Fractured Glass, Failing Cameras: Simulating Physics-Based Adversarial Samples for Autonomous Driving Systems

Manav Prabhakar,Jwalandhar Girnar,ArpanKusari[*]University of Michigan Transportation Research Institute 2901 Baxter Road, Room 202, Ann Arbor, MI-48103{prmanav, jwala, kusari}@umich.edu

Abstract

While much research has recently focused on generating physics-based adversarial samples, a critical yet often overlooked category arises from physical failures within onboard cameras—components essential to the perception systems of autonomous vehicles. Camera failures, whether due to external stresses causing hardware breakdown or internal component faults, can directly jeopardize the safety and reliability of autonomous driving systems. Firstly, we motivate the study using two separate real-world experiments to demonstrate that indeed glass failures would cause the detection-based neural network models to fail. Secondly, we develop a simulation-based study using the physical process of glass breakage to create perturbed scenarios, representing a realistic class of physics-based adversarial samples. Using a finite element model (FEM)-based approach, we generate surface cracks on the camera image by applying a stress field defined by particles within a triangular mesh. Lastly, we use physically-based rendering (PBR) techniques to provide realistic visualizations of these physically plausible fractures. To assess the safety implications, we apply the simulated broken glass effects as image filters to two autonomous driving datasets—KITTI and BDD100K—as well as the large-scale image detection dataset MS-COCO. We then evaluate detection failure rates for critical object classes using CNN-based object detection models (YOLOv8 and Faster R-CNN) and a transformer-based architecture with Pyramid Vision Transformers. To further investigate the distributional impact of these visual distortions, we compute the Kullback-Leibler (K-L) divergence between three distinct data distributions, applying various broken glass filters to a custo m dataset (captured through a cracked windshield), as well as the KITTI and Kaggle cats and dogs datasets. The K-L divergence analysis suggests that these broken glass filters do not introduce significant distributional shifts. Our goal is to provide a robust, physics-based methodology for generating adversarial samples that reflect real-world camera failures, with the overarching aim of improving the resilience and safety of autonomous driving systems against such physical threats.

Code —
https://github.com/manavprabhakar/camera-failure

Introduction

Cameras are ubiquitous as remote sensors, collecting data from an unstructured and dynamic external environment, often in harsh conditions. A failure or fault in a sensor is a divergence from the functional state in at least one given parameter of the system (van Schrick 1997). These faults can occur due to internal (such as wear and tear) or external (temperature, humidity, etc.) causes. For RGB cameras, internal causes include dead pixels while external causes include fractured enclosures or outer lenses, and condensation. These abrupt failures are hard to detect and negatively impact object detection algorithms—reducing accuracy and often leading to hallucination as shown in Fig. 1. The failures occurring in an automated vehicle (AV), for example, can lead to critical safety issues resulting in crashes and, in some cases, fatalities.

Currently, to the best of the authors' knowledge, there are no rigorous methods for generating camera-based sensor failures (Ceccarelli and Secci 2022).

In this work, we focus on sensor failure occurring due to fractures in any glass covering a camera (or camera enclosure), although the process detailed in this paper can be applied to any of the camera failures listed in (Ceccarelli and Secci 2022). These glass fracture effects in a camera can be caused by an external object hitting the camera or as a result of heat and/or pressure developing suddenly within the enclosure. In the parlance of neural networks, an image captured under such conditions is considered an adversarial sample. Previous research (Akhtar and Mian 2018; Carlini and Wagner 2017; Szegedy et al. 2013) shows that even small amounts of corruption, sometimes difficult for human eyes t o detect, are enough to completely fool the neural networks, where a subtle change in inputs can lead to a drastic change in outputs. We would like to note that (Li, Schmidt, and Kolter 2019) provided a physical camera-based adversarial attack paradigm, which serves as the closest related work in this domain. They presented a modification of the image using an overlay of a translucent, carefully crafted sticker that led to misclassification.

To understand the effect of these fractures on the resulting camera images, we conducted two distinct experiments: one

[*]Corresponding author

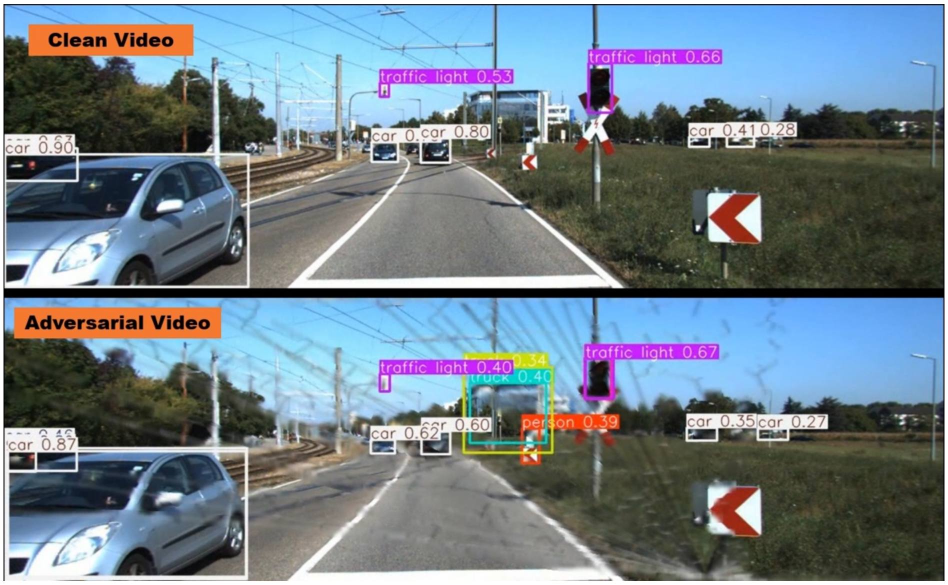


Figure 1: A qualitative comparison of clean vs adversarial video generated using our simulation and rendering method on KITTI. This frame shows false positives and reduced confidence levels for true positives. Refer to the supplementary material for the full video.

in an indoor static environment and the other in a dynamic outdoor environment. The first experiment involved fracturing tempered glass and placing it in front of the camera (see Fig. 2(a)) with a static vehicle in the scene to understand how different fracture patterns affect the quality and appearance of the scene. We captured images at different focal lengths to assess the variability of such corruptions. This helped us answer certain qualitative questions about the visual appearance of these fractures concerning their spread and intensity, motivating our approach in Section Focal Plane and Physical attack simulation. The experimental setup and the detailed experimental results are in Sec. Static Experiment of Supplementary. The second experiment (Fig. 2(b)) consisted of recording an outdoor video with dynamic vehicles under daylight conditions by placing a MobileEye camera next to a windshield crack presented in Fig. 2 (shown in the upper left) and performing inference using YOLOv8 (Jocher, Chaurasia, and Qiu 2023) to gain a primitive understanding of the i mpact of such scenarios on object detection networks. We observed that the model can easily detect the vehicle in a clean image while it suffers from detection failure (lower right) or generates false positives (lower left). Interestingly, the presence of a crack can also unexpectedly increase the confidence in the prediction of the car presenting a clearly defined edge (0.92 in the lower left vs. 0.75 in the upper left). The detailed inference results with vehicle and person class are given in Sec. Dynamic Experiment of Supplementary.

