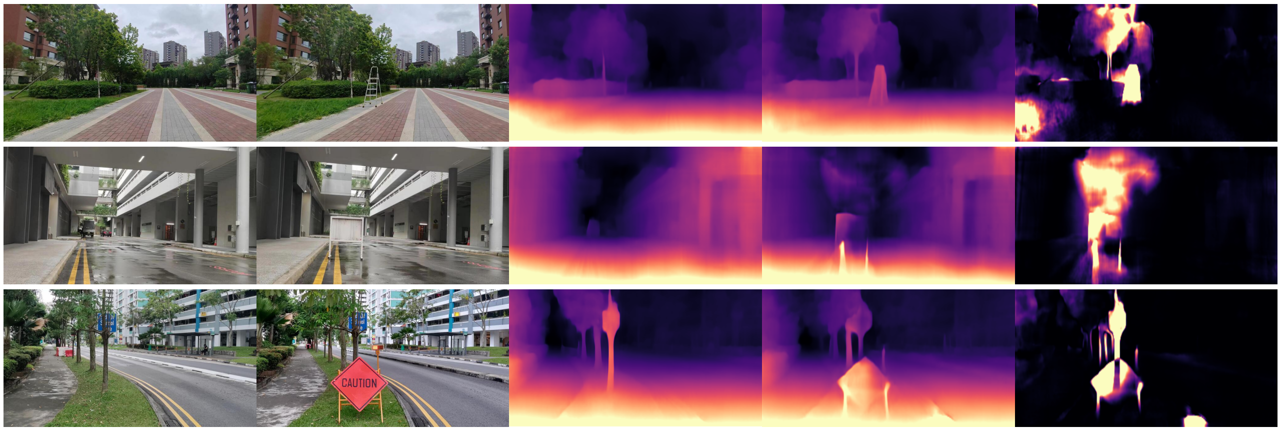
We believe π-Jack is a general attack not limited to any specific model. To support this claim, we assess the scenes π-Jack crafts for MonoDepth2 on two other models, DepthHints and ManyDepth, and present the results in Figure 18. It can be observed that generalization to DepthHints and ManyDepth renders the attack vectors slightly less effective, with the mean depth errors decreased by an average of 2m and 3m, respectively. Nonetheless, even after transferring the π-Jack attack to another model, it still poses a significant threat to the safety of AD. We believe that by optimizing the attack in a model-specific manner, the performance drop can be mitigated.



(a) Target object.

(b) Attack vector.

Figure 18: Generalizability of π-Jack to different models.

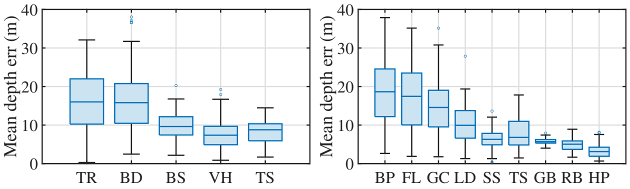


(a) Original scene. (b) Attacked scene. (c) Original depth estimation (d) Attacked depth estimation. (e) Depth difference.

Figure 19: Evaluation with real-world scenes and attack vectors.

6.5 Evaluation in the Wild

To evaluate π-Jack in real life, we conduct experiments in two types of locations to ensure there's no interference from others: 1) dedicated test sites, and 2) closed roads under construction. For the latter, we have secured the necessary approvals from appropriate authorities, guaranteeing that our experiments proceed without affecting public safety or violating any regulations. We collect a total 200 scenes. Figure 19(a) shows three representative scenes for visualization purposes. Figures 19(b), 19(c), 19(d), and 19(e) show the modified scenes, the depth maps of the original and modified scenes, and the map of depth difference, respectively. In the first scene with a tree as the attack target, π-Jack selects a ladder as the attack vector. It can be observed that the target tree is "hijacked" closer to the attack vector. In the second scene with a building including its skylight as the attack target, π-Jack selects a roadblock as the attack vector. It can be observed that the affected area is much larger than the first scene, probably due to the rectangular structure shared by the roadblock and a large portion of the building body. In the third scene, the attack target is a tree, and π-Jack selects a traffic sign "caution" as the attack vector. One may readily observe that the target tree is moved further away from the attack vector in the depth map, creating an "inverse" perspective hijacking effect. Rather than creating a false perspective and makes targets appear closer, π-Jack disrupts the perspective information of the tree by obscuring its root, and makes it blend with a more distant background tree.



(a) Target object. (b) Attack vector.

Figure 20: Summary of real-world evaluations.

We further present the performance statistics in various scenes. The average mean depth error is 10.14 m, as illustrated in Figure 20. In real-world scenarios, the induced depth estimation errors are slightly smaller than those in the composited images, as shown in Figure 7. These differences can be attributed to the bias brought by a smaller number of real-world scenes requiring laborious placement. However, the results across the targets and attack vectors exhibit a similar trend and align with the outcomes obtained from composited images. The results with real-world scenes and objects show that the success of π-Jack does not stem from peculiarities or artifacts in the workflow. Instead, perspective hijacking is a feasible and effective attack in the physical world.

