

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

Manav Prabhakar, Jwalandhar Girnar, Arpan Kusari*

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 originates from physical failures within on-board 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 showcase 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 the 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 custom 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 lens, 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 authors' knowledge, there are no rigorous methods for generating camera based sensor failures (Ceccarelli and Secci 2022).

In this work, we focus on the sensor failure occurring due to fractures in any glass covering a camera (or camera enclosure), although the process detailed in this paper can be used for any of the camera failures listed in (Ceccarelli and Secci 2022). These glass fracture effects in a camera can be caused due to 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 in such conditions is considered as an adversarial sample. Previous research (Akhtar and Mian 2018; Carlini and Wagner 2017; Szegedy et al. 2013) shows that even small amounts of corruptions, sometimes difficult to be seen by human eyes, are enough to completely fool the neural networks where a subtle change of 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 which led to misclassification.

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

*Corresponding author

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Gebroken Glas, Falende Camera's: Simuleren van Fysisch-gebaseerde Adversariële Steekproeven voor Autonome Rijsystemen

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

Samenvatting

Hoewel recent veel onderzoek zich heeft gericht op het genereren van fysiek-gebaseerde adversariële steekproeven, is er een kritieke, maar vaak over het hoofd gezien categorie die voortkomt uit fysieke storingen binnen boordcamera's—onderdelen die essentieel zijn voor de perceptiesystemen van autonome voertuigen. Camerastoringen, of ze nu worden veroorzaakt door externe spanningen die hardwarestoringen veroorzaken van interne componentenfouten, kunnen direct de veiligheid en betrouwbaarheid van autonome rijsystemen in gevaar brengen. Ten eerste motiveren we de studie met behulp van twee afzonderlijke experimenten in de echte wereld om aan te tonen dat glasstoringen inderdaad zouden leiden tot het falen van op detectie gebaseerde neurale netwerkmodellen. Ten tweede ontwikkelen we een simulatie-gebaseerde studie met behulp van het fysieke proces van glasbreuk om verstoerde scenario's te creëren, die een realistische klasse van fysiek-gebaseerde adversariële steekproeven vertegenwoordigen. Met behulp van een eindige-elementenmodel (Eindige Elementen Methode)-gebaseerde benadering genereren we oppervlaktebarsten op de camera-afbeelding door een spanningsveld toe te passen dat wordt gedefinieerd door deeltjes binnen een driehoekig raster. Ten slotte gebruiken we fysiek-gebaseerde rendering (PBR) technieken om realistische visualisaties te bieden van deze fysiek plausibele breuken. Om de veiligheidsimplicaties te beoordelen, passen we de gesimuleerde gebroken glaseffecten toe als afbeeldingsfilters op twee datasets voor autonoom rijden - KITTI en BDD100K - evenals de grootschalige afbeeldingsdetectiedataset MS-COCO. We evalueren vervolgens de detectiefalingspercentages voor kritieke objectklassen met behulp van op CNN gebaseerde objectdetectiemodellen (YOLOv8 en Faster R-CNN) en een transformer-gebaseerde architectuur met Pyramid Vision Transformers. Om de distributie-impact van deze visuele verformingen verder te onderzoeken, berekenen we de Kullback-Leibler (K-L) divergentie tussen drie verschillende data-distributies, waarbij we verschillende gebroken glasfilters toepassen op een aangepaste dataset (vastgelegd door een gebaarten voorruit), evenals de KITTI en Kaggle Cats and Dogs datasets. De K-L divergentieanalyse suggerert dat deze gebroken glasfilters geen significante distributieverschuivingen introduceren. Ons doel is om een robuuste, fysiek-gebaseerde methodologie te bieden voor het genereren van adversariële steekproeven die reële camerastoringen weerspiegelen, met als overkoepelend doel de veerkracht en veiligheid van autonome rijsystemen tegen dergelijke fysieke bedreigingen te verbeteren.

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

Inleiding

Camera's zijn alomtegenwoordig als afstandssensoren, die gegevens verzamelen uit een ongestructureerde en dynamische externe omgeving, vaak onder zware omstandigheden. Een storing of fout in een sensor is een afwijking van de functionele staat in ten minste één gegeven parameter van het systeem (van Schrick 1997). Deze fouten kunnen optreden door interne (zoals slijtage) of externe (temperatuur, vochtigheid, enz.) oorzaken. Voor RGB-camera's omvatten interne oorzaken dode pixels, terwijl externe oorzaken gebroken behuizingen of buitenlenzen en condensatie omvatten. Deze abrupte storingen zijn moeilijk te detecteren en hebben een negatieve invloed op objectdetectie-algoritmen, waardoor de nauwkeurigheid afneemt en vaak tot hallucinaties leidt, zoals getoond in Fig. 1. De storingen die bijvoorbeeld in een geautomatiseerd voertuig (AV) optreden, kunnen leiden tot kritieke veiligheidsproblemen die resulteren in ongelukken en in sommige gevallen, dodelijke slachtoffers.

Op dit moment, voor zover de auteurs weten, zijn er geen rigoureuze methoden voor het genereren van op camera gebaseerde sensorstoringen (Ceccarelli en Secci 2022).

In dit werk richten we ons op het falen van sensoren dat optreedt door breuken in elk glas dat een camera (of camera-omhulsel) bedekt, hoewel het proces dat in dit artikel wordt beschreven, kan worden gebruikt voor elk van de cameraftuinen die worden vermeld in (Ceccarelli en Secci 2022). Deze glasbreukeffecten in een camera kunnen worden veroorzaakt door een extern object dat de camera raakt of als gevolg van plotselinge ontwikkeling van hitte en/of druk binnens het omhulsel. In de terminologie van neurale netwerken wordt een afbeelding die onder dergelijke omstandigheden is vastgelegd, beschouwd als een adversarial sample. Eerder onderzoek (Akhtar en Mian 2018; Carlini en Wagner 2017; Szegedy et al. 2013) toont aan dat zelfs kleine hoeveelheden verstoringen, soms moeilijk te zien met het menselijk oog, voldoende zijn om de neurale netwerken volledig te misleiden, waarbij een subtiele verandering van invoer kan leiden tot een drastische verandering in uitvoer. We willen opmerken dat (Li, Schmidt en Kolter 2019) een fysiek cameragebaseerd adversarial attack-paradigma hebben gepresenteerd, dat dient als het meest verwante werk in dit domein. Zij presenteerden een aanpassing van de afbeelding met behulp van een overlay van een doorschijnende, zorgvuldig vervaardigde sticker die leidde tot verkeerde classificatie.

Om het effect van deze breuken op de resulterende camerabeelden te begrijpen, hebben we twee verschillende experimenten uitgevoerd: één

*Correspondentiauteur Copyright © 2026, Vereniging voor de Bevordering van Kunstmatige Intelligentie(www.aaai.org). Alle rechten voorbehouden.

arXiv:2405.15033v3 [cs.CV] 14 Nov 2025

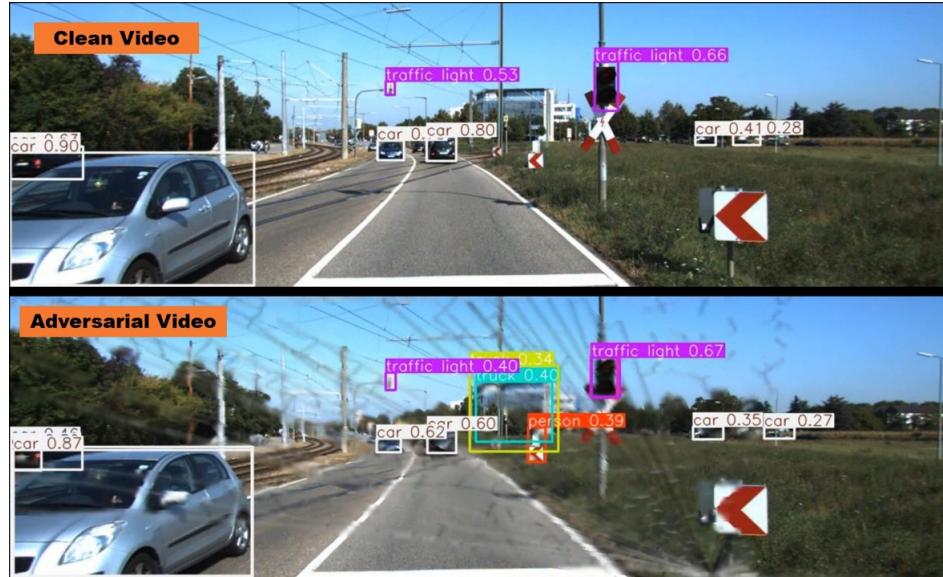
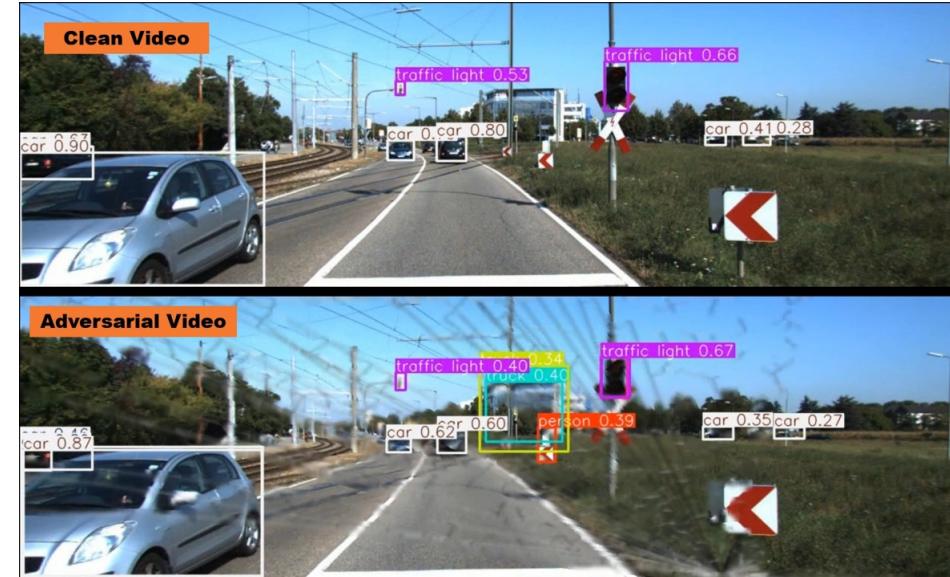


Figure 1: A qualitative comparison of clean vs adversarial video generated using our simulation and rendering method on KITTI. This frame shows false positives, and reduced confidence levels for true positives. Refer to the supplementary material for full video.

in an indoor static environment and the other in a dynamic outdoor environment. The first one 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 judge the variability of such corruptions. This helped us answer certain qualitative questions about the visual appearance of these fractures with respect to 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 impact 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 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 is given in Sec. Dynamic Experiment of Supplementary.

We then looked for real broken glass images online (Sec. Real glass fracture images of Supplementary) but failed to build a dataset large enough to enable a data-driven approach for adversarial defense for these conditions. Additionally, we experimented with CGI tools like Maya and Blender for



Figuur 1: Een kwalitatieve vergelijking van schone versus adversariële video gegenereerd met behulp van onze simulatie- en renderingsmethode op KITTI. Dit frame toont valse positieven en verminderde vertrouwensniveaus voor ware positieven. Raadpleeg het aanvullend materiaal voor de volledige video.

creating such effects but they lack the flexibility, control, scale and physics 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 to scale. As a result, we directed our efforts towards creating a scalable simulation-based pipeline for generating fractures that can be used to advance perception stack.

For a glass fracture, the principal point, force and angle of incidence may be random, but the spread and the resulting pattern follows an inherently physical process (being either linear or radial). We thus build a fracture simulation based on particles in a triangular mesh generated randomly 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 to 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 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 lower left figure of Fig. 2) and extract the objects

in een statische binnenomgeving en de andere in een dynamische buitenomgeving. Het eerste experiment betrof het breken van gehard glas en het voor de camera plaatsen (zie Fig. 2(a)) met een statisch voertuig in de scène om te begrijpen hoe verschillende breukpatronen de kwaliteit en het uiterlijk van de scène beïnvloeden. We maakten opnamen bij verschillende brandpuntsafstanden om de variabiliteit van dergelijke verstoringen te beoordelen. Dit hielp ons bepaalde kwalitatieve vragen te beantwoorden over het visuele uiterlijk van deze breuken met betrekking tot hun verspreiding en intensiteit, wat onze aanpak in de sectie Brandpuntsvlak en Simulatie van fysieke aanval motiveerde. De experimentele opstelling en de gedetailleerde experimentele resultaten zijn te vinden in Sectie Statisch Experiment van Aanvullend. Het tweede experiment (Fig. 2(b)) bestond uit het opnemen van een buitenvideo met dynamische voertuigen onder daglichtomstandigheden door een MobileEye-camera naast een voorruitbreuk te plaatsen zoals gepresenteerd in Fig. 2 (getoond in de linkerbovenhoek) en het uitvoeren van inferentie met behulp van YOLOv8 (Jocher, Chaurasia, en Qiu 2023) om een primitief begrip te krijgen van de impact van dergelijke scenario's op objectdetectienetwerken. We observeerden dat het model het voertuig gemakkelijk kan detecteren in een schoon beeld, terwijl het last heeft van detectiefouten (rechtsonder) of valse positieven genereert (linksonder). Interessant genoeg kan de aanwezigheid van een breuk ook onverwacht het vertrouwen in de voorspelling van de auto verhogen, waarbij een duidelijk gedefinieerde rand wordt gepresenteerd (0,92 linksonder vs. 0,75 linksboven). De gedetailleerde inferentieresultaten met voertuig- en persoonsklasse zijn te vinden in Sectie Dynamisch Experiment van Aanvullend.