We then searched for real broken glass images online (Sec. Real glass fracture images of Supplementary) but were unable to build a dataset large enough to enable a data-driven approach for adversarial defense under these conditions. Additionally, we experimented with CGI tools like Maya and Blender to create such effects, but they lack the flexibility, control, scale, and physics necessary to simulate these conditions. The closest simulation option in existing literature is ArcSim (Pfaff et al. 2014). However, their high-resolution simulation outputs are extremely slow (≈ 20 hours), making it difficult t o scale. As a result, we directed our efforts towards creating a scalable simulation-based pipeline for generating fractures that can be used to advance the perception stack.

For a glass fracture, the principal point, force, and angle of incidence may be random, but the spread and the res ulting pattern follow an inherently physical process (being either linear or radial). We thus build a fracture simulation based on particles in a randomly generated triangular mesh and perform stress propagation through the mesh. Our simulation allows us to produce the fractures within a triangular mesh at every discrete time state δt. We use OpenCV to convert the given mesh into a corresponding broken glass pattern image. We then utilize physically-based rendering (PBR) (Pharr, Jakob, and Humphreys 2023) to realistically render the surface fractures using the bidirectional reflectance distribution function (BRDF) by calculating the amount of light reflected from a given point on a surface as a result of source(s) of light being incident on it.

Combining our rendering approach with three popular open source datasets -KITTI (Geiger et al. 2013), BDD100k (Yu et al. 2020) and MS-COCO (Lin et al. 2014), we are able to generate adversarial images efficiently. A common process for testing the generated adversarial images is to find the number of false positives/negatives across the image space. However, in our case, due to the adversarial effect being local, we cannot rely simply on an image based measure. We therefore, use the adversarial images (similar to the l ower left figure of Fig. 2) and extract the objects.

(a) (b)

Figure 2: (a) Indoor static experiment. Left: Camera with 2 different fractured tempered glass patterns; right - images of the vehicle under the different fractures. (b) Outdoor dynamic experiment. Top left - a coin-sized windshield crack; top right - clean image with the vehicle detected using YOLOv8; bottom left - false positive through the crack; bottom right - detection failure through the glass. More examples from these experiments have been provided in the supplementary material.

which lie within the region where the fracture exists using the ground truth bounding boxes. We then utilize YOLOv8, Faster R-CNN (Ren et al. 2016) and Pyramid Vision Transformer (PVTv2) (Wang et al. 2022) to find the percentage of objects that fail when the adversarial filters are applied. We also provide ablation studies to understand the distributional differences between the three sets of images: Real broken glass images collected experimentally, real broken glass images collected online and the generated images. We compute the Kullbeck-Liebler (K-L) divergence for these image distributions to prove similarity of the generated images to the real broken glass images. We utilize cat images from Kaggle Cats and Dogs dataset as control to understand the difference between image distributions (PK).

The major contributions of the paper can be summarized as follows:

- We provide a novel way of abstracting glass fracture through a combination of stress propagation methods and minimum spanning trees, to generate physically sound broken glass patterns.
- We present a PBR approach to facilitate a realistic render of camera failures that can be used with any kind of existing computer vision datasets - both images and videos.
- Our simulation and rendering pipelines are scalable and computationally efficient ($\approx 1.6s$), allowing them to be used by both academia and industry for enhancing robust and out-of-distribution protection for a wider range of applications.

Background

Physics based adversarial samples

The problem of adversarial sample can be defined as follo ws: for a model M that classifies an input sample X correctly to its designated class, i.e., $M(X) = y_{true}$, adding an error ϵ to the input sample X results in an altered sample X such that $M(X) \neq y_{true}$. Thus, the injection of the error ϵ results in an adversarial sample that causes the model to fail.

Although the idea of adversarial manipulation of the model has been identified in the context of machine learning quite some time ago (Dalvi et al. 2004), in the last decade, the focus has squarely been on the adversarial attacks on neural networks (Szegedy et al. 2013; Goodfellow, Shlens, and Szegedy 2014). In these papers, the researchers showed that a small targeted injection of noise, almost imperceptible to the human eye, changed the labels completely (Szegedy et al. 2013) and conversely, images could be generated that looked completely unrecognizable to humans but which had perfect classifications from the DNNs (Nguyen, Yosinski, and Clune 2015).

While these adversarial samples probe the model for possible failures, they lack any physical realism in their generation and require access to the model. To address this, some recent research has focused on creating physically relevant adversarial samples. One of the first attempts in this area was made by (Kurakin, Goodfellow, and Bengio 2018), who targeted the accuracy of models in the physical world by feeding noisy images from a cell phone camera, which caused the model to incorrectly classify a significant portion of the samples. In a similar vein, (Eykholt et al. 2018) demonstrated that real traffic signs can be altered with simple physical stickers placed strategically to deceive state-of-the-art deep learning algorithms almost perfectly, even with changes in viewpoint. Other researchers have used adversarial images (Kong et al. 2020), translucent patches on cameras (Zolfi et al. 2021), or artificial LiDAR surfaces (Tu et al. 2020) to generate samples that mislead object detectors. While this prior research employs physics i n generating the samples, they do not stem from modeling a rigorous physical process, and we aim to fill this gap in this work.

Cracked/fractured glass theory

The subject of how glass breaks and how it propagates remains an open research question, one that has been contentious with multiple physical theories proposed (Rouxel and Brow 2012). While the microscopic procedure of glass cracking is under debate, on a macroscopic level, the cracking dynamics is well understood. (Liu et al. 2021) analyzed the process of cracking in glass lenses within the precision glass molding application using FEM with a three-dimensional model in physical simulation software. The physical parameters were input into the software, and the crack paths were analyzed using the simulation results. The authors conducted a temperature and stress simulation of a high-precision three-dimensional mesh model of the molded glass. (Iben and O'Brien 2009) provided a method to generate surface fractures in a variety of materials, including glass. As mentioned in the introduction, (Pfaff et al. 2014) provided t he simulation of glass breaking as a thin sheet, which forms the closest related work to our proposed method.

Methodology

Generating realistic glass failures requires creating large-scale physics-based simulations by solving fracture dynamics on a triangulated finite element mesh with glass properties.

Broken glass simulation

We represent glass using particles sampled from a uniform distribution spread across a plane constrained in the form of a 2D mesh using constrained Delaunay triangulation. This removes ill-shaped triangles and avoids uneven and unrealistic edges.