7 Security Analysis and Defense Discussion

7.1 Security Analysis

Since modern AVs adhere to the "Sense-Plan-Act" design paradigm [18], any malfunction occurring in the MDE module (residing in the "Sense" layer), which is targeted by π-Jack, will propagate to the upper layers. To analyze the security implications of π-Jack in the entire AD stack, we conduct end-to-end simulations using the AWSIM [83] and Autoware [60]. We employ the Tokyo West Shinjuku map and a customized Dallara AV-21R vehicle. The simulation features 30 NPCs and uses a traffic seed value of 20. Autoware employs the 3-D object detection algorithm BEVDet [28]. We implement the π-Jack attack by introducing a 3-D model of the attack vectors into the simulation environment. It is important to note that we do not aim to involve performing evaluations on a real AV in the physical world due to the associated costs and safety considerations.

Our findings from the simulation are quite concerning. Even in a traffic-free scenario, π-Jack lead to high incidences of improper lane changes and abrupt braking at rates of 63.4% and 55.0%, respectively. Furthermore, these flawed decision-makings result in a 23.9% probability of the vehicle diverting off its course and colliding with other objects. When introducing traffic into the scene, the compromised MDE results in a 42.4% likelihood of collisions with other vehicles or pedestrians. In stark contrast, the rates of improper lane changes, abrupt stops, and collisions are all 0% in the absence of the π-Jack attack. Our observations clearly illustrate that the repercussions of the π-Jack system are not confined to the "Sense" layer alone, but extend throughout the entire stack of AD systems. This highlights the critical need for both rigorous unit testing and comprehensive strategies to address MDE-related security issues, as indicated by AD industry standards [30].

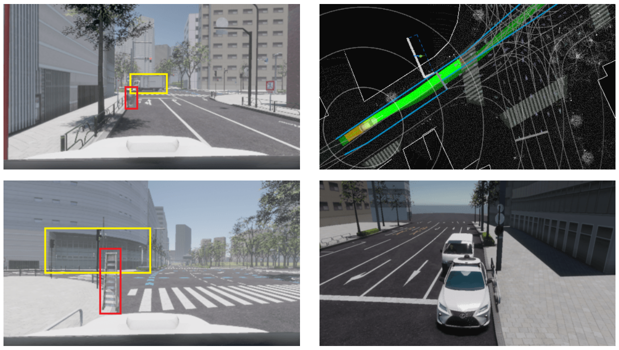

(a) AV scene. (b) π-Jack's effect.

Figure 21: Example scenes from AWSIM and Autoware.

We illustrate two example scenes from AWSIM and Autoware in Figure 21. Attack vectors and targets are highlighted with red and yellow bounding boxes, respectively. In the first scene, a garbage bin is used as an attack vector to target a distant truck. As shown in planning, the AV system erroneously perceives the truck as being on the roadside and changes its lane to avoid it. In the second scene, a ladder is employed to target a distant building. The AV system mistakenly thinks the building is blocking the road and decides to brake abruptly, resulting in a collision with another vehicle behind, as shown in the rotate-around scene.

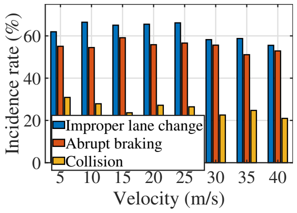


Figure 22: Impact of velocity.

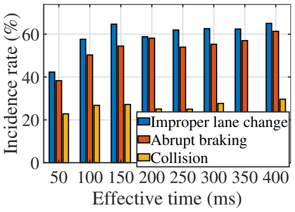


Figure 23: Impact of time.

Velocity and timing also influence the rate of flawed decision-making. In Figure 22, it is shown that the accident rate remains consistent for velocities under 35 m/s but slightly decreases at higher velocities, likely due to the diminished temporal effect of π-Jack on the AV system. Consequently, we further assess the notion of effective time, which we define as the duration an object remains within the target's salient region. The region's boundary is determined by setting a threshold at 10% of the maximum value found in the saliency map. Figure 23 illustrates that when the effective time is below 100 ms, the incidence rate decreases, probably due to the limited perception rate. We leave π-Jack's evaluation on more AD systems such as LGSVL and Apollo to future works.

Table 2: Effects of defense by adversarial training.