We zochten vervolgens online naar echte afbeeldingen van gebroken glas (Sectie Echte glasbreukafbeeldingen van Aanvullend), maar slaagden er niet in een dataset groot genoeg te bouwen om een data-gedreven benadering voor adversariële verdediging onder deze omstandigheden mogelijk te maken. Daarnaast experimenteerden we met CGI-tools zoals Maya en Blender voor

het creëren van dergelijke effecten, maar ze missen de flexibiliteit, controle, schaal en fysica om deze omstandigheden te simuleren. De dichtstbijzijnde simulatieoptie in de bestaande literatuur is ArcSim (Pfaff et al. 2014). Hun simulatie-uitvoer met hoge resolutie is echter extreem traag (≈ 20 uur), wat het moeilijk maakt om op te schalen. Als gevolg daarvan richtten we onze inspanning op het creëren van een schaallbare simulatie-gebaseerde pijplijn voor het genereren van breuken die kunnen worden gebruikt om de perceptiestapel te verbeteren.

Voor een glasbreuk kunnen het hoofdpunt, de kracht en de invalshoek willekeurig zijn, maar de verspreiding en het resulterende patroon volgen een inherent fysiek proces (zijnde lineair of radiaal). We bouwen daarom een breuksimulatie gebaseerd op deeltjes in een willekeurig gegenereerd driehoekig netwerk en voeren spanningsvoortplanting door het netwerk uit. Onze simulatie stelt ons in staat om de breuken binnen een driehoekig netwerk te produceren in elke discrete tijdstoestand δt . We gebruiken OpenCV om het gegeven netwerk om te zetten in een overeenkomstig gebroken glaspatroonbeeld. Vervolgens maken we gebruik van fysiek-gebaseerde rendering (PBR) (Pharr, Jakob, en Humphreys 2023) om de oppervlaktebreuken realistisch te renderen met behulp van de bidirectionele reflectantieverdelingsfunctie (BRDF) door de hoeveelheid licht te berekenen die wordt gereflecteerd vanaf een bepaald punt op een oppervlak als gevolg van lichtbron(nen) die erop invallen.

Door onze renderingsaanpak te combineren met drie populaire open source datasets - KITTI (Geiger et al. 2013), BDD100K (Yu et al. 2020) en MS-COCO (Lin et al. 2014), zijn we in staat om efficiënt adversariële afbeeldingen te genereren. Een gebruikelijke methode om de gegenereerde adversariële afbeeldingen te testen, is het aantal false positives/negatives in de beeldruimte te vinden. Echter, in ons geval, vanwege het lokale adversariële effect, kunnen we niet eenvoudigweg op een beeldgebaseerde maatstaf vertrouwen. Daarom gebruiken we de adversariële afbeeldingen (vergelijkbaar met de figuur linksonder in Fig. 2) en extraheren we de objecten

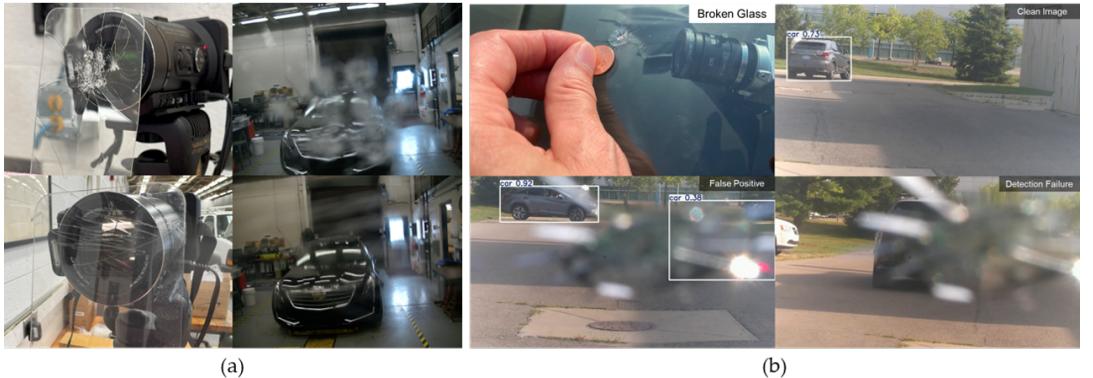


Figure 2: (a) Indoor static experiment. Left: Camera with 2 different fractured tempered glass patterns; right - images of the vehicle under the different fractures. (b) Outdoor dynamic experiment. Top left - a coin sized windshield crack; top right - clean image with the vehicle detected using YOLOv8; 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 set 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 it to be used by both academia and industry for enhancing robust and out of distribution protection for a wide range of applications.

Background

Physics based adversarial samples

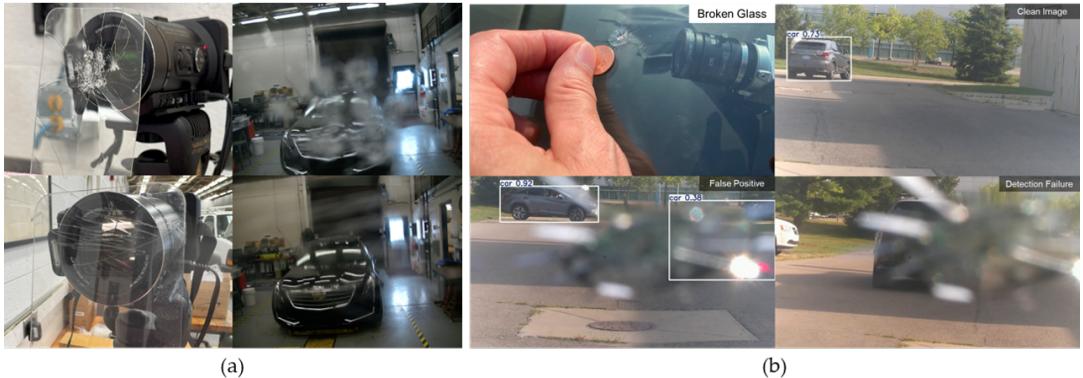
The problem of adversarial sample can be defined as follows: for a model M that classifies an input sample X correctly to its designated class i.e. $M(X) = y_{true}$, adding an error ϵ to the input sample X , results in an altered sample \hat{X} such that $M(\hat{X}) \neq y_{true}$. Thus, the injection 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 behind their generation and need access to the model. To address this, some recent research has targeted building physically relevant adversarial samples. One of the first forays into this was made by (Kurakin, Goodfellow, and Bengio 2018) who targeted the accuracy of the models in the physical world by feeding noisy images from a cell-phone camera that led the model to incorrectly classify a large fraction of the samples. Along the similar vein, (Eykholt et al. 2018) demonstrated that real traffic signs can be perturbed with simple physical stickers placed strategically to fool state-of-the-art DL algorithms almost perfectly even with viewpoint changes. Other researchers have placed adversarial images (Kong et al. 2020), translucent patches on camera (Zolfi et al. 2021) or artificial LiDAR surfaces (Tu et al. 2020) to generate samples which fool object detectors. While these prior research use physics in terms of generating the samples, they do not come 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 is still an open research question and one that has been contentious with multiple physical theories being proposed (Rouxel and Brow 2012). While the microscopic procedure of glass crack is being debated on, on a macroscopic level, the cracking



Figuur 2: (a) Indoor statisch experiment. Links: Camera met 2 verschillende patronen van gebroken gehard glas; rechts - beelden van het voertuig onder de verschillende breuken. (b) Outdoor dynamisch experiment. Linksboven - een muntgrote voorruitbreuk; rechtsboven - schoon beeld met het voertuig gedetecteerd met YOLOv8; linksonder - vals positief door de barst; rechtsonder - detectiefout door het glas. Meer voorbeelden van deze experimenten zijn opgenomen in het aanvullend materiaal.

die zich bevinden binnen de regio waar de breuk bestaat met behulp van de ground truth begrenzingskaders. We maken vervolgens gebruik van YOLOv8, Faster R-CNN (Ren et al. 2016) en Pyramid Vision Transformer (PVTv2) (Wang et al. 2022) om het percentage objecten te vinden dat faal wanneer de adversariële filters worden toegepast. We bieden ook ablatiestudies aan om de distributieverschillen tussen de drie sets afbeeldingen te begrijpen: Echt gebroken glasafbeeldingen die experimenteel zijn verzameld, echt gebroken glasafbeeldingen die online zijn verzameld en de gegenereerde afbeeldingen. We berekenen de Kullbeck-Liebler (K-L) divergentie voor deze afbeeldingsdistributies om de gelijkenis van de gegenereerde afbeeldingen met de echte gebroken glasafbeeldingen te bewijzen. We gebruiken kattenafbeeldingen uit de Kaggle Cats and Dogs dataset als controle om het verschil tussen afbeeldingsdistributies (PK) te begrijpen.

De belangrijkste bijdragen van het artikel kunnen als volgt worden samengevat:

- We bieden een nieuwe manier om glasbreuk te abstracteren door een combinatie van spanningspropagatiemethoden en minimale opspannende bomen, om fysiek kloppende gebroken glaspanelen te genereren.
- We presenteren een PBR-benadering om een realistische weergave van camerafouten te faciliteren die kan worden gebruikt met elk soort bestaand computervisie-dataset - zowel afbeeldingen als video's.
- Onze simulatie- en renderpijplijnen zijn schaalbaar en computatieve efficiënt ($\approx 1.6s$), waardoor ze zowel door de academische wereld als de industrie kunnen worden gebruikt voor het verbeteren van robuustheid en bescherming tegen out-of-distribution voor een breed scala aan toepassingen.

Achtergrond

Fysisch gebaseerde steekproeven

Het probleem van een adversariële steekproef kan als volgt worden gedefinieerd: voor een model M dat een invoersteekproef X correct classificeert naar zijn aangewezen klasse, d.w.z. $M(X) = y_{true}$, leidt het toevoegen van een ϵ aan de invoersteekproef X tot een gewijzigde steekproef X' zodanig dat $M(X') \neq y_{true}$. Dus, de injectie van de fout ϵ resulteert in een adversariële steekproef die ervoor zorgt dat het model faalt.

Hoewel het idee van adversariële manipulatie van het model al geruime tijd geleden is geïdentificeerd in de context van machine learning (Dalvi et al. 2004), is in het afgelopen decennium de focus volledig gericht geweest op de adversariële aanvallen op neurale netwerken (Szegedy et al. 2013; Goodfellow, Shlens, en Szegedy 2014). In deze artikelen toonden de onderzoekers aan dat een kleine gerichte injectie van ruis, bijna onmerkbaar voor het menselijk oog, de labels volledig veranderde (Szegedy et al. 2013) en omgekeerd, dat er beelden konden worden gegenereerd die voor mensen volledig onherkenbaar leken, maar die perfecte classificaties hadden van de DNN's (Nguyen, Yosinski, en Clune 2015).

Hoewel deze adversariële steekproeven het model testen op mogelijke fouten, ontbreekt het hen aan fysieke realiteit bij hun generatie en hebben ze toegang tot het model nodig. Om dit aan te pakken, heeft recent onderzoek zich gericht op het ontwikkelen van fysiek relevante adversariële steekproeven. Een van de eerste pogingen hiertoe werd gedaan door (Kurakin, Goodfellow en Bengio 2018), die zich richtten op de nauwkeurigheid van de modellen in de fysieke wereld door ruisachtige beelden van een mobiele telefooncamera te gebruiken, waardoor het model een groot deel van de steekproeven verkeerd classificeerde. In dezelfde lijn toonden (Eykholt et al. 2018) aan dat echte verkeersborden kunnen worden verstoord met eenvoudige fysieke stickers die strategisch zijn geplaatst om geavanceerde DL-algoritmen bijna perfect te misleiden, zelfs met veranderingen in het gezichtspunt. Andere onderzoekers hebben adversariële beelden (Kong et al. 2020), Translucent Patches op camera (Zolfi et al. 2021) of kunstmatige LiDAR-opervlakken (Tu et al. 2020) geplaatst om steekproeven te genereren die objectdetectoren misleiden. Hoewel dit eerdere onderzoek fysica gebruikt bij het genereren van de steekproeven, komen ze niet voort uit het modelleren van een rigoureus fysiek proces en we streven ernaar deze kloof in dit werk te vullen.