Each particle p_i has a position x_i and has nearest neighbors $\mathbf{k}_i$ within a radius r which have existing edges with p_i. Mathematically, the triangulation mesh $\mathcal{M}$ represents a finite set of 2-simplices such that if

$$\forall (K, K') \in \mathcal{M} \times \mathcal{M}, |K| \cap |K'| = |K \cap K'|. \qquad (1)$$

The crack patterns in glass occur due to stress from the external force (F) at the initial impact point p_I by assuming a specific deformation law (elasticity and plasticity) of the glass (G) (Kuna 2013). We then compute the strength parameters i n the form of effective stress σ_V at the impact point (V) as t he stress state of the impact point. The critical stress values for the strength of glass σ_C are found using tests on simple samples with elementary loading conditions (e.g., tension test). The fracture then occurs when the effective stress is larger than the critical stress divided by the safety factor (S):

$$\sigma_V(G, F) > \frac{\sigma_C}{S}. \qquad (2)$$

From the classical theory of strength of materials, we know that failure in most cases is controlled by the principal stresses σ_I and σ_{II} for 2D elements. The initial crack occurs either through a normal-planar crack, where the fracture faces are oriented perpendicular to the direction of the highest principal stress σ_I (Rankine 1857), or a shear-planar crack, where the fracture faces align with the intersection planes of the maximum shear stress $\tau_{max} = (\sigma_I - \sigma_{II})/2$ (Coulumb 1776). In the case of glass, we assume that the initial fracture occurs perpendicular to the direction of the maximum principal stress.

From the initial impact point p_I, the stress propagation through glass is unstable as the crack grows abruptly without the need to increase external loading. From p_I, stress propagates in the vertex neighborhood $\mathbf{k_i}$ as the stress along the direction $\overrightarrow{p_I p_j}$ where $p_j \in \mathbf{k_i}$ as

$$\sigma_{p_j} = \sigma_V * \frac{\overrightarrow{p_I p_j} \cdot \overrightarrow{n}}{|\overrightarrow{p_I p_j}||\overrightarrow{n}|}. \qquad (3)$$

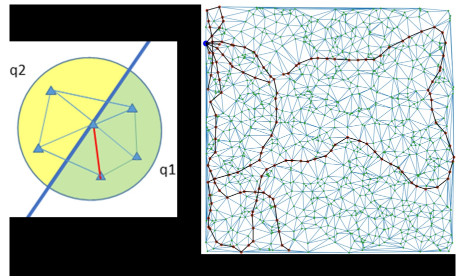


Figure 3: (a) For a splitting plane shown in blue, the summed positive stress q_1 and summed negative stress q_2 are compared, and propagation occurs on the side with greater summed stress and the chosen node that is closest to the splitting plane (shown in red). (b) Illustrates how we si mulate a fracture in a mesh originating from its impact point (marked in blue) to the nodes experiencing stress beyond their threshold strength (marked in red).

With the stress calculated for each edge, the summed positive stress (shown in Fig. 3) can then be expressed as:

$$q_1 = \int_{\partial\Omega} \sigma_{p_j} \mathbb{I}(\sigma_{p_j} > 0) dA \qquad (4)$$

for the continuous surface Ω where $\mathbb{I}$ the indicator function is defined. The summed positive stress for discrete simplices in the corresponding area A of radius R is given as

$$q_1 = \sum_{K \in A_R} \sigma_K \mathbb{I}(\sigma_K > 0). \qquad (5)$$

Similarly, the summed negative stress q_2 is calculated. Then, for a greater magnitude max($|q_1|, |q_2|$), we select the corresponding edge with the highest concentration of stress in the given s egment as the optimal splitting plane since that provides the maximum stress relief. Thus, the stress travels along the mesh edges, dissipating the stress at each node point.

The recursive application of the stress propagation is ru n until convergence of the stress in all states, i.e., $\sigma_p^{(t)} \simeq \sigma_p^{(t-1)} \forall p \in V$.

Propagating the stress in all directions across all nodes results in back-cracking, as explained in (O'Brien and Hodgins 1999). To avoid this, we propagate only along the edges where the stress levels are maximum but perform a stress update on all neighboring nodes. We then use a minimum spanning tree (MST) on a mesh created using these stressed nodes. We combine this MST with our initial stress propagation field along the edges to compute the final crack pattern. The MST is an effective abstraction because it connects the nodes that are closer to each other and within the high stress field while removing redundancies.

Our computational process of stress propagation is defined in Algorithm 1 in Supplementary.

Physically-based rendering

Once we have generated the fractures at the mesh level, our next goal is to create a visual render of these fractures. Like all PBR techniques, our method is based on the microfacet theory, which states that any surface can be described by tiny, perfectly reflective mirrors called microfacets (Pharr, Jakob, and Humphreys 2023).

In accordance with the microfacet theory and energy conservation, we use the reflectance equation,

$$L_o(x, \omega_o, \lambda, t) = L_e(x, \omega_o, \lambda, t) + L_r(x, \omega_o, \lambda, t) \quad (6)$$

where $L_o(x, \omega_o, \lambda, t)$ is the total spectral radiance of wavelength λ directed outward along direction ω_o at time t, from a particular position x. ω_o is the direction of the outgoing light. t is time. L_e is the emitted spectral radiance and L_r is the reflected spectral radiance.

Let I_1 be the bidirectional reflectance distribution function,

$$I_1 = f_r(x, \omega_i, \omega_o, \lambda, t)$$

and let I_2 be the spectral radiance coming inward towards x from direction ω_i at time t.

$$I_2 = L_i(x, \omega_i, \lambda, t)$$

Then, L_r can be defined as

$$L_r(x, \omega_o, \lambda, t) = \int_\Omega I_1 \cdot I_2 \cdot (\omega_i \cdot \mathbf{n})\, d\omega_i \quad (7)$$

where Ω is the unit hemisphere centered around the surface normal $\mathbf{n}$ over ω_i such that $\omega_i \cdot n > 0$.

Abstracting the reflectance equation, we aim to create a visual render of our broken glass mesh. We have $L_e = 0$ as glass does not emit light. Now, for calculating L_r, we consider any crack between the nodes as a microfacet. Then, we can define L_r for every crack as:

$$L_r = L_i\ (\omega_i \cdot \hat{\mathbf{n}}) \quad (8)$$

Given the unit vectors $(\hat{}\omega_\alpha)$ and $(\hat{}\omega_\theta)$ corresponding to the azimuth (α) and zenith (θ) angles respectively, we compute the mean energy incident on the crack as

$$\mathbb{E}(L_r) = \frac{|\hat{\omega}_\alpha \cdot \hat{n}_i| + |\hat{\omega}_\theta \cdot \hat{n}_i|}{2} \quad (9)$$

where $\hat{}n_i$ is the unit surface normal of the crack.

Let (I_r, I_g, I_b) be the mean intensity of the light source. Then the crack intensity, I_c is defined as

$$I_c = (I_r, I_g, I_b) \cdot \frac{\mathbb{E}(L_r)}{\sum L_r} \quad (10)$$

FocalPlane and Physical attacksimulation While we are able to simulate realistic fractures, the primary use case for our work is to generate simulated examples overlaid on existing datasets (KITTI, BDD100k, MS-COCO) and compare them with the real on-road dataset that we created.

Any captured image will exhibit sharp features of the objects in its Focal Plane. The glass enclosure covering the camera is extremely close and is thus not part of the focal plane. When the crack occurs, the light rays bounce u nevenly along the crack and create a blur (example provided in Fig. 4). We create a binary mask based on the crack pattern and then blur the fractures overlaid on the image. This produces a far-focus image. For a short-focus image, we blur the image and focus on the foreground, i.e., the crack.

