

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}](mailto:{prmanav, jwala, kusari}@umich.edu)@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.

Zerbrochenes Glas, versagende Kameras: Simulation physikbasierter adversarialer Muster für autonome Fahrsysteme

Manav Prabhakar, Jwalandhar Girnar, Arpan Kusari*

University of Michigan
Transportation Research Institute 2901 Baxter Road, Raum 202, Ann Arbor,
MI-48103{prmanav, jwala, kusari}@umich.edu

Zusammenfassung

Während sich die Forschung in letzter Zeit stark auf die Erzeugung physikalisch basierter adversarialer Muster konzentriert hat, stammt eine kritische, jedoch oft übersehene Kategorie aus physischen Ausfällen innerhalb der Bordkameras – Komponenten, die für die Wahrnehmungssysteme autonomer Fahrzeuge unerlässlich sind. Kameraausfälle, sei es durch äußere Belastungen, die zu Hardwareausfällen führen, oder durch interne Komponentenfehler, können die Sicherheit und Zuverlässigkeit autonomer Fahrsysteme direkt gefährden. Erstens motivieren wir die Studie durch zwei separate Experimente in der realen Welt, um zu zeigen, dass Glasbrüche tatsächlich dazu führen würden, dass die auf Erkennung basierenden neuronalen Netzmodelle versagen. Zweitens entwickeln wir eine simulationsbasierte Studie, die den physikalischen Prozess des Glasbruchs nutzt, um gestörte Szenarien zu schaffen, die eine realistische Klasse physikalisch basierter adversarialer Muster darstellen. Mit einem auf dem Finite-Elemente-Modell (FEM) basierenden Ansatz erzeugen wir Oberflächenrisse auf dem Kamerabild, indem wir ein Spannungsfeld anwenden, das durch Partikel innerhalb eines Dreiecksnetzes definiert ist. Schließlich verwenden wir physikalisch basierte Rendering-Techniken (PBR), um realistische Visualisierungen dieser physikalisch plausiblen Brüche bereitzustellen. Um die Sicherheitsimplikationen zu bewerten, wenden wir die simulierten Glasbruch-Effekte als Bildfilter auf zwei autonome Fahrdatensätze an - KITTI und BDD100K - sowie auf den groß angelegten Bilddetektionsdatensatz MS-COCO. Anschließend bewerten wir die Erkennungsfehlerquoten für kritische Objektklassen unter Verwendung von CNN-basierten Objekterkennungsmodellen (YOLOv8 und Faster R-CNN) und einer transformerbasierten Architektur mit Pyramid Vision Transformers. Um den verteilungsmäßigen Einfluss dieser visuellen Verzerrungen weiter zu untersuchen, berechnen wir die Kullback-Leibler (K-L)-Divergenz zwischen drei verschiedenen Datenverteilungen, indem wir verschiedene Glasbruchfilter auf einen benutzerdefinierten Datensatz (aufgenommen durch eine gesprungene Windschutzscheibe) sowie auf die KITTI- und Kaggle-Katzen-und-Hunde-Datensätze anwenden. Die K-L-Divergenzanalyse legt nahe, dass diese Glasbruchfilter keine signifikanten verteilungsmäßigen Verschiebungen einführen. Unser Ziel ist es, eine robuste, physikalisch basierte Methodik zur Erzeugung adversarialer Muster bereitzustellen, die reale Kameraausfälle widerspiegeln, mit dem übergeordneten Ziel, die Widerstandsfähigkeit und Sicherheit autonomer Fahrsysteme gegen solche physischen Bedrohungen zu verbessern.

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

Einleitung

Kameras sind allgegenwärtig als Fernsensoren, die Daten aus einer unstrukturierten und dynamischen externen Umgebung sammeln, oft unter rauen Bedingungen. Ein Ausfall oder Fehler in einem Sensor ist eine Abweichung vom funktionalen Zustand in mindestens einem bestimmten Parameter des Systems (van Schrick 1997). Diese Fehler können aufgrund interner (wie Abnutzung) oder externer (Temperatur, Feuchtigkeit usw.) Ursachen auftreten. Bei R GB-Kameras umfassen interne Ursachen tote Pixel, während externe Ursachen gebrochene Gehäuse oder äußere Linsen und Kondensation umfassen. Diese abrupten Ausfälle sind schwer zu erkennen und beeinträchtigen die Objekterkennungsalgorithmen negativ – sie verringern die Genauigkeit und führen oft zu Halluzinationen, wie in Abb. 1 gezeigt. Die in einem automatisierten Fahrzeug (AV) auftretenden Ausfälle können beispielweise zu kritischen Sicherheitsproblemen führen, die zu Unfällen und in einigen Fällen zu Todesfällen führen.

Derzeit gibt es nach bestem Wissen der Autoren keine strengen Methoden zur Erzeugung von kamerabasierten Sensorfehlern (Ceccarelli und Secci 2022).