Theorie van gebrosten/gebroken glas

Het onderwerp van hoe glas breekt en hoe het zich verspreidt, is nog steeds een open onderzoeksraag en een diecontroversieel is met meerdere fysieke theorieën die zijn voorgesteld (Rouxel en Brow 2012). Terwijl de microscopische procedure van glasbreuk wordt bediscussieerd, is op macroniveau het barsten

dynamics is well understood. (Liu et al. 2021) analyzed the process of cracking of glass lens in the precision glass molding application using FEM with a three-dimensional model in a physical simulation software. The physical parameters were input into the software and the crack paths were analyzed using the simulation results. The authors performed a temperature and stress simulation of a high-precision three-dimensional mesh model of the molded glass. (Iben and O'Brien 2009) provided a way to generate surface fractures in variety of materials including glass. As already mentioned in the introduction, (Pfaff et al. 2014) provided the simulation of glass breaking as a thin sheet which forms the closest related work to our proposed method.

Methodology

Generating realistic glass failures require 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 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

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

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 in the form of effective stress σ_V at the impact point (V) as the stress state of the impact point. The critical stress values for the strength of glass σ_C is 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):

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

From the classical theory of strength of materials, we know that the failure in most cases is controlled by the principal stresses σ_I and σ_{II} for 2D elements. The initial crack happens either by the normal-planar crack where the fracture faces are located perpendicular to the direction of the highest principal stress σ_I (Rankine 1857) or shear-planar crack where the fracture faces coincide 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 happens perpendicular to the direction of the maximum principal stress.

From the initial impact point p_I , the stress propagation through glass is unstable since the crack grows abruptly without the need to increase external loading. From p_I ,

stress propagates in the vertex neighborhood k_i as the stress along the direction $\overrightarrow{p_I p_j}$ where $p_j \in k_i$ as

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

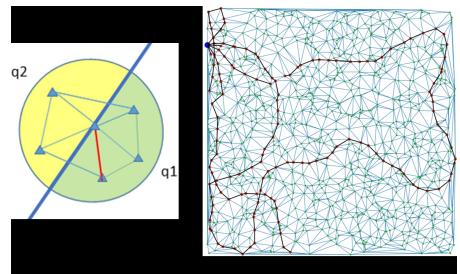


Figure 3: (a) For a splitting plane given in blue, the summed positive stress q_1 and summed negative stress q_2 are compared and the propagation happens on the side with greater summed stress and chosen node which is the closest to the splitting plane (given in red). (b) Shows how we simulate a fracture 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 (showed in Fig. 3) can then be given as:

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

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

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

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

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

Propagating the stress in all directions across all nodes, results in back-cracking as explained in (O'Brien and Hodgins 1999). To avoid it, 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 which 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.

dynamica goed begrepen. (Liu et al. 2021) analyseerden het proces van het barsten van een glazen lens in de precisieglassvormtoepassing met behulp van de Eindige Elementen Methode met een driedimensionaal model in een fysiek simulatiesoftware. De fysieke parameters werden ingevoerd in de software en de barstspaden werden geanalyseerd met behulp van de simulatie-resultaten. De auteurs voerden een temperatuur- en spanningssimulatie uit van een hoogprecisie driedimensionaal meshmodel van het gevormde glas. (Iben en O'Brien 2009) boden een manier om oppervlaktebreuken te genereren in verschillende materialen, waaronder glas. Zoals al in de inleiding vermeld, leverde (Pfaff et al. 2014) de simulatie van glasbreuk als een dunne plaat, wat het meest verwante werk vormt met onze voorgestelde methode.

Methodologie

Het genereren van realistische glasschade vereist het creëren van grootschalige natuurkundige simulaties door breukdynamica op te lossen op een getriëerd eindig-elementenrooster met glas eigenschappen.

Simulatie van gebroken glas

We vertegenwoordigen glas met behulp van deeltjes die zijn bemonsterd uit een uniforme verdeling verspreid over een vlak, beperkt in de vorm van een 2D-rooster met behulp van beperkte Delaunay-triangulatie. Dit verwijdert slecht gevormde driehoeken en voorkomt ongelijke en unrealistische randen.

Elk deeltje p_i heeft een positie x_i en heeft nabije burenkinnen een straal r die bestaande randen hebben met p_i . Wiskundig gezien vertegenwoordigt het triangulatierooster \mathcal{M} een eindige verzameling van 2-simplexen zodanig dat als

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

De scheurpatronen in glas ontstaan door spanning van de externe kracht (F) op het initiële impactpunt p_I door een specifieke vervormingswet (elasticiteit en plasticiteit) van het glas (G) aan te nemen (Kuna 2013). We berekenen vervolgens de sterkteparameters in de vorm van effectieve spanning σ_V op het impactpunt (V) als de spanningsstaat van het impactpunt. De kritische spanningswaarden voor de sterkte van glas σ_C worden gevonden met behulp van tests op eenvoudige monsters met elementaire belastingcondities (bijv. trekproef). De break treedt dan op wanneer de effectieve spanning groter is dan de kritische spanning gedeeld door de veiligheidsfactor (S):

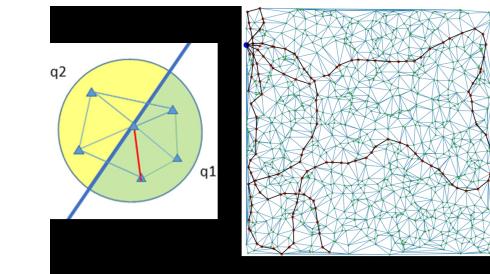
$$\sigma_V(G, F) > \frac{\sigma_C}{S}. \quad (2)$$

Uit de klassieke theorie van sterkteleer weten we dat het falen in de meeste gevallen wordt beheerst door de hoofdspanningen σ_I en σ_{II} voor 2D-elementen. De initiële scheur ontstaat ofwel door de normaal-vlakke scheur waarbij de breukvlakken zich loodrecht op de richting van de hoogste hoofdspanning σ_I bevinden (Rankine 1857), of door de schuif-vlakke scheur waarbij de breukvlakken samenvallen met de snijvlakken van de maximale schuifspanning $\tau_{max} = (\sigma_I - \sigma_{II})/2$ (Coulumb 1776). In het geval van glas gaan we ervan uit dat de initiële break loodrecht op de richting van de maximale hoofdspanning plaatsvindt.

Vanaf het initiële impactpunt p_I is de spanningsvoortplanting door glas onstabiel, aangezien de scheur abrupt groeit zonder dat er een toename van externe belasting nodig is. Vanaf p_I ,

propageert de spanning in de buurt van de hoek k_i als de spanning in de richting $\overrightarrow{p_I p_j}$ waar $p_j \in k_i$ als

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



Figuur 3: (a) Voor een splijtvlaak aangegeven in blauw, worden de opgetelde positieve spanning q_1 en opgetelde negatieve spanning q_2 vergeleken en vindt de voortplanting plaats aan de zijde met de grotere opgetelde spanning en de gekozen knoop die het dichtst bij het splijtvlaak ligt (aangegeven in rood). (b) Toont hoe we een break simuleren in een mesh die ontstaat vanuit het impactpunt (gemarkeerd in blauw) naar de knopen die spanning ervaren boven hun drempelsterkte (gemarkeerd in rood).

Met de berekende spanning voor elke rand kan de som van de positieve spanning (getoond in Fig. 3) dan worden gegeven als:

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

voor een continue oppervlakte Ω waar de indicatorfunctie is. De som van de positieve spanning voor discrete simplices in het overeenkomstige gebied A met straal R wordt gegeven als

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

Op dezelfde manier wordt de som van de negatieve spanning q_2 berekend. Dan, voor een grotere magnitude $\max(|q_1|, |q_2|)$, kiezen we de overeenkomstige rand met de hoogste concentratie van spanning i in het gegeven segment als het optimale splitsingsvlak, omdat dat de maximale spanningsverlichting biedt. Zo reist de spanning langs de maasranden en verspreidt de spanning zich bij elk knooppunt.

De recursieve toepassing van de spanningspropagatie wordt uitgevoerd totdat de spanning in alle staten convergeert, d.w.z. $\sigma_p^{(t)} \approx \sigma_p^{(t-1)} \forall p \in V$.

Het propageren van de spanning in alle richtingen over alle knooppunten resulteert in terugbarsten zoals uitgelegd in (O'Brien en Hodgins 1999). Om dit te voorkomen, propageren we alleen langs de randen waar de spanningsniveaus maximaal zijn, maar voeren we een spanningsupdate uit op alle naburige knooppunten. We gebruiken dan een minimale omspannende boom (MST) op een maas die is gemaakt met deze gespannen knooppunten. We combineren deze MST met ons initiële spanningspropagatieveld langs de randen om het uiteindelijke barstpatroon te berekenen. De MST is een effectieve abstractie omdat het de knooppunten verbindt die dichter bij elkaar liggen en binnen de hoge spanningsveld, terwijl het overbodigheden verwijdt.

Ons computationele proces van spanningsvoortplanting is gedefinieerd in Algoritme 1 in Aanvullend.

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 little 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,

$$I_1 = f_r(x, \omega_i, \omega_o, \lambda, t) \quad (7)$$

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

$$I_2 = L_i(x, \omega_i, \lambda, t) \quad (8)$$

Then, L_r can be defined as

$$L_r(x, \omega_o, \lambda, t) = \int_{\Omega} I_1 \cdot I_2 \cdot (\omega_i \cdot \mathbf{n}) d\omega_i \quad (9)$$

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

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:

$$L_r = L_i(\omega_i \cdot \hat{\mathbf{n}}) \quad (10)$$

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

$$\mathbb{E}(L_r) = \frac{|\hat{\omega}_\alpha \cdot \hat{n}_i| + |\hat{\omega}_\theta \cdot \hat{n}_i|}{2} \quad (11)$$

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

$$I_c = (I_r, I_g, I_b) \cdot \frac{\mathbb{E}(L_r)}{\sum L_r} \quad (12)$$

Focal Plane and Physical attack simulation While we are able to simulate realistic fractures, the primary use case for our work is to be able to generate simulated examples overlayed 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 happens, the light rays bounce unevenly along the crack and creates a blur (example provided in Fig. 4). We create a binary mask based on the crack pattern and then blur the fractures overlayed 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 pattern - 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 domain while the last one is a general purpose image dataset. The real broken glass pattern images are collected from FreePik website¹ and represent the baseline in our case. We collected 65 images in total and expanded them to a set of 10,000 images via image augmentation using random shifts, image flips and cropping techniques. We also generate 10,000 images using our physics simulator. 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 in a CPU. A KD-tree from the SciPy python package (Virtanen et al. 2020) using default parameters is constructed to find the approximate nearest neighbors of each particle. A Delaunay triangulation is then run 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. The stress propagation happens until a threshold of 300 units is reached. The PBR is performed on 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 in adversarial examples is that our generated adversarial patterns do not affect all pixels in an image universally. Therefore, the comparison needs to be done only for the image region where the pattern exists. For this purpose, we create a binary mask of each pattern and output the results of the objects which exist in that pattern only.

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 drop as expected with the decrease in AP corresponding to the percentage of image occupied with 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,

Fysiek-gebaseerde rendering

Zodra we de breuken op het mesh-niveau hebben gegenereerd, is ons volgende doel om een visuele weergave van deze breuken te creëren. Zoals alle PBR-technieken is onze methode gebaseerd op de microfacet theorie, die stelt dat elk oppervlak kan worden beschreven door piepkleine, perfect reflecterende spiegels genaamd microfacetten (Pharr, Jakob, en Humphreys 2023).

In overeenstemming met de microfacet theorie en energiebesparing, gebruiken we de reflectantievergelijking,

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

waarbij $L_o(x, \omega_o, \lambda, t)$ de totale spectrale stralingssterkte is van golflengte λ die naar buiten is gericht langs richting ω_o op tijdstip t , vanaf een bepaalde positie x . ω_o is de richting van het uitgaande licht. t is tijd. L_e is de uitgezonden spectrale stralingssterkte en L_r is de gereflecteerde spectrale stralingssterkte.

Laat I_1 de bidirectionele reflectantieverdelingsfunctie zijn,

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

en laat I_2 de spectrale stralingssterkte zijn die naar binnen komt richting x vanuit richting ω_i op tijdstip t .

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

Dan kan L_r worden gedefinieerd als

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

waarbij Ω de eenheidshemisfeer is geцentreerd rond de oppervlakte-normaal \mathbf{n} over ω_i zodat $\omega_i \cdot \mathbf{n} > 0$.

Door de reflectantievergelijking te abstraheren, willen we een visuele weergave van ons gebroken glasnet maken. We hebben $L_e = 0$ aangezien glas geen licht uitstraalt. Nu, voor het berekenen van L_r , beschouwen we elke barst tussen de knooppunten als een microfacet. Dan kunnen we L_r voor elke barst definiëren als:

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

Gegeven de eenheidsvectoren $(\hat{\omega}_\alpha)$ en $(\hat{\omega}_\theta)$ die respectievelijk overeenkomen met de azimut (α) en zenith (θ) hoeken, berekenen we de gemiddelde energie die op de barst valt als

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

waarbij \hat{n}_i de eenheidsoppervlakte-normaal van de barst is.