Experimentation

Dataset

We benchmark two types of broken glass patterns—real and simulated—on three popular open-source datasets: KITTI (Geiger et al. 2013), BDD100k (Yu et al. 2020), and MS-COCO (Lin et al. 2014). The first two represent specific autonomous driving domains, while the last one is a general-purpose i mage dataset. The real broken glass pattern images are collected from the FreePik website [1] and serve as the baseline in our case. We collected a total of 65 images and expanded them to a set of 10,000 images through image augmentation using random shifts, image flips, and cropping techniques. We also generated 10,000 images using our physics si mulator. We then overlay these cracked glass patterns using our PBR pipeline onto every validation image in the datasets and collect the aggregate results. We use three model architectures: YOLOv8, Faster R-CNN, and PVTv2 model with pretrained weights to generate object detection results.

Implementation

Our simulation model is developed by randomly sampling 10^4 particles from a uniform spatial distribution in the given frame on a CPU. A KD-tree from the SciPy Python package (Virtanen et al. 2020) is constructed using default para meters to find the approximate nearest neighbors of each particle. A Delaunay triangulation is then performed on the particles to create a constrained triangular mesh. We use an impact force of 500 units with a random impact point and a random impact vector. Stress propagation occurs until a threshold of 300 units is reached. The PBR is executed on the CPU by implementing the methods described in the previous section using OpenCV and Python.

Results and Discussion

A major shift from most of the previous works on adversarial examples is that our generated adversarial patterns do not universally affect all pixels in an image. Therefore, the comparison needs to be made only for the image region where the pattern exists. For this purpose, we create a binary mask for each pattern and output the results of the objects that exist only within that pattern.

Table 1 shows the results of the average precision (AP) under the adversarial images generated using the two types of crack patterns (collected online and simulated) for different classes. For KITTI, the AP of other classes drops as expected, with the decrease in AP corresponding to the percentage of the image occupied by the truck class recording the highest drop. For BDD100K with PVTv2-B0, we see that the drop in AP is largest in the simulated images, but overall,

[1] https://www.freepik.com

Figure 4: (a) Shows the simulated image with the road and vehicles in the Focal Plane (PBR and Far-focus). (b) denotes the simulated crack pattern in the Focal Plane (PBR and short focus).

Table 1: Average precision (in percentage) of different classes in KITTI, BDD100k, and MS-COCO under different adversarial i mages. x provides the overlay relation between the dataset and glass-crack type. Clean x Dataset - refers to the particular images without any adversarial sample. RO x Dataset - refers to real images of cracked glass collected online overlaid on clean images. Si m x Dataset - refers to simulated crack patterns overlaid on clean images.

Dataset	IoU threshold	Category	Clean x Dataset	RO x Dataset	Sim x Dataset
KITTI (YOLOv8)	0.5	Pedestrian	25.64	69.72	**17.84**
		Truck	12.39	**3.59**	3.76
		Car	58.99	**50.7**	57.73
	0.75	Pedestrian	6.83	33.88	**6.02**
		Truck	11.29	**2.67**	2.79
		Car	31.25	**23.85**	30.15
BDD100k (PVTv2)	0.5	Pedestrian	66.47	54.33	**25.95**
		Truck	61.97	**52.83**	**52.02**
		Car	80.37	70.14	**56.78**
	0.75	Pedestrian	27.06	22.72	**10.60**
		Truck	47.03	**38.23**	42.52
		Car	46.23	45.97	**42.99**
MS-COCO (Faster R-CNN)	0.5	Person	0.035	0.024	**0.024**
		Vehicles	2.14	**1.45**	1.87
		Food	35.34	**28.07**	30.65
	0.75	Person	0.032	**0.022**	0.023
		Vehicles	1.56	**1.05**	1.07
		Food	24.59	**18.85**	22.00

the trend is maintained with the pedestrian class showing t he steepest drop. For MS-COCO, we aggregated the AP for the super-categories: person, vehicles, and food. This is because many objects in MS-COCO occupy a smaller area in the i mage frame, making it difficult to obtain meaningful results from all categories. A very intriguing result is that the pedestrian class has a multifold increase in AP under the real broken glass patterns. While this trend might seem counter-intuitive, it resonates with the results in Fig. 2, where the confidence of the car increases due to an edge. This, in fact, shows that the AP is highly dependent on the crack pattern, making it extremely important to create defense methodologies to mitigate these adversarial attacks.

Ablation studies

Our results indicate that the simulated images achieve a similar adversarial effect as the real images. Therefore, an important ablation study for us is to understand how closely the simulated crack patterns resemble the real cracked glass patterns and those collected online. We form 5 distributions.

- Real on-road dataset (depicted in Fig. 2)
- Crack patterns collected online (Fig. 5 top left)
- Simulated crack patterns (Fig. 5 bottom left)
- Simulated crack patterns overlayed on KITTI (Fig. 5 bottom right)
- Crack patterns collected online overlayed on KITTI (Fig. 5 top right)

We now compute the K-L divergence among all these distributions to determine how similar they are to each other (see Fig. 6). To provide a control, we compare KITTI to i mages of cats from the Kaggle dataset, yielding a K-L divergence of 2.434. On that scale, the PBR images of broken glass have a difference of 0.36 compared to the real broken glass patterns, while the broken glass filters overlaid on KITTI images exhibit a similar K-L divergence.

Fig.7 shows an analysis of the computation time for each of our modules and across different numbers of particles. We conduct this analysis over 100 runs, generating random impact points, impact angles, and mesh structures with a fixed number of particles. The variation in computation time for different runs can be attributed to the impact point and im-

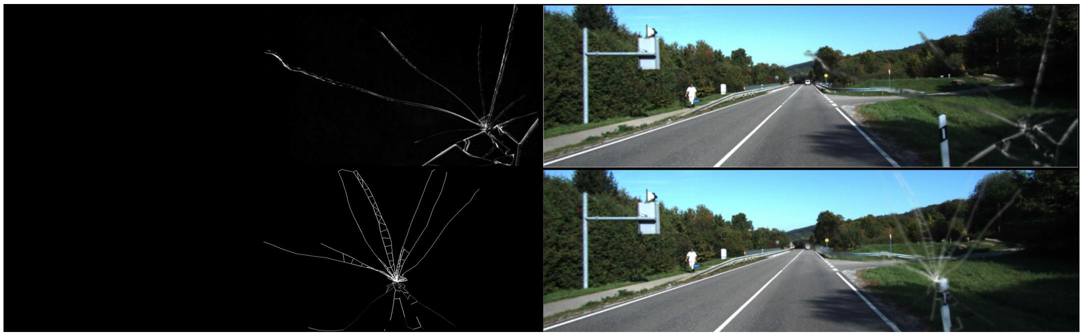


Figure 5: Top left - Crack pattern collected online from FreePik; top right - online crack pattern overlaid on KITTI; bottom left - simulated crack pattern with PBR; bottom right - simulated crack pattern overlaid on KITTI.