Mean depth error (m)	Adversarial training (conventional)	Adversarial training ([10])
MonoDepth2	13.87 (↓ 0.88)	12.33 (↓ 2.42)
DepthHints	10.93 (↓ 1.89)	9.91 (↓ 2.91)
ManyDepth	9.61 (↓ 1.69)	8.72 (↓ 2.58)

7.2 Potential Defense Methods

One potential defense method to defend against the perspective hijacking attack is conventional adversarial training. The idea is to collect AV data with adversarial examples containing 3-D objects in the model's training stage to improve the model's robustness. We follow the steps in Section 4 to place 100 effective attack vectors, and use the KLT algorithm in Section 4.4 to insert the 3-D objects into consecutive camera frames. We anticipate that this kind of training can make the MDE model less sensitive to perspective hijacking. We train the MDE model with this method and test its performance against π-Jack attacks using composited images. Table 2 indicates that this defense can improve the MDE model's robustness and raise the attack difficulty. However, the 0.88 m, 1.89 m, and 1.69 m decreases in mean depth error are not very impressive, probably because the limited number of attack vectors we craft cannot cover all vulnerabilities in the feasible region. We leave a more effective defense based on adversarial training to future work.

Besides conventional adversarial training, a novel strategy specifically tailored for self-supervised MDE is introduced by [10]. This method distinguishes itself from the conventional approach by inserting adversarial objects into only the latter of two consecutive frames during self-supervised training. This selective placement can prevent the model from converging on inaccurate yet robust depth estimations, which might occur when 3D objects are employed across all frames. By ensuring depth estimation consistency between consecutive frames with the adversarial object appearing in just the latter, this approach effectively removes the constant bias introduced by an adversarial object. Table 2 shows that this refined defense mechanism enhances the robustness of MDE models, further reducing the depth error by 2.42, 2.91, and 2.58 m for three MDE models, outperforming conventional adversarial training. It should be noted that the refined defense's success against stronger patch attacks relies on the MDE model's familiarity with the attacks, not their strength. In the π-Jack attack, only a few attributes (e.g., pose and distance) are manipulated, unlike patch attacks where every pixel can change. This limitation reduces the defense's effectiveness in adapting the MDE model to π-Jack attacks compared to patch attacks. However, the effectiveness of both adversarial training methods remain somewhat limited due to the constrained number of attack vectors available, underscoring the efficacy of the proposed π-Jack attack.

A different approach to defend against π-Jack is adversarial detection. The method identifies and rejects adversarial inputs before feeding them to the model. For example, the AV-MDE system may use auxiliary classifiers to detect anomalous inputs that may contain perspective-hijacking patterns. Another possible defense mechanism is to use multiple sensors or modalities to estimate depth, such as lidar and radar that directly measure the distance of objects. By fusing the information from different sources, AV can reduce the dependence on a single cue and increase the robustness to perspective hijacking. However, these methods face additional challenges such as sensor calibration, data alignment, and computational cost. Moreover, they might be infeasible as some car manufacturers such as Tesla rely on the all-vision AV solution [1, 59].

7.3 Safety Considerations

Since most vision-based commercial AD/robotics systems rely on either explicit or implicit depth estimation in their object detection and segmentation modules, they should be universally vulnerable to the π-Jack attack. Testing all of these systems is beyond the capability of the authors, so we leave the case-by-case investigation to future research. Section 7.2 presents defenses that, upon enhancement, could potentially counter the π-Jack attack effectively. It is essential to emphasize that the exploration of the π-Jack attack adheres to ethical research standards, with no intention to facilitate real-world attacks against AVs. Instead, by bringing the π-Jack attack to light, we seek to alert the academic community to this vulnerability, encouraging a concerted research effort to safeguard against these threats. This proactive disclosure is intended to preemptively counteract malevolent entities from exploiting these vulnerabilities in real-world scenarios, ensuring the advancement of cybersecurity in the AV sector.

8 Related Work

8.1 Attack on Autonomous Driving Perception

AD systems rely on sensors such as camera, lidar, radar, ultrasonic, and IMU for perception [76, 78, 79]. However, these sensors are vulnerable to spoofing (i.e., broadcasting false or modified signals that cause the sensors to produce incorrect results) [3, 6, 48, 52, 64, 72, 73] and jamming (i.e., overpowering the signals with noise that overwhelm the receiver) [52, 72] attacks. These attacks target vulnerabilities at the signal processing level, without exploiting the algorithms that process the sensor information. In contrast, more recent attacks focus on higher levels (e.g., AD perception and autonomy level) by exploiting vulnerabilities intrinsic to AD software and algorithms. Some examples include attacks on camera/lidar object detection [6, 8, 54, 75], tracking [33], localization [51], lane detection [34], intrusion detection [58], steering angle detection [11], and end-to-end AD [62].

8.2 Digital Adversarial Attacks

Conventional adversarial attacks typically occur in the digital domain, where crafted perturbations are directly fed to the model. These perturbations are usually bounded by the l_p norm [7, 22, 44, 77] to remain imperceptible. Digital adversarial attacks can be either white-box or black-box, depending on the attacker's access to the model. In white-box attacks, the attacker has full knowledge of the model (e.g., input, output, weights) and can use the model's gradient to generate adversarial perturbations [7, 22, 40, 44, 46]. In black-box attacks, the attacker can only input data and observe the model's output, without knowing the model's architecture and parameters. However, even in black-box settings, attackers can exploit the transferability of perturbations to similar models [13, 45, 63] or estimate model information through multiple queries [29].