Laat (I_r, I_g, I_b) de gemiddelde intensiteit van de lichtbron zijn. Dan wordt de barstintensiteit, I_c gedefinieerd als

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

Brandpuntsvlak en Simulatie van fysieke aanval Hoewel we in staat zijn realistische breuken te simuleren, is het primaire gebruik van ons werk het genereren van gesimuleerde voorbeelden die over bestaande datasets (KITTI, BDD100k, MS-COCO) worden gelegd en deze te vergelijken met de echte dataset op de weg die we hebben gecreëerd.

Elk vastgelegd beeld zal scherpe kenmerken vertonen van de objecten in het brandpuntsvlak. De glazen behuizing die de camera bedekt, is extreem dichtbij en maakt daarom geen deel uit van het brandpuntsvlak.

Wanneer de barst ontstaat, weerkaatsen de lichtstralen ongelijkmatig langs de barst en ontstaat er een waas (voorbild gegeven in Fig. 4). We maken een binaire masker op basis van het barstpatroon en vervagen vervolgens de breuken die op het beeld zijn gelegd. Dit produceert een veraf-focus beeld. Voor een kort-focus beeld vervagen we het beeld en richten we ons op de voorgond, d.w.z. de barst.

Experimentatie

Dataset

We benchmarken twee soorten gebroken glaspatronen - echt en gesimuleerd - op drie populaire open-source datasets - KITTI (Geiger et al. 2013), BDD100k (Yu et al. 2020) en MS-COCO (Lin et al. 2014). De eerste twee vertegenwoordigen specifieke domeinen van autonoom rijden, terwijl de laatste een algemeen beeldendataset is. De echte gebroken glaspatroonafbeeldingen zijn verzameld van de FreePik-website¹ en vormen de basislijn in ons geval. We hebben in totaal 65 afbeeldingen verzameld en deze uitgebreid tot een set van 10,000 afbeeldingen via beeldvergroting met behulp van willekeurige verschuivingen, beeldomkeringen en bijnijdttechnieken. We genereren ook 10,000 afbeeldingen met behulp van onze fysicsimulator. Vervolgens leggen we deze gebroken glaspatronen met onze PBR-pijplijn over elke validatieafbeelding in de datasets en verzamelen de geaggregeerde resultaten. We gebruiken drie modelarchitecturen YOLOv8, Faster R-CNN en PVTv2 model met voorgetrainde gewichten om objectdetectieresultaten te genereren.

Implementatie

Ons simulatiemodel is ontwikkeld door willekeurig 10^4 deeltjes te bemonsteren uit een uniforme ruimtelijke verdeling in het gegeven frame in een CPU. Een KD-boom uit het SciPy Python-pakket (Virtanen et al. 2020) met standaardparameters wordt geconstrueerd om de benaderde dichtstbijzijnde buren van elk deeltje te vinden. Vervolgens wordt een Delaunay-triangulatie uitgevoerd op de deeltjes om een beperkt driehoekig netwerk te creëren. We gebruiken een impactkracht van 500 eenheden met een willekeurig impactpunt en een willekeurige impactvector. De spanningspropagatie vindt plaats totdat een drempel van 300 eenheden is bereikt. De PBR wordt uitgevoerd op de CPU door de methoden te implementeren die in de vorige sectie zijn beschreven met behulp van OpenCV en Python.

Resultaten en Discussie

Een belangrijke verschuiving ten opzichte van de meeste eerdere werken in adversariële voorbeelden is dat onze gegenereerde adversariële patronen niet alle pixels in een afbeelding universeel beïnvloeden. Daarom moet de vergelijking alleen worden gemaakt voor het afbeeldingsgebied waar het patroon zich bevindt. Voor dit doel maken we een binaire maskering van elk patroon en geven we de resultaten weer van de objecten die alleen in dat patroon bestaan.

Tabel 1 toont de resultaten van de gemiddelde precisie (AP) onder de adversariële afbeeldingen die zijn gegenereerd met behulp van de twee soorten scheurpatronen (online verzameld en gesimuleerd) voor verschillende klassen. Voor KITTI daalt de AP van andere klassen zoals verwacht met de afname in AP die overeenkomt met het percentage van de afbeelding dat wordt ingenomen door de vrachtwagenklasse, die de grootste daling registreert. Voor BDD100K met PVTv2-B0 zien we dat de daling in AP het grootst is in de gesimuleerde afbeeldingen, maar over het algemeen,

¹<https://www.freepik.com>

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 images. x provides the overlay relation between dataset and glass-crack type. Clean x Dataset - refers to directly the particular images without any adversarial sample. RO x Dataset - refers to Real images of cracked glass collected online overlayed on clean images. Sim x Dataset - refers to simulated crack patterns overlayed 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 the steepest drop. For MS-COCO, we aggregated the AP for the super-categories: person, vehicles and food. This is because a lot of objects in MS-COCO occupy smaller area in the image frame making it difficult to get 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 because of 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

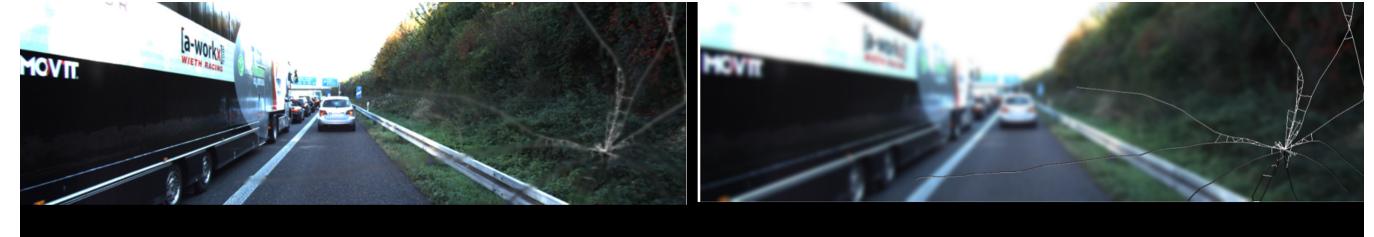
Our results indicate that the simulated images obtain a similar adversarial effect as the real images. Thus, an important ablation study for us is to understand how close the simulated crack patterns are to the real cracked glass patterns and those collected online. We form 5 distributions

- 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 compute how similar they are to each other (see Fig. 6). In order to provide a control, we compare KITTI to images of cats from the Kaggle dataset, providing a K-L divergence of 2.434. In that scale, the PBR images of broken glass have a difference of 0.36 to the real broken glass patterns while the broken glass filters overlaid on KITTI images have similar K-L divergence.

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



Figuur 4: (a) Toont het gesimuleerde beeld met de weg en voertuigen in het brandpuntsvlak (PBR en ver-focus). (b) geeft het gesimuleerde barstpatroon in het brandpuntsvlak weer (PBR en kort focus).

Tabel 1: Gemiddelde precisie (in procenten) van verschillende klassen in KITTI, BDD100k en MS-COCO onder verschillende adversariële afbeeldingen. x geeft de overlay-relatie tussen dataset en glasscheurtypen weer. Schoon x Dataset - verwijst direct naar de specifieke afbeeldingen zonder enige adversariële steekproef. RO x Dataset - verwijst naar echte afbeeldingen van gebroken glas die online zijn verzameld en over schone afbeeldingen zijn gelegd. Sim x Dataset - verwijst naar gesimuleerde scheurpatronen die over schone afbeeldingen zijn gelegd.

Dataset	IoU-drempel	Categorie	Schoon x Dataset	RO x Dataset	Sim x Dataset
KITTI (YOLOv8)	0.5	Voetganger	25.64	69.72	17.84
		Vrachtwagen	12.39	3.59	3.76
		Car	58.99	50.7	57.73
	0.75	Voetganger	6.83	33.88	6.02
		Vrachtwagen	11.29	2.67	2.79
		Car	31.25	23.85	30.15
BDD100k (PVTv2)	0.5	Voetganger	66.47	54.33	25.95
		Vrachtwagen	61.97	52.83	52.02
		Car	80.37	70.14	56.78
	0.75	Voetganger	27.06	22.72	10.60
		Vrachtwagen	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
		Voertuigen	2.14	1.45	1.87
		Food	35.34	28.07	30.65
	0.75	Person	0.032	0.022	0.023
		Voertuigen	1.56	1.05	1.07
		Food	24.59	18.85	22.00

de trend wordt voortgezet met de voetgangersklasse die de sterkste daling vertoont. Voor MS-COCO hebben we de AP voor de supercategorieën: persoon, voertuigen en voedsel geaggregeerd. Dit komt omdat veel objecten in MS-COCO een kleiner gebied in het beeldkader innemen, waardoor het moeilijk is om betekenisvolle resultaten uit alle categorieën te halen. Een zeer intrigerend resultaat is dat de voetgangersklasse een veelvoudige toename in AP vertoont onder de echte gebroken glaspatronen. Hoewel deze trend misschien contra-intuïtief lijkt, sluit het aan bij de resultaten in Fig. 2 waar het vertrouwen van de auto toeneemt vanwege een rand. Dit toont in feite aan dat de AP sterk afhankelijk is van het barstpatroon, waardoor het uiterst belangrijk is om verdedigingsmethodologieën te creëren om deze adversariële aanvallen te mitigeren.

Ablatiestudies

Onze resultaten geven aan dat de gesimuleerde beelden een vergelijkbaar adversarisch effect hebben als de echte beelden. Daarom is een belangrijke ablatiestudie voor ons om te begrijpen hoe dicht de gesimuleerde barstpatronen bij de echte gebroken glaspatronen en die online verzameld zijn. We vormen 5 distributies

- Echt dataset van de weg (afgebeeld in Fig. 2)

- Barstpatronen verzameld online (Fig. 5 linksboven)
- Gesimuleerde barstpatronen (Fig. 5 linksonder)
- Gesimuleerde barstpatronen overgelegd op KITTI (Fig. 5 rechtsonder)
- Barstpatronen online verzameld, overgelegd op KITTI (Fig. 5 rechtsboven)

We berekenen nu de K-L divergentie tussen al deze distributies om te bepalen hoe vergelijkbaar ze zijn met elkaar (zie Fig. 6). Om een controle te bieden, vergelijken we KITTI met afbeeldingen van katten uit de Kaggle dataset, wat een K-L divergentie van 2.434 oplevert. Op die schaal hebben de PBR-afbeeldingen van gebroken glas een verschil van 0.36 met de echte gebroken glaspatronen, terwijl de gebroken-glasfilters overgelegd op KITTI-afbeeldingen een vergelijkbare K-L divergentie hebben.

Fig. 7 toont een analyse van de rekentijd voor elk van onze modules en over verschillende aantal deeltjes. We voeren deze analyse uit op 100 runs, waarbij we willekeurige impactpunten, impacthoeken en meshstructuren genereren met een vast aantal deeltjes. Het verschil in rekentijd voor verschillende runs kan worden toegeschreven aan het impactpunt en de impacthoek.

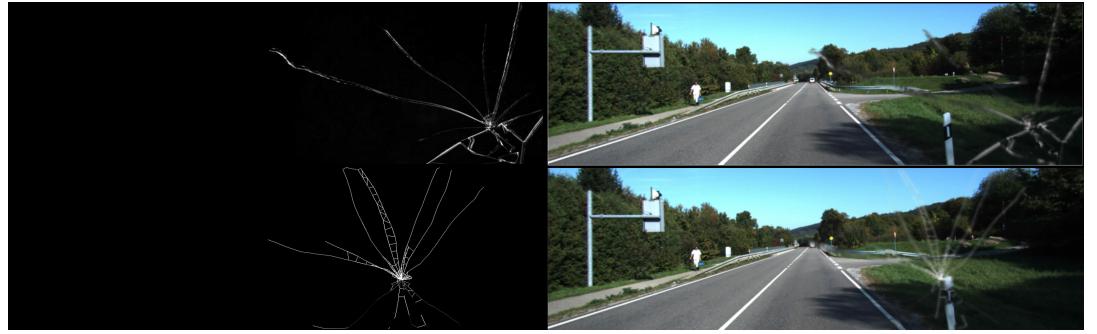


Figure 5: Top left - Crack pattern collected online on Freepik; top right - online crack pattern overlayed on KITTI; bottom left - simulated crack pattern with PBR; bottom right - simulated crack pattern overlayed on KITTI.