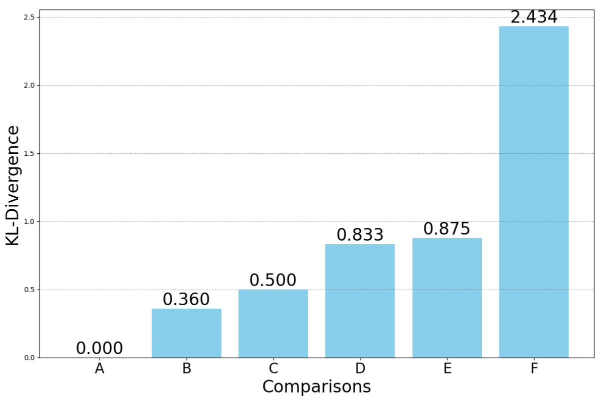


Figure 6: K-L divergence of different pairs of image distributions. Datasets: RC - Real on-road dataset (see Fig. 2), KITTI, and Cats. Filters: RO - Real (collected online) and Sim - Simulated. K-L divergence between (x - overlay relation): A - (Sim x KITTI) vs (Sim x KITTI); B - (Sim vs RO); C - (Clean RC vs KITTI); D - (Broken RC) vs (RO x KITTI); E - (Broken RC) vs (Sim x KITTI); F - KITTI vs Cats.

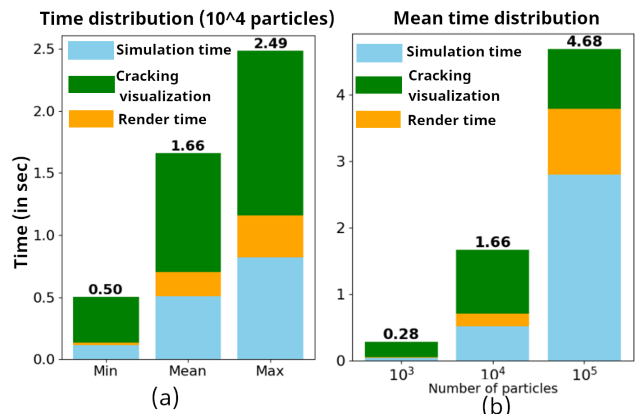


Figure 7: (a) Mean time taken by different modules of our pipeline across 100 runs. (b) The minimum, maximum, and mean time taken by different modules across 100 runs for a 10^4 particle mesh. For these plots, we showcase the time taken for simulation (simulation time), converting the mesh to glass (cracking visualization), and finally rendering (render time).

pact angle. The visualization of cracking and render time also varies due to different sized masks formed by varying fracture patterns. We also change the number of particles and observe how runtime increases exponentially with the increase in particles. All these runs were rendered on images from the KITTI dataset with dimensions of $(375 \times 1242 \times 3)$.

Conclusion and Future Scope

We have introduced a novel class of adversarial failures res ulting from the physical process of failures in the camera. In this paper, we provide an approach to generate a realistic broken glass pattern from a physical simulation and subsequently e mbed that into existing image datasets using physically based rendering. We show that the simulated adversarial images can lead to significant errors in object detection.

In this work, we address black-box adversarial attacks stemming from real-world, naturally occurring physical phenomena, not artificially crafted to exploit specific model vulnerabilities. We assume no knowledge of the model attributes, weights, or architecture, ensuring attacks are transferable across various models. Physical adversarial methods (Translucent Patch, RP2) can all be termed as occlusions of either the camera or the objects being captured. The adversariality comes from the effect of the model inference due to these occlusions. Our PBR pipeline blends the cracks with source images as translucent, blurry patter ns, impacting latent space encoding rather than causing direct occlusion, resulting in incorrect detections.

While this work introduces a physics-based method specifically for generating broken glass patterns, camera failures also encompass other effects such as sun glare, overexposure, underexposure, condensation, etc. Our future work will focus on creating an adversarial toolbox for the realistic generation of these effects using physics and subsequently placing them on existing image datasets and c ar simulation platforms to promote further research in the field of partial camera failures.

References

Akhtar, N.; and Mian, A. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6: 14410–14430.

Carlini, N.; and Wagner, D. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In*Proceedings of the 10th ACM workshop on artificial intelligence and security*, 3–14.

Ceccarelli, A.; and Secci, F. 2022. RGB camera failures and their effects in autonomous driving applications. *IEEETransactions on Dependable and SecureComputing*. Coulumb, C.-A. 1776. Essai sur une application des règles des maximis et minimis à quelques problèmes de statique relatifs à l'architecture. *Mem. Acad. Roy. Div. Sav.*, 7: 343–387. Dalvi, N.; Domingos, P.; Mausam; Sanghai, S.; and Verma, D. 2004. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 99–108. Eykholt, K.; Evtimov, I.; Fernandes, E.; Li, B.; Rahmati, A.; Xiao, C.; Prakash, A.; Kohno, T.; and Song, D. 2018. Robust physical-world attacks on deep learning visual classification. In *Proceedings ofthe IEEE conference on computer visionand patternrecognition*, 1625–1634. Geiger, A.; Lenz, P.; Stiller, C.; and Urtasun, R. 2013. Vision meets Robotics: The KITTI Dataset. *International Journalof RoboticsResearch(IJRR)*. Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprintarXiv:1412.6572*. Iben, H. N.; and O'Brien, J. F. 2009. Generating surface crack patterns. *GraphicalModels*, 71(6): 198–208. Jocher, G.; Chaurasia, A.; and Qiu, J. 2023. Ultralytics YOLO. Kong, Z.; Guo, J.; Li, A.; and Liu, C. 2020. Physgan: Generating physical-world-resilient adversarial examples for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14254–14263. Kuna, M. 2013. Finite elements in fracture mechanics. *Solidmechanics andits applications*, 201: 153–192. Kurakin, A.; Goodfellow, I. J.; and Bengio, S. 2018. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, 99–112. Chapman and Hall/CRC. Li, J.; Schmidt, F.; and Kolter, Z. 2019. Adversarial camera stickers: A physical camera-based attack on deep learning systems. In *International conferenceon machine learning*, 3896–3904. PMLR. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Doll´ar, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *Computer Vision– ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 740–755. Springer.