8.3 Physical-World Adversarial Attacks

In contrast to digital domain attacks, physical-world adversarial attacks on MDE are more realistic yet more challenging, as they require robust perturbations that can survive various environmental factors (e.g., lighting, weather, distance, and angle) and affect the 3D scene understanding of autonomous systems. Previous works have investigated the "sticker-pasting" strategy [15] on the *pixel level* to attach stickers on traffic signs [15], cars [8, 9, 71], and roads [71] to mislead deep neural networks such as classfiers [4, 15], object detectors [8, 8, 16, 54, 61, 75], and depth estimation networks [9, 15]. These attacks leverage techniques such as style transfer [9] and EoT [2, 32] to make the stickers less noticeable and more robust to real-world conditions.

More recent physical-world attacks exploit light and shadow phenomena to achieve non-invasive attacks without physically modifying target objects or scenes. For instance, AdvLB [14] and DoubleStar [81] use light beams directly as attack vectors, proving to be effective physical-world attacks on DNNs. OPAD [20] deceives classifiers by projecting structured illumination with a low-cost projector. Zhong et al.[80] use natural-looking shadows to mislead traffic sign classifiers without alerting humans. Sayles et al.[50] leverage the rolling shutter effect to generate desired outputs. Most of these attacks work by transferring pixel-level attacks to the physical world, except for [34], which operates on the scene level by placing line markings on the road.

9 Conclusion

In this paper, we propose π-Jack as the first physical adversarial attack on AV-MDE systems utilizing perspective hijacking. Exploiting ordinary 3-D objects as attack vectors, π-Jack offers superior effectiveness, robustness, accessibility, and inconspicuity. Our experiments validate the high attack success rate and large depth difference achieved by π-Jack, demonstrating its successful application in both composited and real-world AV scenes. We also explore π-Jack's extensions to two downstream tasks of road segmentation and 3-D vehicle detection. Our results expose critical vulnerabilities in widely-used AV-MDE models and hence underscore an urgent need for enhanced security measures against such risks.

Acknowledgments

Tianyue Zheng is funded by the research start-up grant from the Southern University of Science and Technology and the Shenzhen Higher Education Institutions Stable Support Program (No. 20231120215201001).

References

[1] A. Karpathy. AI for Full-Self Driving at Tesla. https://www.youtube.com/watch?v=hx7BXih7zx8, 2023. Online; accessed 8 June 2023.

[2] A. Athalye et al. Synthesizing Robust Adversarial Examples. In *Proc. of the 35th ICML*, pages 284–293, 2018.

[3] S. H. V. Bhupathiraju et al. EMI-LiDAR: Uncovering Vulnerabilities of LiDAR Sensors in Autonomous Driving Setting using Electromagnetic Interference. In *Proc. of the 16th ACM WiSec*, pages 329–340, 2023.

[4] T. B. Brown et al. Adversarial Patch. *arXiv preprint arXiv:1712.09665*, 2017.

[5] Y. Cao et al. Invisible for Both Camera and Lidar: Security of Multi-Sensor Fusion based Perception in Autonomous Driving under Physical-World Attacks. In *Proc. of the 42nd IEEE S&P*, pages 176–194, 2021.

[6] Y. Cao et al. Adversarial Sensor Attack on Lidar-based Perception in Autonomous Driving. In *Proc. of the 26th ACM CCS*, pages 2267–2281, 2019.

[7] N. Carlini and D. Wagner. Towards Evaluating the Robustness of Neural Networks. In *Proc. of the 38th IEEE S&P*, pages 39–57, 2017.

[8] S.-T. Chen et al. ShapeShifter: Robust Physical Adversarial Attack on Faster R-CNN Object Detector. In *Prof. of ECML PKDD*, pages 52–68. Springer, 2019.

[9] Z. Cheng et al. Physical Attack on Monocular Depth Estimation with Optimal Adversarial Patches. In *Proc. of the 16th ECCV*, pages 514–532. Springer, 2022.

[10] Z. Cheng et al. Adversarial Training of Self-supervised Monocular Depth Estimation against Physical-world Attacks. In *Proc. of the 11st ICLR*, pages 1–19, 2023.

[11] A. Chernikova et al. Are Self-driving Cars Secure? Evasion Attacks against Deep Neural Networks for Steering Angle Prediction. In *Proc. of IEEE SPW*, pages 132–137. IEEE, 2019.

[12] S. Darula and R. Kittler. CIE General Sky Standard Defining Luminance Distributions. *Proceedings eSim*, 11:13, 2002.

[13] A. Demontis et al. Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks. In *Proc. of the 28th USENIX Security*, pages 321–338, 2019.