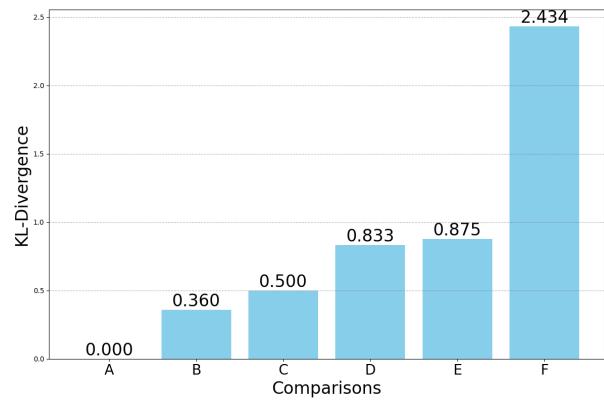


Figure 6: K-L divergence of different pairs of image distributions. Datasets: RC - Real on-road dataset (see Fig. 2), KITTI and Cats. Filters: RO - Real (collected online) and Sim - Simulated. K-L divergence between (x - overlay relation): A - (Sim x KITTI) vs (Sim x KITTI); B - (Sim vs RO); C - (Clean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

pact angle. The cracking visualization and render time also vary owing to different sized masks formed due to varying fracture patterns. We also vary the number of particles and see 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 resulting 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 embed that to 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

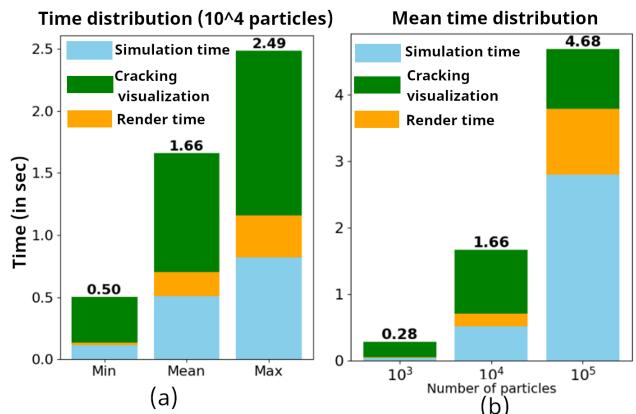
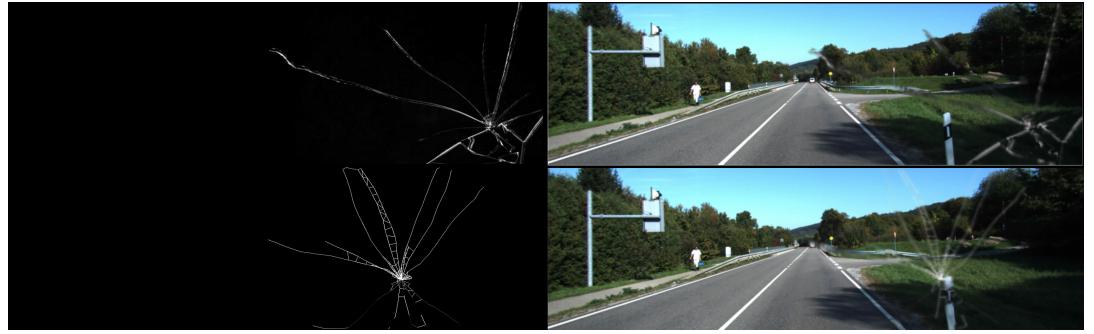


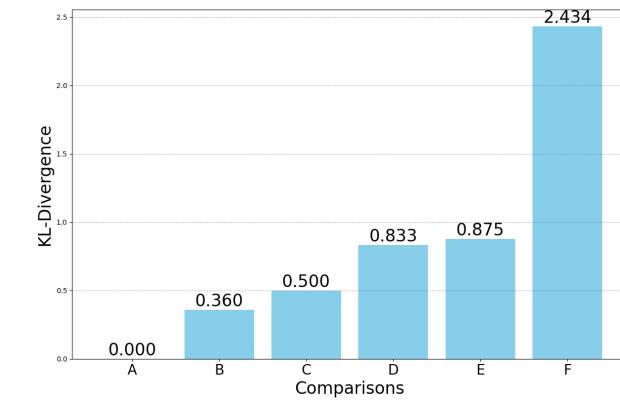
Figure 7: (a) Mean time taken by different modules of our pipeline across 100 runs. (b) The minimum, maximum and mean time taken by different modules across 100 runs for a 10^4 particle mesh. For these plots, we showcase the time taken for simulation (simulation time), converting the mesh to glass (cracking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

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 patterns, impacting latent space encoding rather than causing direct occlusion, resulting in incorrect detections.

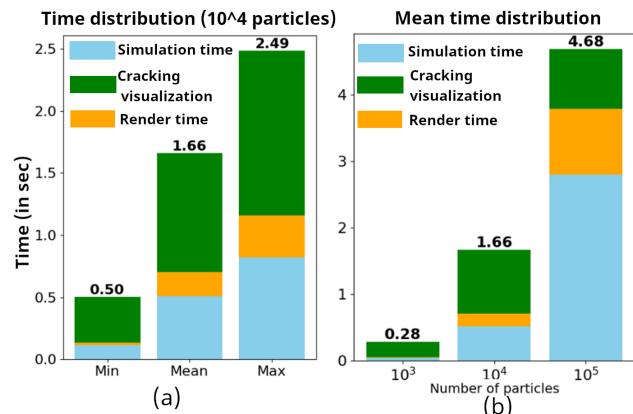
While this work introduces a physics-based method for broken glass pattern generation specifically, camera failures encompass other effects such as sun-glare, overexposure, underexposure, condensation etc. Our future work will focus on creating an adversarial toolbox for realistic generation of these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



Figuur 5: Linksboven - Scheurpatroon online verzameld op Freepik; rechtsboven - online scheurpatroon overgelegd op KITTI; linksonder - gesimuleerd scheurpatroon met PBR; rechtsonder - gesimuleerd scheurpatroon overgelegd op KITTI.



Figuur 6: K-L divergentie van verschillende paren van afbeeldingsdistributies. Datasets: RC - Reëel dataset op de weg (zie Fig. 2), KITTI en Katten. Filters: RO - Reëel (online verzameld) en Sim - Gesimuleerd. K-L divergentie tussen (x - overlay relatie): A - (Sim x KITTI) vs (Sim x KITTI); B - (Sim vs RO); C - (Schoon RC vs KITTI); D - (Gebroken RC) vs (RO x KITTI); E - (Gebroken RC) vs (Sim x KITTI); F - KITTI vs Katten.



Figuur 7: (a) Gemiddelde tijd die door verschillende modules van onze pijplijn wordt genomen over 100 runs. (b) De minimale, maximale en gemiddelde tijd die door verschillende modules wordt genomen over 100 runs voor een 10^4 deeltjesnet. Voor deze grafieken tonen we de tijd die nodig is voor simulatie (simulatietijd), het omzetten van het net naar glas (barstvisualisatie) en uiteindelijk het renderen (rendertijd).

kwetsbaarheden uit te buiten. We gaan uit van geen kennis van de modelattributen, gewichten of architectuur, waardoor aanvallen overdraagbaar zijn tussen verschillende modellen. Fysieke adversariële methoden (Translucent Patch, RP2) kunnen allemaal worden aangeduid als oclusies van ofwel de camera of de objecten die worden vastgelegd. De adversariële aard komt voort uit het effect van de modelinference als gevolg van deze oclusies. Onze PBR-pijplijn mengt de barsten met bronafbeeldingen als translucente, wazige patronen, wat invloed heeft op de latente ruimte-encoder in plaats van directe oclusies te veroorzaken, resulterend in onjuiste detecties.

Conclusie en Toekomstige Scope

We hebben een nieuwe klasse van adversariële fouten geïntroduceerd die voorkomen uit het fysieke proces van fouten in de camera. In dit artikel bieden we een benadering om een realistisch gebroken glaspatroon te genereren vanuit een fysieke simulatie en dat vervolgens in bestaande afbeeldingsdatasets te integreren met behulp van fysiek gebaseerde rendering. We tonen aan dat de gesimuleerde adversariële afbeeldingen kunnen leiden tot significante fouten in objectdetectie.

In dit werk behandelen we black-box adversariële aanvallen die voorkomen uit fysieke fenomenen die in de echte wereld natuurlijk voorkomen, en niet kunstmatig zijn gemaakt om specifieke model

Hoewel dit werk specifiek een op fysica gebaseerde methode voor het genereren van gebroken glaspatronen introduceert, omvatten camerastorings ook andere effecten zoals zonneschittering, overbelichting, onderbelichting, condensatie, enz. Ons toekomstig werk zal zich richten op het creëren van een adversariële gereedschapskist voor de realistische generatie van deze effecten met behulp van fysica en deze vervolgens plaatsen op bestaande afbeeldingsdatasets en auto-simulatieplatforms om verder onderzoek op dit gebied van gedeeltelijke camerastorings te bevorderen.

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 cameras failures and their effects in autonomous driving applications. *IEEE Transactions on Dependable and Secure Computing*.
- Coulumb, C.-A. 1776. Essai sur une application des regles des maximis et minimis a quelques problemes de statique relatifs, a la 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.
- Eykholz, 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 of the IEEE conference on computer vision and pattern recognition*, 1625–1634.
- Geiger, A.; Lenz, P.; Stiller, C.; and Urtasun, R. 2013. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*.
- Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.
- Iben, H. N.; and O’Brien, J. F. 2009. Generating surface crack patterns. *Graphical Models*, 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. *Solid mechanics and its 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 conference on machine learning*, 3896–3904. PMLR.
- Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, 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 *Proceedings of ACM SIGGRAPH 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. <https://www.kaggle.com/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 of Applied 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. *arXiv preprint arXiv: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, İ.; 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.

References

- Akhtar, N.; en Mian, A. 2018. Bedreiging van adversariële aanvallen op deep learning in computervisie: Een overzicht. *Ieee Access*, 6: 14410–14430.
- Carlini, N.; en Wagner, D. 2017. Adversarial voorbeelden zijn niet gemakkelijk te detecteren: Het omzeilen van tien detectiemethoden. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 3–14.
- Ceccarelli, A.; en Secci, F. 2022. Falen van RGB-camera's en hun effecten in toepassingen voor autonoom rijden. *IEEETransactions on Dependable and SecureComputing*. Coulumb, C.-A. 1776. Essai sur une application des regles des maximis et minimis a quelques problemes de statique relatifs, a la architecture. *Mem. Acad. Roy. Div. Sav.*, 7: 343–387. Dalvi, N.; Domingos, P.; Mausam; Sanghai, S.; en Verma, D. 2004. Adversarial classificatie. 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.; en Song, D. 2018. Robuste fysieke-wereld aanvallen op diepe leermethoden voor visuele classificatie. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1625–1634. Geiger, A.; Lenz, P.; Stiller, C.; en Urtasun, R. 2013. Vision ontmoet Robotica: De KITTI Dataset. *International Journal of RoboticsResearch(IJRR)*. Goodfellow, I. J.; Shlens, J.; en Szegedy, C. 2014. Verklaren en benutten van adversarial voorbeelden. *arXiv preprintarXiv:1412.6572*. Iben, H. N.; en O’Brien, J. F. 2009. Genereren van oppervlakte scheurpatronen. *GraphicalModels*, 71(6): 198–208. Jocher, G.; Chaurasia, A.; en Qiu, J. 2023. Ultralytics YOLO. Kong, Z.; Guo, J.; Li, A.; en Liu, C. 2020. Physgan: Genereren van fysieke-wereld-resistente adversarial voorbeelden voor autonoom rijden. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14254–14263. Kuna, M. 2013. Eindige elementen in breukmechanica. *Solidmechanics andits applications*, 201: 153–192. Kurakin, A.; Goodfellow, I. J.; en Bengio, S. 2018. Adversarial voorbeelden in de fysieke wereld. In *Artificial intelligence safety and security*, 99–112. Chapman and Hall/CRC. Li, J.; Schmidt, F.; en Kolter, Z. 2019. Adversarial camerastickers: Een fysieke camera-gebaseerde aanval op diepe leersystemen. In *International conference on machine learning*, 3896–3904. PMLR. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; en 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.
- van Schrick, D. 1997. Opmerkingen over terminologie op het gebied van supervisie, foutdetectie en diagnose. *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, İ.; 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: Fundamentele algoritmen voor wetenschappelijk rekenen in Python. *Nature Methods*, 17: 261–272. Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; en Shao, L. 2022. Pvt v2: Verbeterde baselines met pyramid vision transformer. *Computational visualmedia*, 8(3): 415–424.
- Liu, Y.; Xing, Y.; Li, C.; Yang, C.; en Xue, C. 2021. Analyse van lensbreuk in precisieglassvormen met de eindige-elementenmethode. *Applied Optics*, 60(26): 8022–8030. Nguyen, A.; Yosinski, J.; en Clune, J. 2015. Diepe neurale netwerken zijn gemakkelijk te misleiden: Hoge vertrouwensvoorspellingen voor onherkenbare beelden. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 427–436. O'Brien, J. F.; en Hodgins, J. K. 1999. Grafische modellering en animatie van broze breuk. In *Proceedings of ACM SIGGRAPH 1999*, 137–146. ACM Press/Addison-Wesley Publishing Co. Pfaff, T.; Narain, R.; De Joya, J. M.; en O'Brien, J. F. 2014. Adaptief scheuren en barsten van dunne platen. *ACM Transactions on Graphics (TOG)*, 33(4): 1–9. Pharr, M.; Jakob, W.; en Humphreys, G. 2023. *Fysiek gebaseerde rendering: Van theorie tot implementatie*. MIT Press. PK, A. ??? Kaggle katten en honden mini dataset. h t t p s : / / www.kaggle.com/datasets/aleemaparakatta/cats-and-dogs-mini-dataset. Geraadplegd: 2024-09-30. Rankine, W. J. M. 1857. II. Over de stabilitet van losse aarde. *Philosophical transactions of the Royal Society of London*, (147): 9–27. Ren, S.; He, K.; Girshick, R.; en Sun, J. 2016. Snellere R-CNN: Naar real-time objectdetectie met regio-voorstelnetwerken. *IEEETransactions on pattern analysis and machine intelligence*, 39(6): 1137–1149. Rouxel, T.; en Brow, R. K. 2012. De stroming en breuk van geavanceerde glazen—een overzicht. *International Journal of Applied Glass Science*, 3(1): 1–2. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; en Fergus, R. 2013. Intrigerende eigenschappen van neurale netwerken. *arXivpreprintarXiv:1312.6199*. Tu, J.; Ren, M.; Manivasagam, S.; Liang, M.; Yang, B.; Du, R.; Cheng, F.; en Urtasun, R. 2020. Fysiek realiseerbare vijandige voorbeelden voor LiDAR objectdetectie. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 13716–13725. van Schrick, D. 1997. Opmerkingen over terminologie op het gebied van supervisie, foutdetectie en diagnose. *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, İ.; 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.; en SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamentele algoritmen voor wetenschappelijk rekenen in Python. *Nature Methods*, 17: 261–272. Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; en Shao, L. 2022. Pvt v2: Verbeterde baselines met pyramid vision transformer. *Computational visualmedia*, 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 15232–15241.