Liu, Y.; Xing, Y.; Li, C.; Yang, C.; and Xue, C. 2021. Analysis of lens fracture in precision glass molding with the finite element method. *Applied Optics*, 60(26): 8022–8030. Nguyen, A.; Yosinski, J.; and Clune, J. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 427–436. O'Brien, J. F.; and Hodgins, J. K. 1999. Graphical Modeling and Animation of Brittle Fracture. In *Proceedingsof ACMSIGGRAPH 1999*, 137–146. ACM Press/Addison-Wesley Publishing Co. Pfaff, T.; Narain, R.; De Joya, J. M.; and O'Brien, J. F. 2014. Adaptive tearing and cracking of thin sheets. *ACM Transactions on Graphics (TOG)*, 33(4): 1–9. Pharr, M.; Jakob, W.; and Humphreys, G. 2023. *Physically based rendering: From theory to implementation*. MIT Press. PK, A. ???? Kaggle cats and dogs mini dataset.
h
t
t
p
s://
w
w
w.
k
a
g
gle.co
m/datasets/aleemaparakatta/cats-and-dogs-mini-dataset. Accessed: 2024-09-30. Rankine, W. J. M. 1857. II. On the stability of loose earth. *Philosophical Transactions of the Royal Society of London*, (147): 9–27. Ren, S.; He, K.; Girshick, R.; and Sun, J. 2016. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6): 1137–1149. Rouxel, T.; and Brow, R. K. 2012. The Flow and Fracture of Advanced Glasses—an Overview. *International Journal ofApplied Glass Science*, 3(1): 1–2. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXivpreprintarXiv:1312.6199*. Tu, J.; Ren, M.; Manivasagam, S.; Liang, M.; Yang, B.; Du, R.; Cheng, F.; and Urtasun, R. 2020. Physically realizable adversarial examples for lidar object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 13716–13725. van Schrick, D. 1997. Remarks on terminology in the field of supervision, fault detection and diagnosis. *IFAC Proceedings Volumes*, 30(18): 959–964. Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Millman, K. J.; Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, ˙I.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272. Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; and Shao, L. 2022. Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3): 415–424.

Yu, F.; Chen, H.; Wang, X.; Xian, W.; Chen, Y.; Liu, F.; Madhavan, V.; and Darrell, T. 2020. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2636–2645. Zolfi, A.; Kravchik, M.; Elovici, Y.; and Shabtai, A. 2021. The translucent patch: A physical and universal attack on object detectors. In *Proceedingsof the IEEE/CVF conferenceon computervision andpattern recognition*, 15232–15241.

Algorithm of stress propagation

Algorithm 1 describes the procedure for simulating the propagation of stress through a material following an impact event. The algorithm takes as inputs the location of the impact (pt), the magnitude of the impact force (F), the impact direction vector (v), and the parent edge (PE) associated with the impact site. It also uses a nearest neighbor radius R to determine the set of candidate locations for stress propagation.

Algorithm 1: Stress Propagation

```
pt ← Impact Point
F ← Impact Force
PE ← Parent Edge
v ← Impact Vector
R ← Nearest neighbor radius

procedure PROPAGATESTRESS(Pt, F, V, PE)
    frontiers ← KDTree − queryRadius(R)
    NN ← (frontiers−pt) / ||frontiers−pt||
    cos(θ) ← NN · v
    stress ← calculateStress(cos(θ) F)
    frontiers ← frontiers[argmax(stress)]
    v ← v[argmax[stress]]
    PE ← PE[argmax[stress]]
    PROPAGATESTRESS(Pt, F, V, PE)
end procedure
```

First, it uses a KD-tree data structure to efficiently query all points (frontiers) within a given radius R of the impact point. For each frontier, it computes a unit direction vector from the impact point to the frontier (NN). It then projects the impact vector v onto this direction to obtain the cosine similarity $cos(\theta)$, capturing the angular relationship between the impact direction and the candidate propagation direction. For each candidate, the resulting value is used, together with the impact force, to calculate the corresponding stress at that point. The algorithm then selects the candidate with the maximum stress value. The impact vector v and parent edge PE are updated to correspond to this new direction. The process is recursively repeated, allowing the simulated stress wave to propagate iteratively through the material along the path of greatest stress transfer.

This approach aims to mimic how stress from an impact point is most likely to radiate through a material—preferentially following paths defined by both geometric proximity and mechanical alignment with the original impact.

The final output of the simulation is the realization of the mesh as an image that corresponds to the broken lens pattern (final image of Fig. 8).

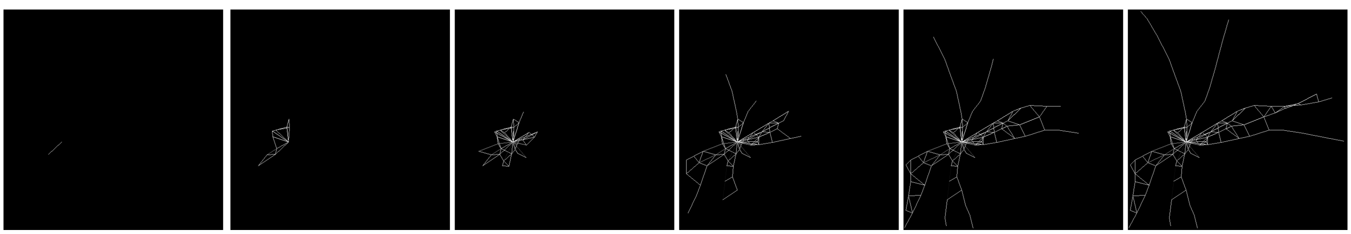


Figure 8: An animation of the fracturing of a lens simulated by setting the stress field and applying Sim. R.

Static Experiment

To understand the effect of these fractures on the resultant images, we first conduct an indoor static experiment as referenced in the Section Introduction. For this experiment, we use various tempered glass sheets, which we randomly break using a small hammer with either a single or multiple break points. Then, we place a 36 MP JVC GC-PX10 hybrid camera mounted on a tripod with a clamp in front of the tempered glass.

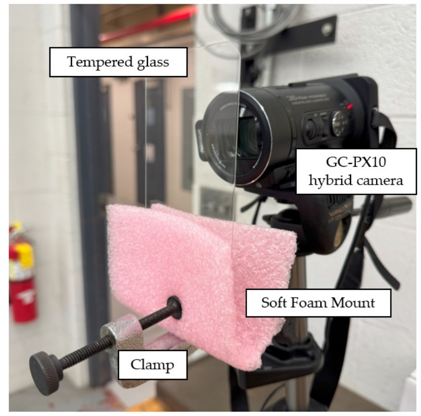
Fig. 9(a) shows the detailed setup with the camera mount and the tempered glass held in place with a clamp. Fig. 9(b) displays the image captured by the camera, while Fig. 9(c) shows the single vehicle positioned as the primary object being captured by the camera through the tempered glass. The scene is illuminated using overhead fluorescent lights.

Fig. 10 illustrates some of the fractures and scratched patterns on the tempered glass. These patterns were intentionally randomized, employing multiple focal points and varying levels of force to mimic the unpredictable and diverse nature of real-world glass damage. By applying different force strengths, we were able to produce a spectrum of fractures and scratches, ranging from fine surface abrasions to more pronounced fractures. This approach was chosen to closely replicate the types of damage that glass surfaces may encounter in actual conditions—such as those caused by impacts, debris, or environmental stressors—thereby ensuring the relevance and realism of our experimental setup. These representative damage patterns allow us to more effectively analyze the influence of glass imperfections on sensor performance and object detection algorithms.

Two different fracture patterns and their resultant images are shown in Fig. 11 and Fig. 12. We would like to note that we considerably varied the focal lengths of the camera to understand how the images appear under near- and far-focus. The outputs demonstrate that even minor scratched patterns are visible in the image output, while much stronger multi-fracture patterns can blur almost the entire image. This experiment provides the intuition on which our simulation and visualization framework is built.