[14] R. Duan et al. Adversarial Laser Beam: Effective Physical-World Attack to DNNs in a Blink. In *Proc. of the 34th IEEE/CVF CVPR*, pages 16062–16071, 2021.

[15] K. Eykholt et al. Robust Physical-World Attacks on Deep Learning Visual Classification. In *Proc. of the 31st IEEE/CVF CVPR*, pages 1625–1634, 2018.

[16] K. Eykholt et al. Note on Attacking Object Detectors with Adversarial Stickers. *arXiv preprint arXiv:1712.08062*, 2017.

[17] F. Lambert. Hacker Shows What Tesla Full Self-Driving's Vision Depth Perception Neural Net Can See. https://electrek.co/2021/07/07/hacker-tesla-full-self-drivings-vision-depth-perception-neural-net-can-see/, 2021. Online; accessed 8 June 2023.

[18] E. Gat et al. On Three-layer Architectures. *Artificial Intelligence and Mobile Robots*, 195:210, 1998.

[19] A. Geiger et al. Vision Meets Robotics: The KITTI Dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[20] A. Gnanasambandam et al. Optical Adversarial Attack. In *Proc. of the 36th IEEE/CVF ICCV*, pages 92–101, 2021.

[21] C. Godard et al. Digging into Self-supervised Monocular Depth Estimation. In *Proc. of the 34th IEEE/CVF ICCV*, pages 3828–3838, 2019.

[22] I. J. Goodfellow et al. Explaining and Harnessing Adversarial Examples. In *Proc. of the 3rd ICLR*, pages 1–11, 2015.

[23] R. L. Gregory and P. Heard. Border Locking and the Café Wall Illusion. *Perception*, 8(4):365–380, 1979.

[24] C. Griwodz et al. AliceVision Meshroom: An Open-source 3D Reconstruction Pipeline. In *Proc. of the 12th ACM MMSys*, pages 241–247, 2021.

[25] V. Guizilini et al. 3D Packing for Self-Supervised Monocular Depth Estimation. In *Prof. of the 33rd IEEE/CVF CVPR*, pages 2485–2494, 2020.

[26] Y. Hold-Geoffroy et al. Deep Outdoor Illumination Estimation. In *Proc. of the 30th IEEE/CVF CVPR*, pages 7312–7321, 2017.

[27] L. Hosek and A. Wilkie. An Analytic Model for Full Spectral Sky-dome Radiance. *ACM Transactions on Graphics (TOG)*, 31(4):1–9, 2012.

[28] J. Huang et al. BEVDet: High-performance Multi-camera 3D Object Detection in Bird-eye-view. *arXiv preprint arXiv:2112.11790*, 2021.

[29] A. Ilyas et al. Black-box Adversarial Attacks with Limited Queries and Information. In *Proc. of the 35th ICML*, pages 2137–2146. PMLR, 2018.

[30] I. O. for Standardization. ISO 26262: Road Vehicles – Functional Safety. https://www.iso.org/standard/68383.html, 2011. Online; accessed 8 June 2023.

[31] B. Iraci. *Blender Cycles: Lighting and Rendering Cookbook*. Packt Publishing Ltd, 2013.

[32] S. T. Jan et al. Connecting the Digital and Physical World: Improving the Robustness of Adversarial Attacks. In *Proc. of the 33rd AAAI*, volume 33, pages 962–969, 2019.

[33] Y. J. Jia et al. Fooling Detection Alone Is Not Enough: Adversarial Attack against Multiple Object Tracking. In *Proc. of the 8th ICLR*, 2020.

[34] P. Jing et al. Too Good to Be Safe: Tricking Lane Detection in Autonomous Driving with Crafted Perturbations. In *Proc. of the 30th USENIX Security*, pages 3237–3254, 2021.

[35] H. R. Kang. *Computational Color Technology*. SPIE Press Bellingham, 2006.

[36] J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Proc. of ICNN*, volume 4, pages 1942–1948. IEEE, 1995.

[37] A. Kirillov et al. Segment Anything. *arXiv preprint arXiv:2304.02643*, 2023.

[38] V. R. Kumar et al. SynDistNet: Self-Supervised Monocular Fisheye Camera Distance Estimation Synergized with Semantic Segmentation for Autonomous Driving. In *Proc. of the IEEE/CVF WACV*, pages 61–71, 2021.

[39] H. Leibowitz et al. Ponzo Perspective Illusion as A Manifestation of Space Perception. *Science*, 166(3909):1174–1176, 1969.

[40] S. Li et al. Adversarial Perturbations against Real-time Video Classification Systems. In *Proc. of the 26th NDSS*, 2019.

[41] X. Liu et al. Learning Auxiliary Monocular Contexts Helps Monocular 3D Object Detection. In *Proc. of the 36th AAAI*, volume 36, pages 1810–1818, 2022.