In dieser Arbeit konzentrieren wir uns auf den Sensorausfall, der durch Risse in einem Glas, das eine Kamera (oder ein Kamera-Gehäuse) bedeckt, verursacht wird, obwohl der in diesem Papier beschriebene Prozess für alle in (Ceccarelli und Secci 2022) aufgeführten Kameraausfälle verwendet werden kann. Diese Glasbruch-Effekte in einer Kamera können durch ein äußeres Objekt, das die Kamera trifft, oder als Folge von plötzlicher Hitze- und/oder Druckentwicklung innerhalb des Gehäuses verursacht werden. Im Jargon der neuronalen Netze wird ein unter solchen Bedingungen aufgenommenes Bild als adversariales Beispiel betrachtet. Frühere Forschungen (Akhtar und Mian 2018; Carlini und Wagner 2017; Szegedy et al. 2013) zeigen, dass selbst kleine Mengen an Störungen, die manchmal schwer mit dem menschlichen Auge zu erkennen sind, ausreichen, um die neuronalen Netze vollständig zu täuschen, wobei eine subtile Änderung der Eingaben zu einer drastischen Änderung der Ausgaben führen kann. Wir möchten darauf hinweisen, dass (Li, Schmidt und Kolter 2019) ein physisches, kamerabasiertes adversariales Angriffsparadigma bereitgestellt haben, das als die am engsten verwandte Arbeit in diesem Bereich dient. Sie präsentierten eine Modifikation des Bildes durch eine Überlagerung eines durchsichtigen, sorgfältig gestalteten Aufklebers, der zu einer Fehlklassifizierung führte.

Um die Auswirkungen dieser Brüche auf die resultierenden Kamerabilder zu verstehen, führen wir zwei unterschiedliche Experimente durch: eines

Korrespondierender Autor Copyright © 2026, Association for the Advancement of Artificial Intelligence(www.aaai.org). Alle Rechte vorbehalten.

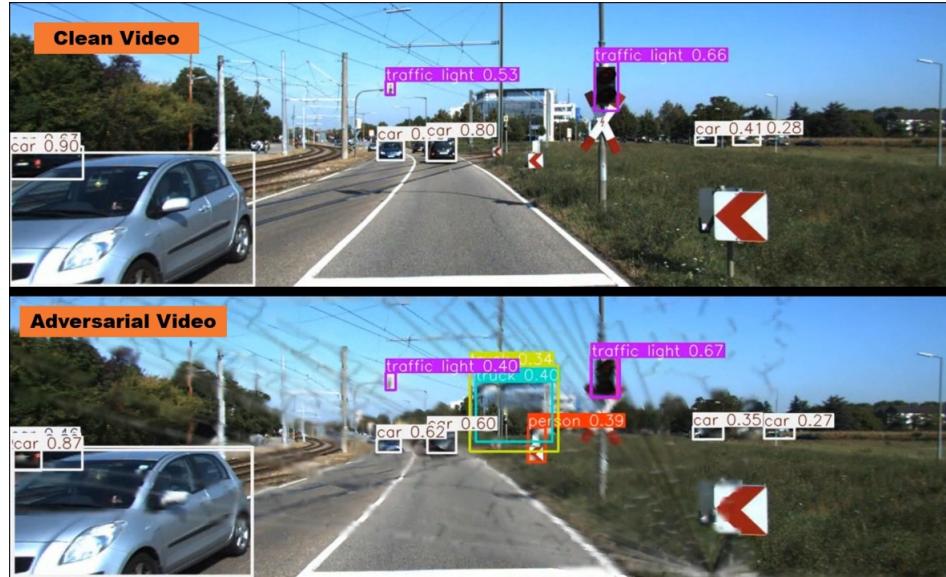


Figure 1: A qualitative comparison of clean vs adversarial video generated using our simulation and rendering method on KITTI. This frame shows false positives, and reduced confidence levels for true positives. Refer to the supplementary material for full video.

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

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

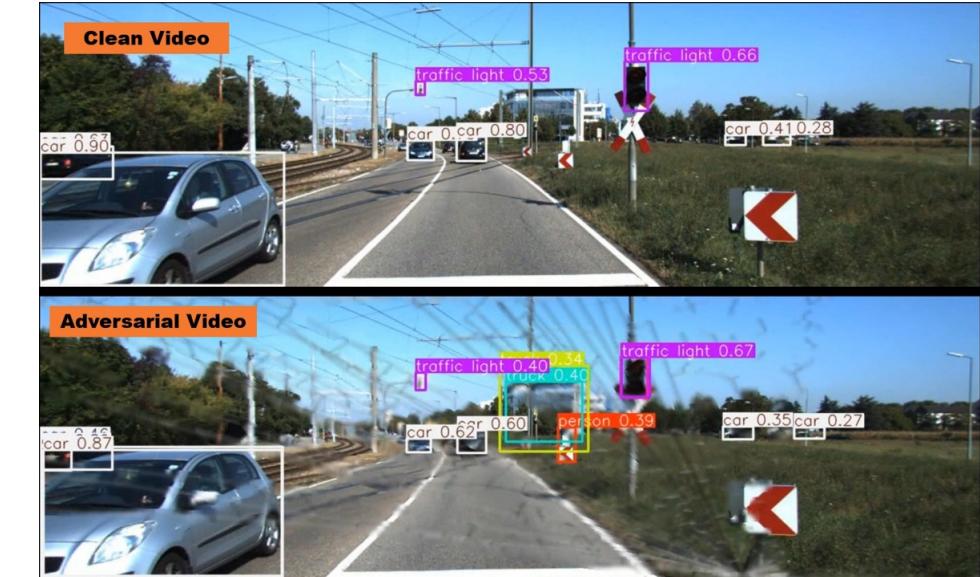


Abbildung 1: Ein qualitativer Vergleich von sauberen vs. adversarialen Videos, die mit unserer Simulations- und Rendering-Methode auf KITTI erzeugt wurden. Dieses Bild zeigt Fehlalarme und verringerte Vertrauensniveaus für echte Positive. Siehe das Ergänzungsmaterial für das vollständige Video.