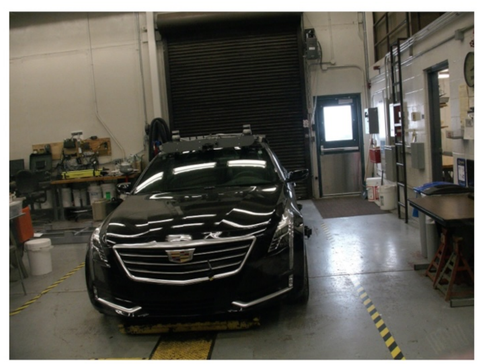
Increased AP for pedestrians in KITTI

We would like to point out that the increased AP for the pedestrian class was something that even surprised us at first. However, a careful qualitative deep-dive analysis helped us understand that this was occurring as a result of the glass cracks making it easier for the model to classify pedestrians due to enhanced edges around them. This wasn't an edge artifact but rather the glass crack acting as an additional edge boundary that clearly separates the pedestrian and the background. A similar result was also observed in [1] where the overall AP was increased in adversarial images.



(a)

(b)



(c)

Figure 9: Experimental setup for collecting images affected by scratched or broken outer layers of a camera. (a) displays the entire setup for capturing adversarial images. (b) illustrates the position of the camera relative to the scene being captured. (c) depicts the scene being captured by the camera.

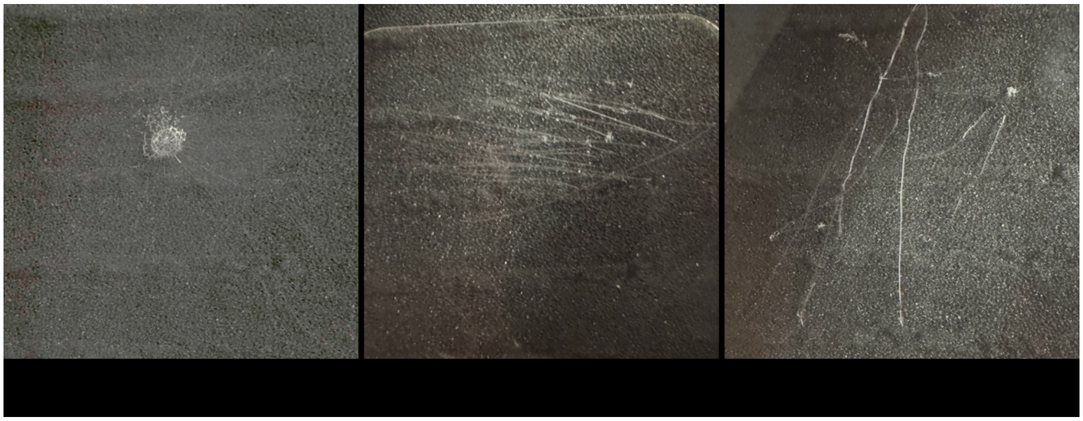


Figure 10: Some fractures/scratched patterns on the glass we used for collecting the images. (a) A sharp force applied perpendicular to the glass surface, producing fractures occurring radially. (b) and (c) replicate a glass with scratches.

Figure 11: (a) Shows the scratched pattern placed in front of the camera, (b) shows the camera POV. (c) shows the image captured by the camera (short-focus). (d) shows the image captured by the camera (far-focus)

Figure 12: (a) Shows the broken glass pattern in front of the camera, (b) shows the camera POV. (c) shows the image captured by t he camera (short-focus). (d) shows the image captured by the camera (far-focus)

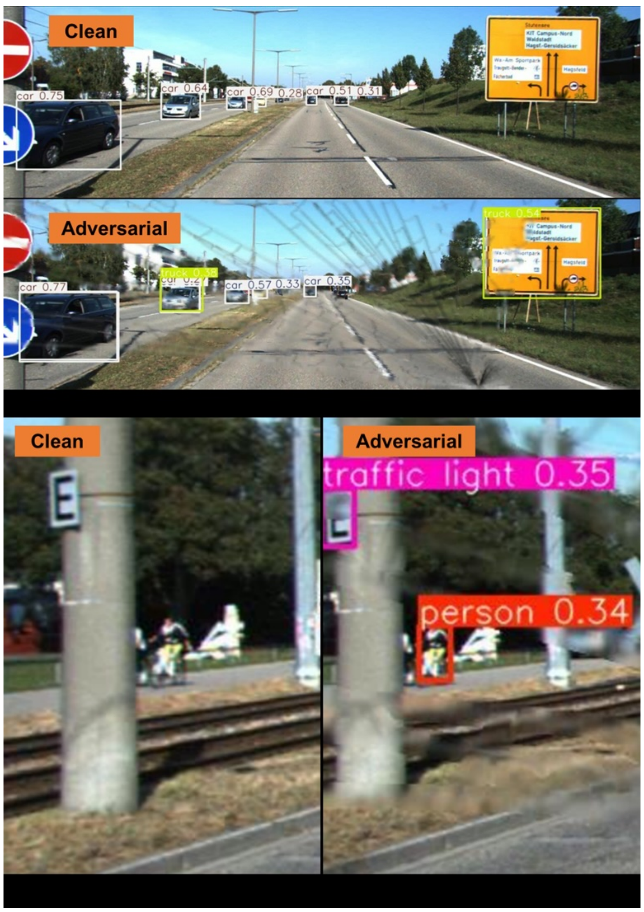


Figure 13: (a) Top - detections on a clean image; bottom - detections on an adversarial image. (b) YoLo fails to detect t he person (c) Glass cracks allows the model to detect the person.

动态实验

In this section, we describe the dynamic experiment mentioned in the 引言部分. We perform this experiment to understand the temporal perturbation introduced by a crack. We use a 挡风玻璃裂缝 of a 车辆 and place a small camera on the dashboard behind the crack. Then we photograph two dynamic objects - a 车辆 and a pedestrian as they move across the scene. 图14 provides some specific image frames with inference from YOLOv8 for the vehicle class. We show that with the crack, the 车辆 remains undetected in most frames. Additionally, almost every frame contains a false positive. Correspondingly, we present 图15 as the frames with a person walking in the scene. We show that it intermittently provides detection and occasionally with a wrong class (冲浪板).

Figure 14: Specific frames of the images taken with the windshield crack using YOLOv8 inference for the vehicle class. A - false positive with no object in the scene; B - no inference on the vehicle; C - no inference on the vehicle; D - first detection on the ve hicle; E - two different detections on the same vehicle; F - wrong bounding box area.