[42] J. Long et al. Fully Convolutional Networks for Semantic Segmentation. In *Proc. of the 28th IEEE/CVF CVPR*, pages 3431–3440, 2015.

[43] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proc. of the 7th IJCAI*, volume 2, pages 674–679, 1981.

[44] A. Madry et al. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[45] N. Papernot et al. Transferability in Machine Learning: From Phenomena to Black-box Attacks Using Adversarial Samples. *arXiv preprint arXiv:1605.07277*, 2016.

[46] N. Papernot et al. The Limitations of Deep Learning in Adversarial Settings. In *Proc. of the 1st IEEE Euro S&P*, pages 372–387, 2016.

[47] V. Petsiuk et al. Black-box Explanation of Object Detectors via Saliency Maps. In *Proc. of the 34th IEEE/CVF CVPR*, pages 11443–11452, 2021.

[48] R. Quinonez et al. SAVIOR: Securing Autonomous Vehicles with Robust Physical Invariants. In *Proc. of the 29th USENIX Security*, pages 895–912, 2020.

[49] T. Sato et al. Dirty Road Can Attack: Security of Deep Learning based Automated Lane Centering under {Physical-World} Attack. In *Proc. of the 30th USENIX Security*, pages 3309–3326, 2021.

[50] A. Sayles et al. Invisible Perturbations: Physical Adversarial Examples Exploiting the Rolling Shutter Effect. In *Proc. of the 34th IEEE/CVF CVPR*, pages 14666–14675, 2021.

[51] J. Shen et al. Drift with Devil: Security of Multi-Sensor Fusion based Localization in High-Level Autonomous Driving under GPS Spoofing. In *Proc. of the 29th USENIX Security*, pages 931–948, 2020.

[52] H. Shin et al. Illusion and Dazzle: Adversarial Optical Channel Exploits against Lidars for Automotive Applications. In *Proc. of the 19th CHES*, pages 445–467. Springer, 2017.

[53] K. Simonyan et al. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv preprint arXiv:1312.6034*, 2013.

[54] D. Song et al. Physical Adversarial Examples for Object Detectors. In *Proc. of the 12th USENIX WOOT*, 2018.

[55] J. Stumpfel et al. Direct HDR Capture of the Sun and Sky. In *ACM SIGGRAPH 2006 Courses*, pages 1–5. 2006.

[56] P. Sturm et al. Camera Models and Fundamental Concepts used in Geometric Computer Vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(1–2):1–183, 2011.

[57] R. T. Surdick et al. Relevant Cues for the Visual Perception of Depth: Is Where You See It Where It Is? In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 38, pages 1305–1309, 1994.

[58] K. Z. Teng et al. PAID: Perturbed Image Attacks Analysis and Intrusion Detection Mechanism for Autonomous Driving Systems. In *Proc of the 9th CPSS Workshop*, pages 3–13, 2023.

[59] Tesla. AI & Robotics. `https://www.tesla.com/AI`, 2023. Online; accessed 8 June 2023.

[60] T. A. Foundation. Autoware. `https://autoware.org/`, 2024. Online; accessed 8 May 2024.

[61] S. Thys et al. Fooling Automated Surveillance Cameras: Adversarial Patches to Attack Person Detection. In *Proc. of the 32nd IEEE/CVF CVPR Workshops*, pages 1–7, 2019.

[62] Y. Tian et al. DeepTest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars. In *Proc. of the 40th ACM ICSE*, pages 303–314, 2018.

[63] F. Tramèr et al. The Space of Transferable Adversarial Examples. *arXiv preprint arXiv:1704.03453*, 2017.

[64] Y. Tu et al. Injected and Delivered: Fabricating Implicit Control over Actuation Systems by Spoofing Inertial Sensors. In *Proc. of the 27th USENIX Security*, pages 1545–1562, 2018.

[65] J. Van Brummelen et al. Autonomous Vehicle Perception: The Technology of Today and Tomorrow. *Transportation Research Part C: Emerging Technologies*, 89:384–406, 2018.

[66] L. Von Stumberg et al. LM-Reloc: Levenberg-Marquardt based Direct Visual Relocalization. In *Proc. of 3DV*, pages 968–977. IEEE, 2020.

[67] J. Watson et al. Self-supervised Monocular Depth Hints. In *Proc. of the 34th IEEE/CVF ICCV*, pages 2162–2171, 2019.

[68] J. Watson et al. The Temporal Opportunist: Self-supervised Multi-frame Monocular Depth. In *Proc. of the 34th IEEE/CVF CVPR*, pages 1164–1174, 2021.

[69] F. Wimbauer et al. MonoRec: Semi-supervised Dense Reconstruction in Dynamic Environments from a Single Moving Camera. In *Proc. of the 13rd IEEE/CVF CVPR*, pages 6112–6122, 2021.