Yu, F.; Chen, H.; Wang, X.; Xian, W.; Chen, Y.; Liu, F.; Madhavan, V.; en Darrell, T. 2020. BDD100K: Een divers rijdataset voor heterogeen multitask leren. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2636–2645.

Zolfi, A.; Kravchik, M.; Elovici, Y.; en Shabtai, A. 2021. De translucent patch: Een fysieke en universele aanval op objectdetectoren. In *Proceedings of the IEEE/CVF conference on computer vision and pattern 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

```

1:  $pt \leftarrow$  Impact Point
2:  $F \leftarrow$  Impact Force
3:  $PE \leftarrow$  Parent Edge
4:  $v \leftarrow$  Impact Vector
5:  $R \leftarrow$  Nearest neighbor radius
6:
7: procedure PROPAGATESTRESS( $Pt, F, V, PE$ )
8:    $frontiers \leftarrow KDT\!ree - queryRadius(R)$ 
9:    $NN \leftarrow \frac{frontiers - pt}{\|frontiers - pt\|}$ 
10:   $\cos(\theta) \leftarrow NN \cdot v$ 
11:   $stress \leftarrow calculateStress(\cos(\theta), F)$ 
12:   $frontiers \leftarrow frontiers[argmax(stress)]$ 
13:   $v \leftarrow v[argmax[stress]]$ 
14:   $PE \leftarrow PE[argmax[stress]]$ 
15:  PROPAGATESTRESS( $Pt, F, V, PE$ )
16: 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 which corresponds to broken lens pattern (final image of Fig. 8).

Algoritme van spanningspropagatie

Algoritme 1 beschrijft de procedure voor het simuleren van de propagatie van spanning door een materiaal na een impactgebeurtenis. Het algoritme neemt als invoer de locatie van de impact (pt), de grootte van de impactkracht (F), de impactrichtingsvector (v) en de ouderzijde (PE) die geassocieerd is met de impactlocatie. Het gebruikt ook een nabijste buurstraal R om de set van kandidaatlocaties voor spanningspropagatie te bepalen.

Algoritme 1: Spanningspropagatie