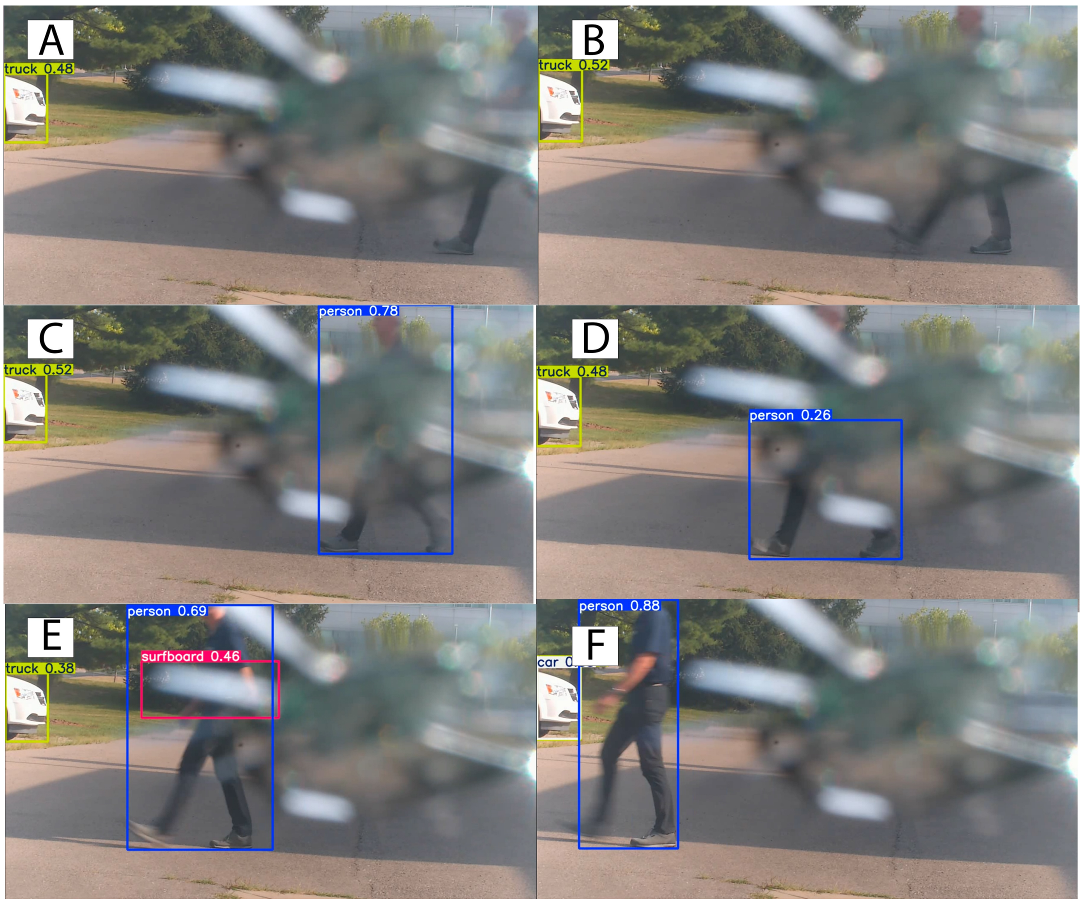


Figure 15: Specific frames of the images taken with the windshield crack using YOLOv8 inference for the person class. A - first entry of a person in the scene with no detection; B - no inference of a person; C - first detection of a person; D - partial detection of a person; E - detection of a person with another class; F - full detection of a person.

Real glass fracture images

We present an example of the glass fracture images collected from the FreePik website overlaid on the KITTI dataset along with YOLOv8 inference (Fig. 16). We show that the fracture removes some detections and decreases the detection confidence of others.

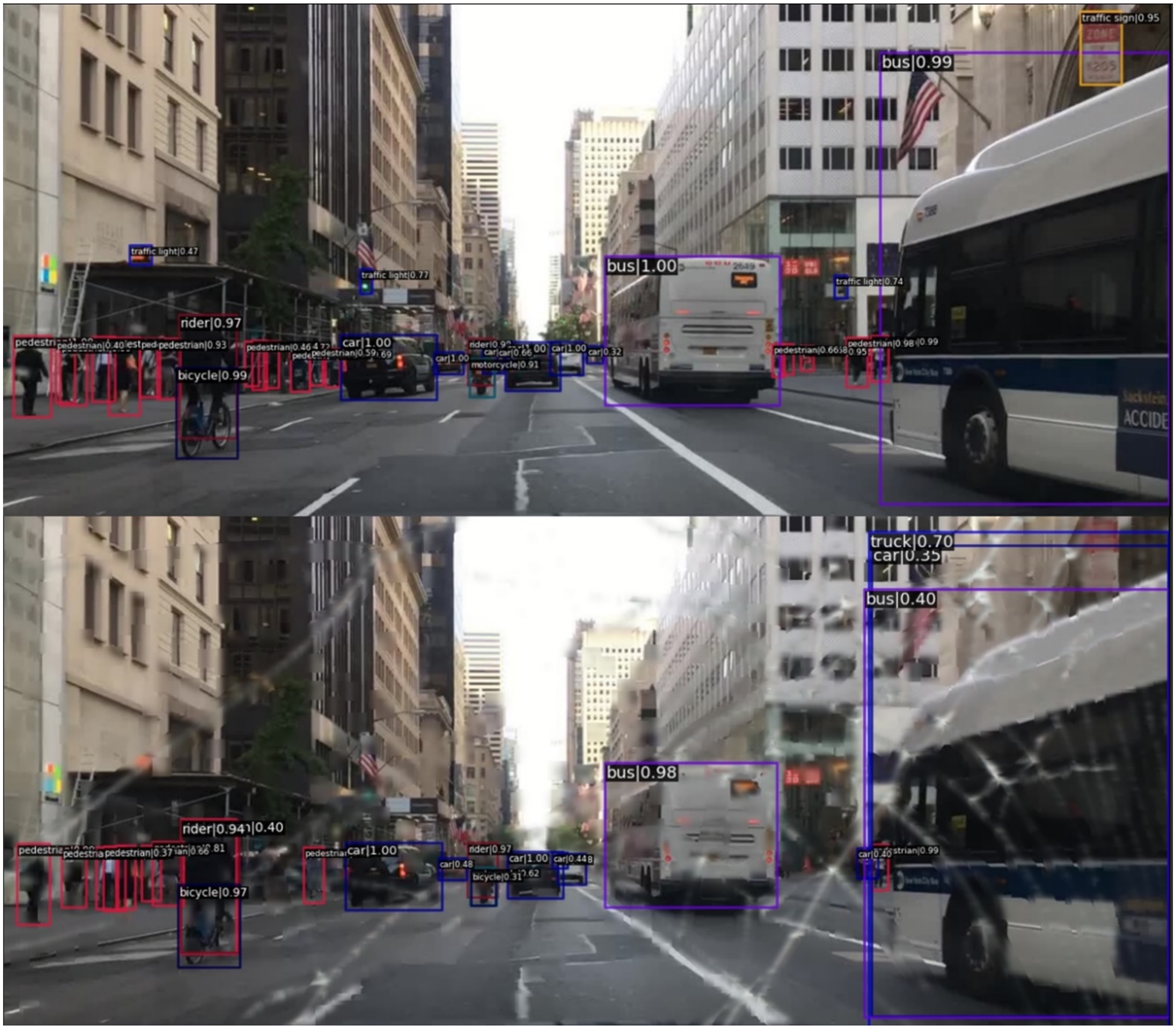


Figure 16: Top inference of PTv2 on a clean image from BDD100k. Bottom - Inference for a real broken glass image overlaid on BDD100k for comparison. We observe two additional false positives on the right side (truck, car) and several false negatives for the pedestrian class on the left of the adversarial image.