[70] A. Wong et al. Targeted Adversarial Perturbations for Monocular Depth Prediction. *Proc. of the 34th NeurIPS*, 33:8486–8497, 2020.

[71] K. Yamanaka et al. Adversarial Patch Attacks on Monocular Depth Estimation Networks. *IEEE Access*, 8:179094–179104, 2020.

[72] C. Yan et al. Can You Trust Autonomous Vehicles: Contactless Attacks against Sensors of Self-driving Vehicle. *Def Con*, 24(8):109, 2016.

[73] Y. Cao et al. You Can't See Me: Physical Removal Attacks on LiDAR-based Autonomous Vehicles Driving Frameworks. In *Proc. of the 32nd USENIX Security*, 2023.

[74] Z. Zhang et al. Adversarial Attacks on Monocular Depth Estimation. *arXiv preprint arXiv:2003.10315*, 2020.

[75] Y. Zhao et al. Seeing isn't Believing: Towards More Robust Adversarial Attack against Real World Object Detectors. In *Proc. of the 26th ACM CCS*, pages 1989–2004, 2019.

[76] T. Zheng et al. V^{2}iFi: in-Vehicle Vital Sign Monitoring via Compact RF Sensing. In *Proc. of the 20th ACM UbiComp*, pages 70:1–27, 2020.

[77] T. Zheng et al. Adv-4-Adv: Thwarting Changing Adversarial Perturbations via Adversarial Domain Adaptation. *Neurocomputing*, 569:127114, 2024.

[78] T. Zheng et al. Enhancing RF Sensing with Deep Learning: A Layered Approach. *IEEE Communications Magazine*, 59(2):70–76, 2021.

[79] T. Zheng et al. AutoFed: Heterogeneity-aware Federated Multimodal Learning for Robust Autonomous Driving. In *Proc. of the 29th ACM MobiCom*, pages 1–15, 2023.

[80] Y. Zhong et al. Shadows Can Be Dangerous: Stealthy and Effective Physical-World Adversarial Attack by Natural Phenomenon. In *Proc. the 35th IEEE/CVF CVPR*, pages 15345–15354, 2022.

[81] C. Zhou et al. DoubleStar: Long-Range Attack Towards Depth Estimation based Obstacle Avoidance in Autonomous Systems. In *Proc. of the 31st USENIX Security Symposium*, pages 1885–1902, 2022.

[82] libmv. Libmv Multiview Reconstruction and Tracking Library. https://projects.blender.org/blender/libmv, 2023. Online; accessed 8 June 2023.

[83] tier4. AWSIM Document. https://tier4.github.io/AWSIM/Introduction/AWSIM/, 2024. Online; accessed 8 May 2024.

[84] Toyota Research Institute. Monocular Depth in the Real World. https://medium.com/toyotaresearch/monocular-depth-in-the-real-world-99c2b287df34, 2023. Online; accessed 8 June 2023.

[85] Waymo LLC. Depth Estimation Matters Most: Improving Per-Object Depth Estimation for Monocular 3D Detection and Tracking. https://waymo.com/research/depth-estimation-matters-most-improving-per-object-depth-estimation-for-monocular-3d-detection-and-/, 2023. Online; accessed 8 June 2023.

[86] Waymo LLC. Unsupervised Monocular Depth Learning in Dynamic Scenes. https://waymo.com/research/unsupervised-monocular-depth-learning-in-dynamic-scenes/, 2023. Online; accessed 8 June 2023.

A Impact on Downstream Tasks

In this section, we evaluate π-Jack's impact on road segmentation and 3-D vehicle detection.

A.1 Road Segmentation

Road segmentation is critical for AD, enabling the vehicle's AI to distinguish drivable from non-drivable areas for safe navigation and accident prevention. Moreover, it assists in comprehending the environment for effective decision-making and planning. Attacks on depth estimation can severely affect the segmentation by misleading the vehicle on object distance and location. This can result in an erroneous interpretation of drivable areas, leading to unsafe driving conditions. To demonstrate π-Jack's impact on road segmentation, we employ the widely used fully convolutional networks to generate road segmentation before and after π-Jack attack. As illustrated in Figures 24(a) and 24(b), the introduction of a flag in the first row of Figure 24 disrupts the perspective of a distant tree, causing a complete failure in the segmentation between the flag and the tree. Similarly, the second row demonstrates how a clump of grass disrupts the perspectives of both distant trees and nearby lawns, leading to the recognition of a large "forbidden" area on the road.