```

1:  $pt \leftarrow$  Impactpunt
2:  $F \leftarrow$  Impactkracht
3:  $PE \leftarrow$  Ouderzijde
4:  $v \leftarrow$  Impactvector
5:  $R \leftarrow$  Nabijste buurstraal
6:
7: procedure PROPAGEREERSTRESS( $Pt, F, V, PE$ )
8:    $frontiers \leftarrow KDT\!ree - queryRadius(R)$ 
9:    $NN \leftarrow \frac{frontiers - pt}{\|frontiers - pt\|}$ 
10:   $\cos(\theta) \leftarrow NN \cdot v$ 
11:   $stress \leftarrow calculateStress(\cos(\theta), F)$ 
12:   $frontiers \leftarrow frontiers[argmax(stress)]$ 
13:   $v \leftarrow v[argmax[stress]]$ 
14:   $PE \leftarrow PE[argmax[stress]]$ 
15:  PROPAGATESTRESS( $Pt, F, V, PE$ )
16: einde procedure
```

Eerst gebruikt het een KD-boom datastructuur om efficiënt alle punten (grenzen) binnen een gegeven straal R van het impactpunt op te vragen. Voor elke grens berekent het een eenheidsrichtingsvector van het impactpunt naar de grens (NN). Vervolgens projecteert het de impactvector v op deze richting om de cosinusgelijkenis $\cos(\theta)$ te verkrijgen, waarmee de hoekrelatie tussen de impactrichting en de kandidaat-propagatierichting wordt vastgelegd. Voor elke kandidaat wordt de resulterende waarde gebruikt, samen met de impactkracht, om de overeenkomstige spanning op dat punt te berekenen. Het algoritme selecteert vervolgens de kandidaat met de maximale spanningswaarde. De impactvector v en ouderzijde PE worden bijgewerkt om overeen te komen met deze nieuwe richting. Het proces wordt recursief herhaald, waardoor de gesimuleerde spanningsgolf iteratief door het materiaal kan voortplanten langs het pad van de grootste spanningsoverdracht.

Deze benadering is bedoeld om na te bootsen hoe spanning vanaf een impactpunt zich waarschijnlijk door een materiaal zal verspreiden—voorkeur gevend aan paden die worden gedefinieerd door zowel geometrische nabijheid als mechanische uitlijning met de oorspronkelijke impact.

De uiteindelijke output van de simulatie is de realisatie van het mesh als een afbeelding die overeenkomt met het patroon van een gebroken lens (eindafbeelding van Fig. 8).

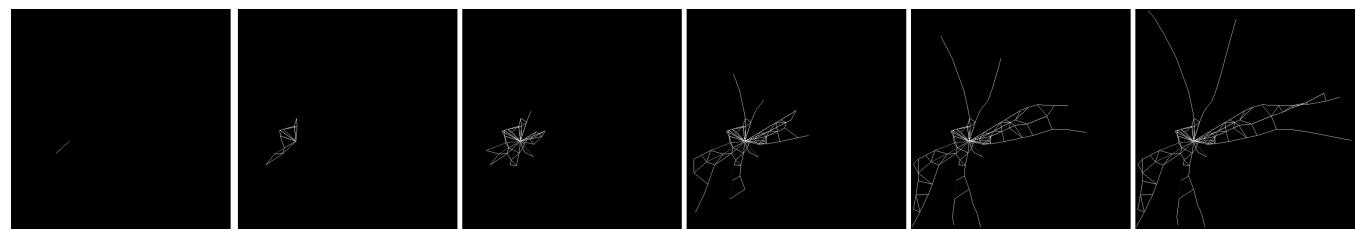
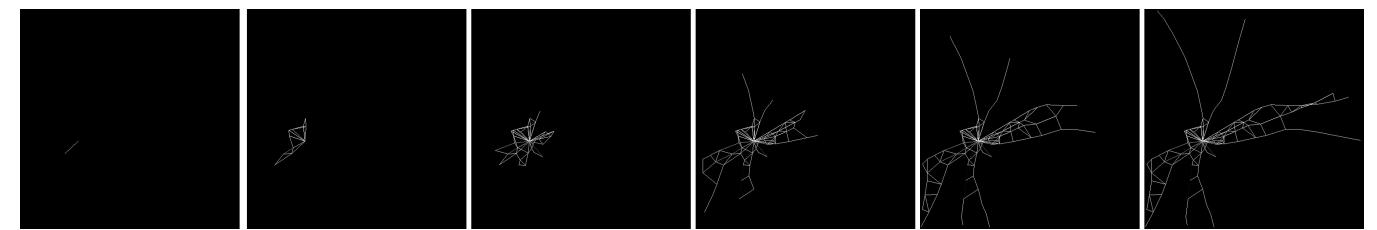


Figure 8: An animation of fracturing of a lens simulated by setting the stress field and applying PBR.



Figu^{re} 8: Een animatie van het breken van een lens gesimuleerd door het spanningsveld in te stellen en PB toe te passen R.

Static Experiment

In order to understand the effect of these fractures on the resultant images, we first conduct a indoor static experiment as referenced in Section Introduction. We use various tempered glass sheets for this experiment, which we break randomly using a small hammer with one 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 for the tempered glass.

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

Fig. 10 shows some of the fractures/scratched patterns on the tempered glass. These patterns were intentionally randomized, employing multiple focal points and different levels of force to mimic the unpredictable and varied nature of real-world glass damage. By applying diverse 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 varied the focal lengths of the camera considerably to understand how the images look under near- and far-focus. The outputs show that even minor scratched patterns show up in the image output whereas much stronger multi-fracture pattern 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 we were surprised 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 because of enhanced edges around them. This wasn't an edge artifact but instead the glass crack acting as an additional edge boundary clearly separating the pedestrian and the background. A similar result was also observed in [1] where the overall AP was increased in adversarial images.



Figure 9: Experimental setup for collecting images impacted by scratched/broken outer layers for a camera. (a) shows the entire setup for taking adversarial images. (b) shows the position of the camera w.r.t. the scene being captured. (c) shows the scene being captured by the camera

Statisch Experiment

Om het effect van deze breuken op de resulterende beelden te begrijpen, voeren we eerst een statisch experiment binnenshuis uit zoals vermeld in Sectie Inleiding. Voor dit experiment gebruiken we verschillende geharde glasplaten, die we willekeurig breken met een kleine hamer met één of meerdere breekpunten. Vervolgens plaatsen we een 36 MP JVC GC-PX10 hybride camera gemonteerd op een statief met een klem voor het geharde glas.

Fig. 9(a) toont de gedetailleerde opstelling met de camerabevestiging en het geharde glas dat met een klem op zijn plaats wordt gehouden. Fig. 9(b) toont het beeld dat door de camera is vastgelegd en Fig. 9(c) toont het enkele voertuig dat als het primaire object wordt vastgelegd door de camera door het geharde glas. De scène wordt verlicht met bovenliggende fluorescentielampen.

Fig. 10 toont enkele van de breuk- en kraspatronen op het geharde glas. Deze patronen werden opzettelijk gerandomiseerd, waarbij meerdere brandpunten en verschillende niveaus van kracht werden gebruikt om de onvoorspelbare en gevarieerde aard van echte glasschade na te bootsen. Door diverse krachten toe te passen, konden we een spectrum van breuken en krassen produceren, variërend van fijne oppervlaktebeschadigingen tot meer uitgesproken breuken. Deze aanpak werd gekozen omdat de soorten schade die glassurfaces in werkelijke omstandigheden kunnen tegenkomen—zoals die veroorzaakt door impact, puin of omgevingsstressoren—nauwkeurig te repliceren, waardoor de relevantie en realisme van onze experimentele opstelling worden gewaarborgd. Deze representatieve schadepatronen stellen ons in staat om effectiever de invloed van glasimperfecities op sensorprestaties en objectdetectie-algoritmen te analyseren.

Twee verschillende breukpatronen en hun resulterende beelden worden getoond in Fig. 11 en Fig. 12. We willen opmerken dat we de brandpuntsafstanden van de camera aanzienlijk hebben gevarieerd om te begrijpen hoe de beelden eruitzien bij dichtbij- en veraf-focus. De resultaten laten zien dat zelfs kleine kraspatronen zichtbaar zijn in de beeldoutput, terwijl een veel sterker multi-breukpatroon bijna het hele beeld kan vervagen. Dit experiment biedt de intuïtie waarop ons simulatie- en visualisatieframework is gebouwd.