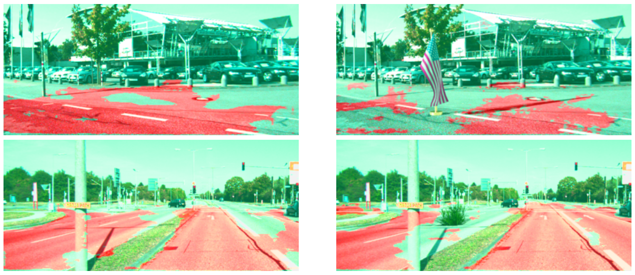
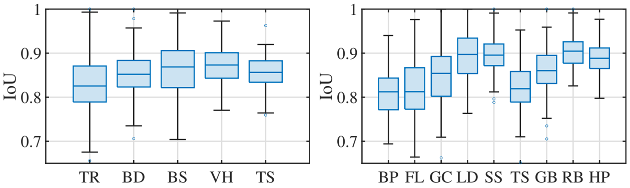

(a) Original scene. (b) Attacked scene.

Figure 24: Road segmentation.

We further quantitatively assess how π-Jack affects road segmentation. Without the π-Jack, the average IoU between the segmented road and the ground truth is 0.942. However, upon launching the π-Jack attack, the average IoU drops significantly to 0.867. The detailed IoU values for different targets and attack vectors are illustrated in Figures 25(a) and 25(b), respectively. It becomes evident that there is a negative correlation between the IoU and mean depth error induced by the π-Jack attack (shown in Figure 7), firmly indicating that the hijacked depth disrupts the downstream task of road segmentation.



(a) Target object. (b) Attack vector.

Figure 25: IoU of segmented road before and after attack.

A.2 3-D Vehicle Detection

3-D vehicle detection is another important task of AD, as it allows the vehicle to accurately perceive and locate surrounding vehicles in 3-D space, which is essential for safe navigation and decision-making. Given that 3-D vehicle detection largely depends on depth information, a π-Jack attack can significantly compromise the accuracy and reliability of the detection system. This might lead to incorrect or incomplete object detection, thereby jeopardizing the safety of the AV and its passengers. We use the state-of-the-art MonoCon [41] to perform vehicle detection. Figure 26 illustrates the outcome of vehicle detection, with the first two rows presenting the 3-D bounding boxes before and after the attack, and the last two rows showing the detection results from a bird's-eye view. It can be clearly seen from Figure 26(a) how the introduction of a garbage bin can alter the perspective of a distant vehicle by approximately 5 m. Similarly, in Figure 26(b), a ladder can distort a vehicle's perspective even without overlapping with it in the image space.

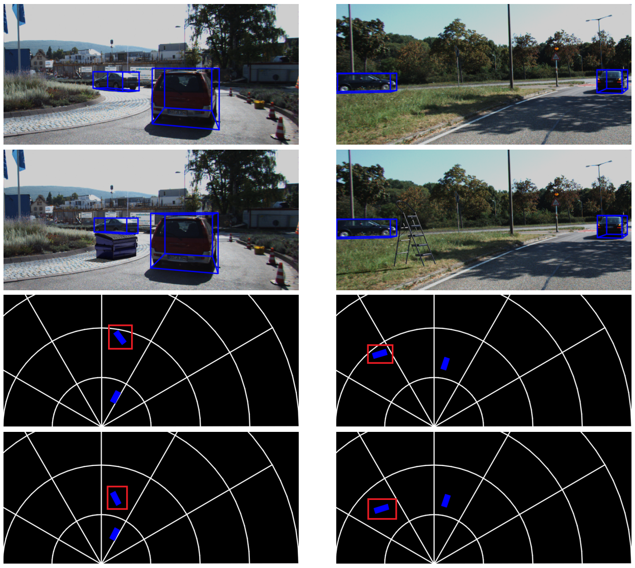
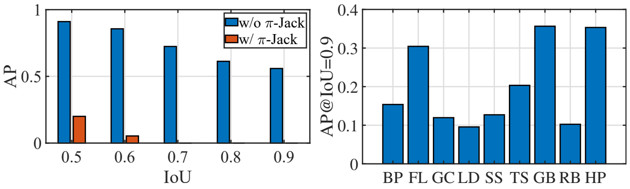

(a) Example scene 1. (b) Example scene 2.

Figure 26: 3-D vehicle detection.



(a) IoU threshold. (b) Attack vector.

Figure 27: AP of detected vehicle before and after attack.

We further evaluate how π-Jack affects 3-D vehicle detection quantitatively. As shown in Figure 27(a), without the π-Jack attack, the AP of vehicle detection at IoU thresholds from 0.5 to 0.9 are 91.04%, 85.63%, 72.38%, 61.27%, and 55.88%, respectively. However, upon launching the π-Jack attack, the AP decreases to 20.3%, 5.33%, 0%, 0%, and 0%, respectively, indicating an AP drop of more than 71%. We further analyze how different attack vectors affect the AP of vehicle detection, focusing on AP@IoU=0.5 due to the small APs at IoU thresholds from 0.5 and 0.9. As shown in Figure 27(b), we determine that the most effective attack vectors for vehicle detection are grass clump, ladder, and roadblock, with APs of 11.96%, 9.56%, and 10.25%, respectively. The prominent performance of these attack vectors can be attributed to their similar structures and locations with vehicles on the road.