Verhoogde AP voor voetgangers in KITTI

We willen erop wijzen dat de verhoogde AP voor de voetgangersklasse iets was waar zelfs wij aanvankelijk door verrast waren. Echter, een zorgvuldige kwalitatieve diepgaande analyse hielp ons te begrijpen dat dit gebeurde als gevolg van de glasbreuken die het voor het model gemakkelijker maakten om voetgangers te classificeren vanwege verbeterde randen rondom hen. Dit was geen randartefact, maar in plaats daarvan ungeerde de glasbreuk als een extra randgrens die de voetganger en de achtergrond duidelijk scheidde. Een vergelijkbaar resultaat werd ook waargenomen in [1] waar de algehele AP werd verhoogd in adversariële beelden.



Figuur 9: Experimentele opstelling voor het verzamelen van beelden die worden beïnvloed door bekraaste/gebroken buitenlagen voor een camera. (a) toont de volledige opstelling voor het maken van adversariële beelden. (b) toont de positie van de camera ten opzichte van de scène die wordt vastgelegd. (c) toont de scène die door de camera wordt vastgelegd.



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



Figuur 10: Enkele breuken/krassenpatronen op het glas dat we gebruikten voor het verzamelen van de afbeeldingen. (a) Een scherpe kracht die loodrecht op het glasoppervlak wordt uitgeoefend, waardoor breuken radiaal ontstaan. (b) en (c) repliceren een glas met krassen.

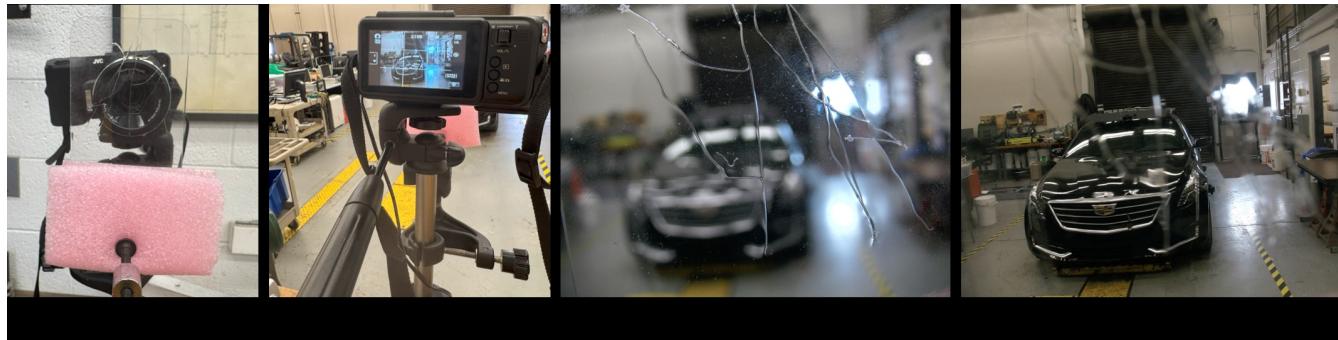
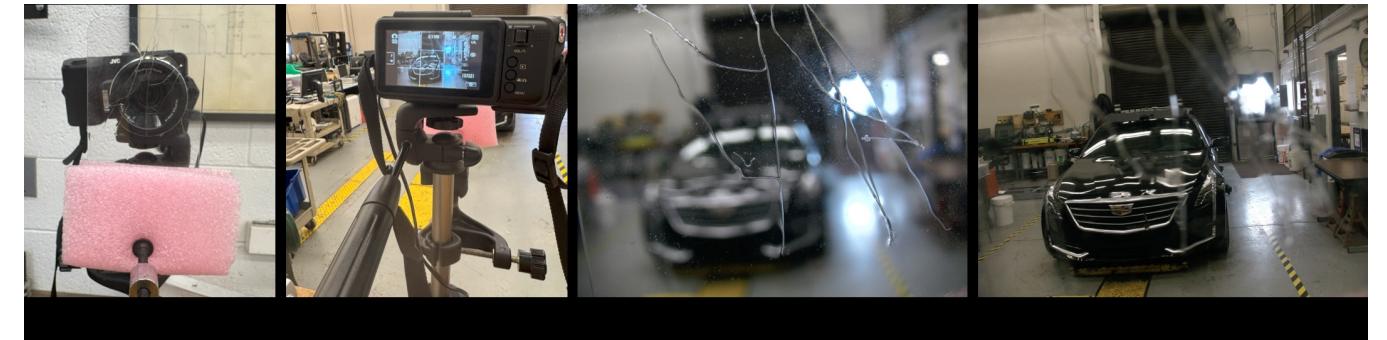


Figure 11: (a) Shows the scratched pattern placed in front of the camera, (b) shows the camera POV. (c) shows the image captured by the camera (short-focus). (d) shows the image captured by the camera (far-focus)



Figuur 11: (a) Toont het krassenpatroon geplaatst voor de camera, (b) toont het camerastandpunt. (c) toont de afbeelding vastgelegd door de camera (kortere focus). (d) toont de afbeelding vastgelegd door de camera (verre focus).

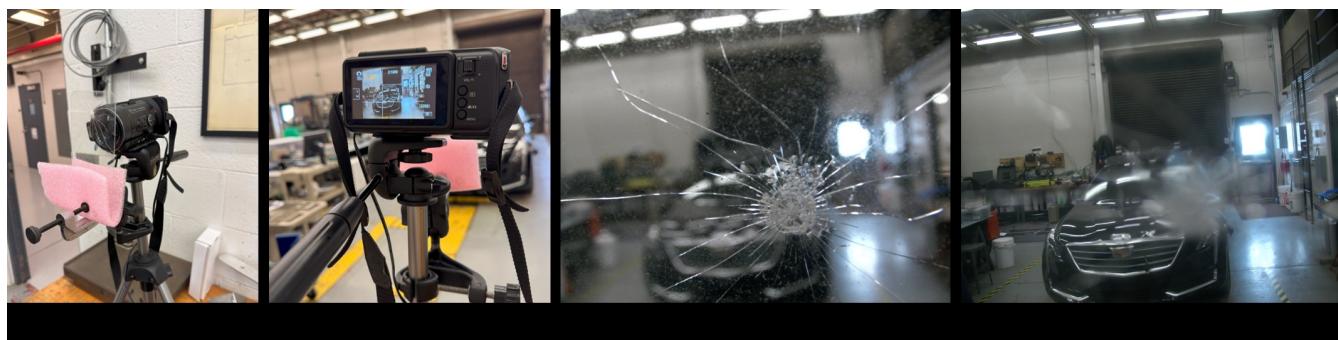
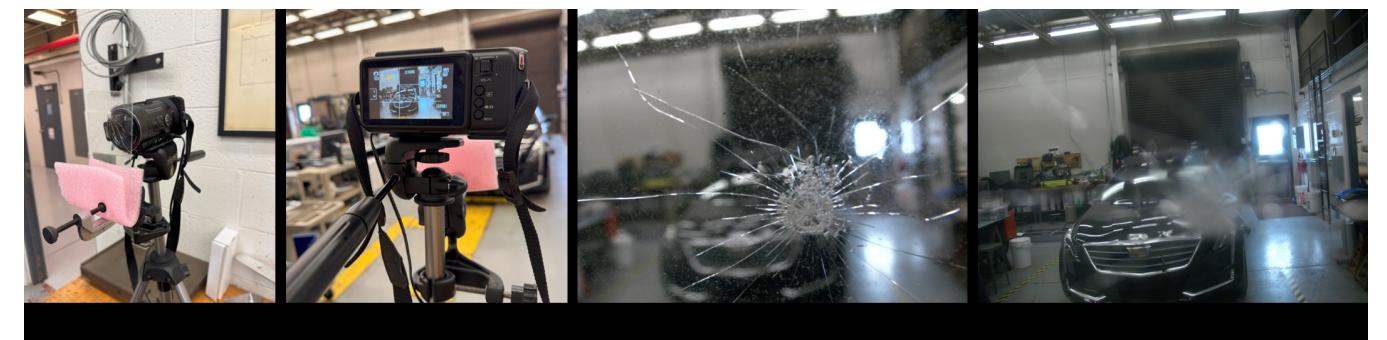


Figure 12: (a) Shows the broken glass pattern in front of the camera, (b) shows the camera POV. (c) shows the image captured by the camera (short-focus). (d) shows the image captured by the camera (far-focus)



Figuur 12: (a) Toont het gebroken glaspatroon voor de camera, (b) toont het camerastandpunt. (c) toont de afbeelding vastgelegd door de camera (kortere focus). (d) toont de afbeelding vastgelegd door de camera (verre focus).

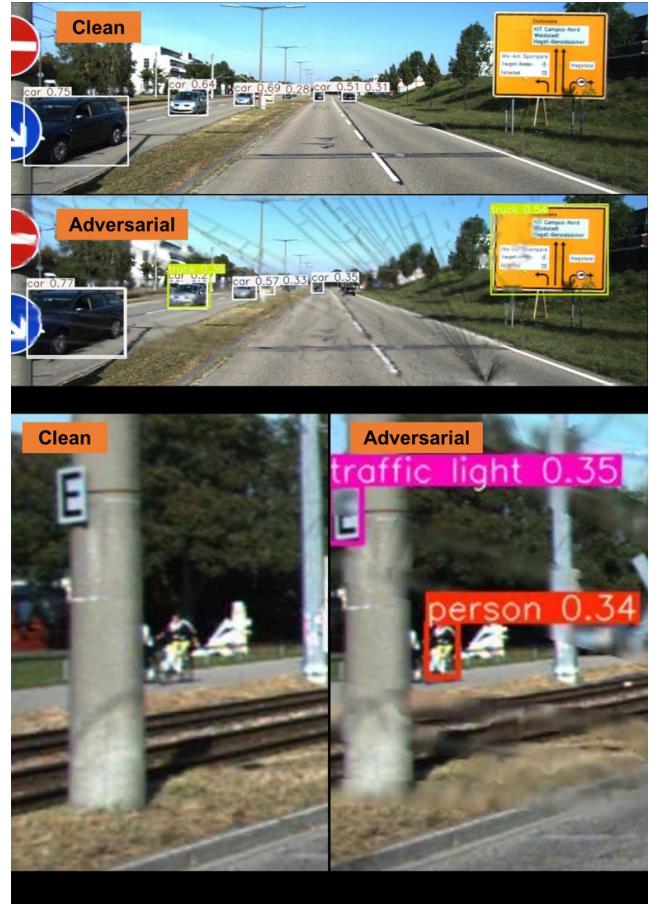
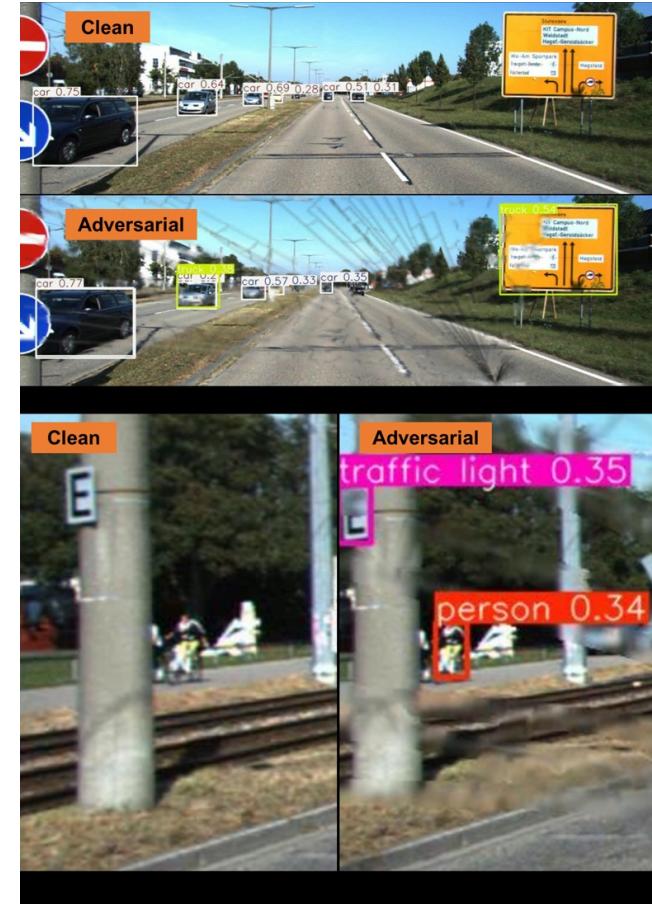


Figure 13: (a) Top - detections on a clean image; bottom - detections on an adversarial image.(b) YoLo fails to detect the person (c) Glass cracks allows the model to detect the person.



Figuur 13: (a) Boven - detecties op een schoon beeld; onder - detecties op een adversarief beeld. (b) YoLo slaagt er niet in de persoon te detecteren (c) Glasbreuken stellen het model in staat de persoon te detecteren.

Dynamic Experiment

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

Dynamisch Experiment

In deze sectie beschrijven we het dynamische experiment dat wordt genoemd in Sectie Inleiding. We voeren dit experiment uit om de temporele verstoring te begrijpen die door een barst wordt geïntroduceerd. We gebruiken een barst in de voorruit van een voertuig en plaatsen een kleine camera op het dashboard achter de barst. Vervolgens fotograferen we twee dynamische objecten - een voertuig en een voetganger terwijl ze door de scène bewegen. Fig. 14 biedt enkele specifieke beeldframes met inferentie van YOLOv8 voor de voertuigklasse. We tonen aan dat met de barst het voertuig in de meeste frames onopgemerkt blijft. Bovendien bevat bijna elk frame een vals-positieve detectie. Overeenkomstig presenteren we Fig. 15 als de frames met een persoon die door de scène loopt. We tonen aan dat het af en toe detectie biedt en soms met een verkeerde klasse (surfplank).



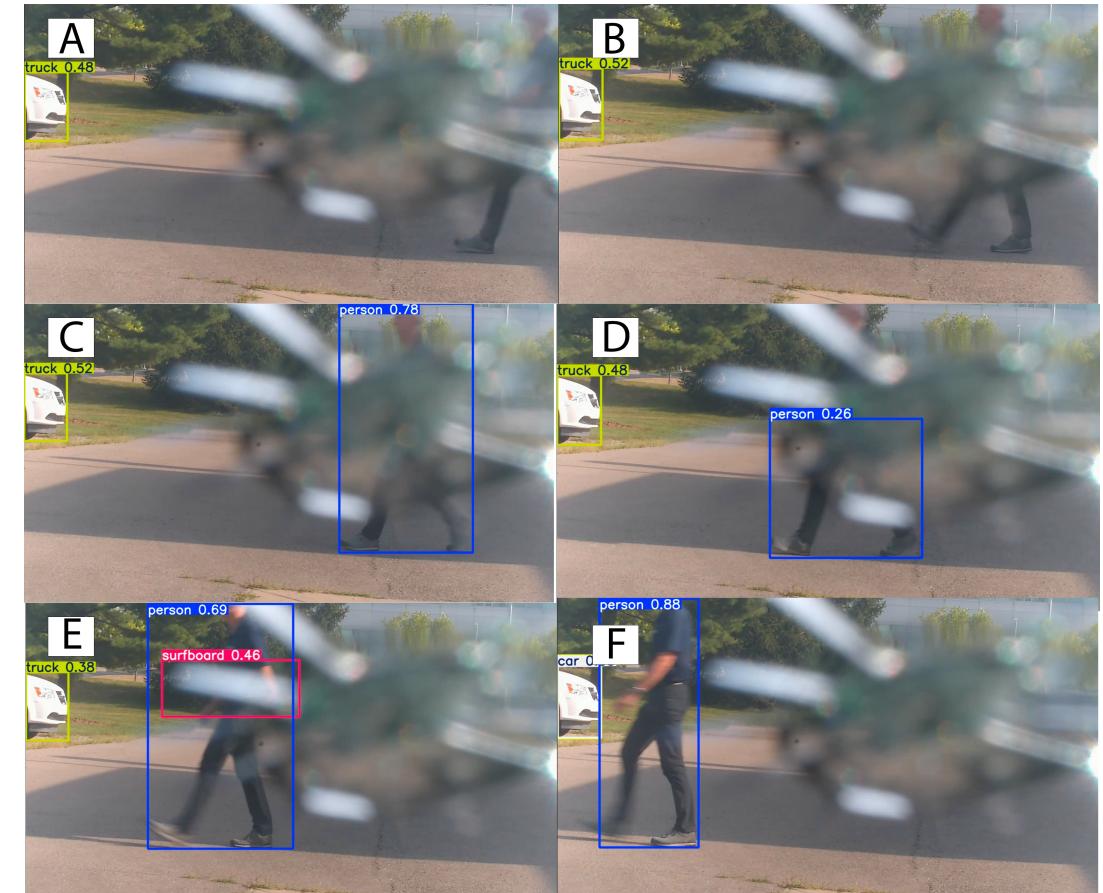
Figure 14: Specific frames of the images taken with the windshield crack with YOLOv8 inference for the vehicle class. A - false positive with no object in scene; B - no inference on vehicle; C - no inference on vehicle; D - first detection on vehicle; E - two different detections on the same vehicle;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



Figuur 14: Specifieke frames van de beelden genomen met de voorruitbarst met YOLOv8-inferentie voor de voertuigklasse. A - vals positief zonder object in de scène; B - geen inferentie op voertuig; C - geen inferentie op voertuig; D - eerste detectie op voertuig; E - twee verschillende detecties op hetzelfde voertuig; F - verkeerde begrenzingsvakgebied.



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



Figuur 15: Specifieke frames van de beelden genomen met de voorruitbarst met YOLOv8-inferentie voor de klasse persoon. A - eerste verschijning van persoon in scène zonder detectie; B - geen inferentie van persoon; C - eerste detectie van persoon; D - gedeelteelijke detectie van persoon; E - detectie van persoon met andere klasse; F - volledige detectie van persoon.

Real glass fracture images

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

Echte glasbreukafbeeldingen

We presenteren een voorbeeld van de glasbreukafbeeldingen verzameld van de FreePik-website, overgelegd op het KITTI Dataset samen met YOLOv8-inferentie (Fig. 16). We tonen aan dat de breuk enkele detecties verwijdert en het vertrouwen in andere detecties verminderd.

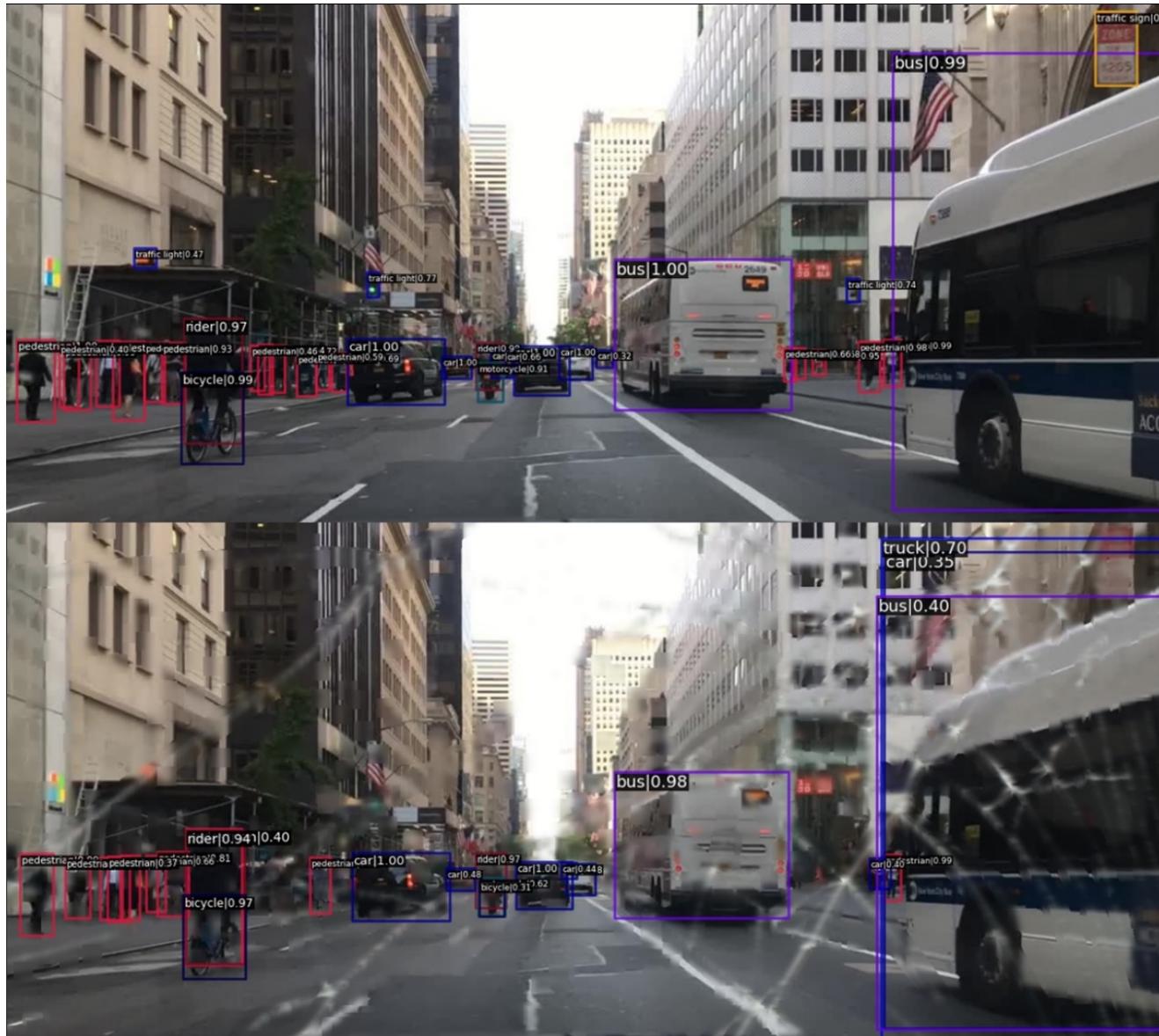
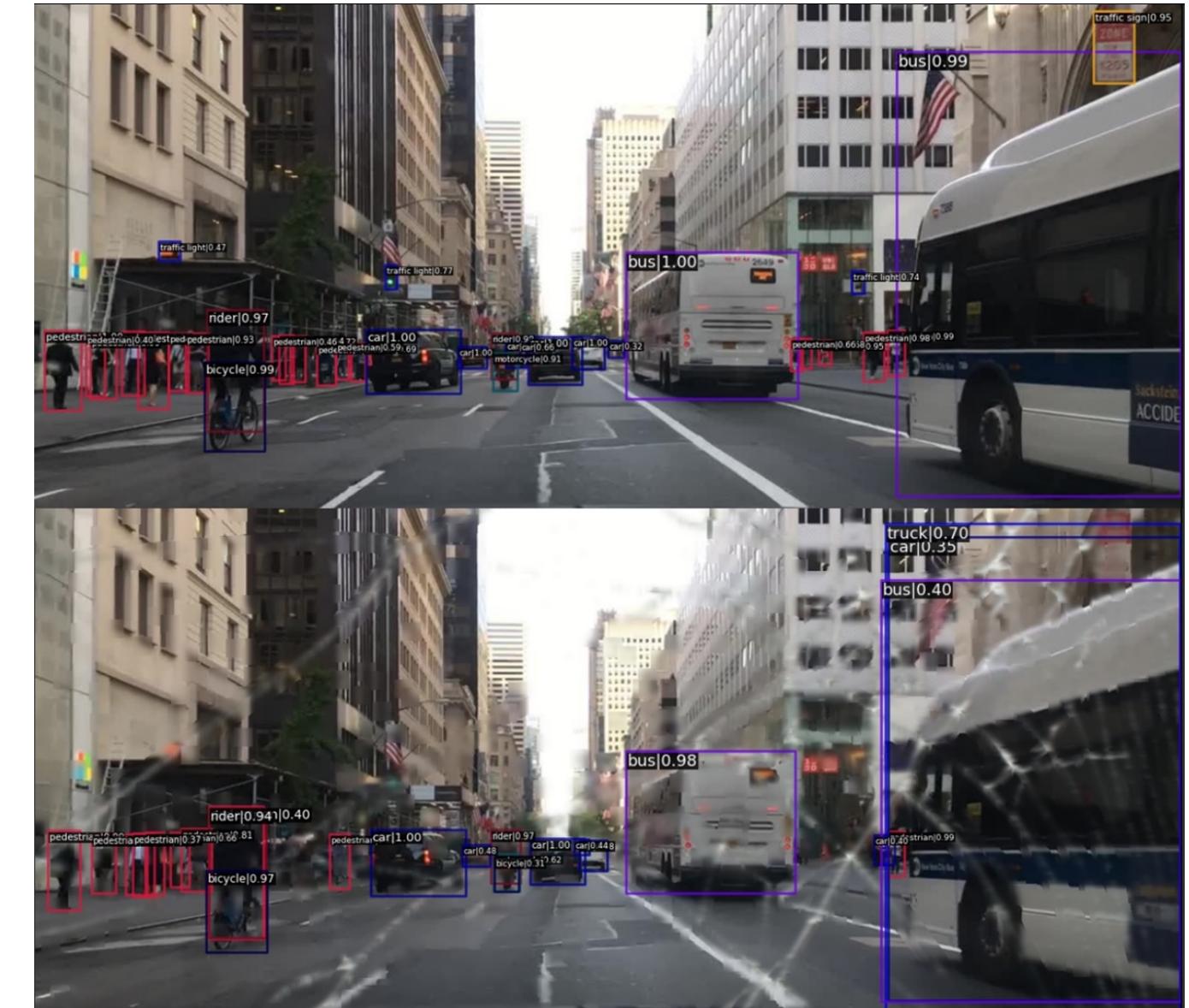


Figure 16: Top - Inference of PTv2 on a clean image from BDD100k. Bottom - Inference for a real broken glass image overlaid on BDD100k for comparison. We see two extra false positives in on the right side (truck, car) and several false negatives for the pedestrian class on the left of the adversarial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



Figuur 16: Boven - Inferentie van PTv2 op een schoon beeld uit BDD100K. Onder - Inferentie voor een echte gebroken glas afbeelding over BDD100K gelegd ter vergelijking. We zien twee extra fout-positieve aan de rechterkant (vrachtwagen, auto) en verschillende fout-negatieve voor de voetgangersklasse aan de linkerkant van de adversariële afbeelding.