in einer statischen Innenumgebung und das andere in einer dynamischen Außenumgebung. Das erste Experiment beinhaltet das Zerbrechen von gehärtetem Glas und das Platzieren vor der Kamera (siehe Abb. 2(a)) mit einem statischen Fahrzeug in der Szene, um zu verstehen, wie verschiedene Bruchmuster die Qualität und das Erscheinungsbild der Szene beeinflussen. Wir erfassen Bilder bei unterschiedlichen Brennweiten, um die Variabilität solcher Störungen zu beurteilen. Dies half uns, bestimmte qualitative Fragen über das visuelle Erscheinungsbild dieser Brüche in Bezug auf ihre Ausbreitung und Intensität zu beantworten, was unseren Ansatz in den Abschnitten Fokalebene und Physische Angriffssimulation motivierte. Der experimentelle Aufbau und die detaillierten experimentellen Ergebnisse sind in Abschnitt Statisches Experiment der Ergänzung zu finden. Das zweite Experiment (Abb. 2(b)) bestand darin, ein Outdoor-Video mit dynamischen Fahrzeugen bei Tageslichtbedingungen aufzunehmen, indem eine MobileEye-Kamera neben einem in Abb. 2 dargestellten Windschutzscheibenrisse (oben links gezeigt) platziert wurde und eine Inferenz mit YOLOv8 (Jocher, Chaurasia, und Qiu 2023) durchgeführt wurde, um ein grundlegendes Verständnis der Auswirkungen solcher Szenarien auf Objekterkennungsnetzwerke zu gewinnen. Wir beobachteten, dass das Modell das Fahrzeug in einem sauberen Bild leicht erkennen kann, während es bei der Erkennung versagt (unten rechts) oder falsche Positive erzeugt (unten links). Interessanterweise kann das Vorhandensein eines Risses auch unerwartet das Vertrauen in die Vorhersage des Autos mit einer klar definierten Kante erhöhen (0,92 unten links vs. 0,75 oben links). Die detaillierten Inferenz-Ergebnisse mit Fahrzeug- und Person-Klasse sind in Abschnitt Dynamisches Experiment der Ergänzung angegeben.

Wir suchten dann online nach echten Bildern von zerbrochenem Glas (Abschnitt Echte Glasbruchbilder der Ergänzung), konnten jedoch keinen ausreichend großen Datensatz erstellen, um einen datengetriebenen Ansatz für die Abwehr von Angriffen unter diesen Bedingungen zu ermöglichen. Zusätzlich experimentierten wir mit CGI-Tools wie Maya und Blender,

um solche Effekte zu erzeugen, aber sie fehlen an Flexibilität, Kontrolle, Maßstab und Physik, um diese Bedingungen zu simulieren. Die nächstliegende Simulationsoption in der vorhandenen Literatur ist ArcSim (Pfaff et al. 2014). Allerdings sind ihre hochauflösenden Simulationsausgaben extrem langsam (≈ 20 Stunden), was es schwierig macht, sie zu skalieren. Daher richten wir unsere Bemühungen darauf, eine skalierbare, simulationsbasierte Pipeline zu schaffen, um Brüche zu erzeugen, die zur Weiterentwicklung des Wahrnehmungstyps verwendet werden können.

Bei einem Glasbruch können der Hauptpunkt, die Kraft und der Einfallswinkel zufällig sein, aber die Ausbreitung und das resultierende Muster folgen einem inhomogenen physikalischen Prozess (entweder linear oder radial). Wir erstellen daher eine Bruchsimulation basierend auf Partikeln in einem zufällig erzeugten Dreiecksnetz und führen die Spannungsverteilung durch das Netz durch. Unsere Simulation ermöglicht es uns, die Brüche innerhalb eines Dreiecksnetzes zu jedem diskreten Zeitpunkt δt zu erzeugen. Wir verwenden OpenCV, um das gegebene Netz in ein entsprechendes zerbrochenes Glas-Musterbild umzuwandeln. Anschließend nutzen wir physikalisch basiertes Rendering (PBR) (Pharr, Jakob, und Humphreys 2023), um die Oberflächenbrüche realistisch darzustellen, indem wir die bidirektionale Reflexionsverteilungsfunktion (BRDF) verwenden, um die Menge des von einem bestimmten Punkt auf einer Oberfläche reflektierten Lichts zu berechnen, die durch einfallende Lichtquellen verursacht wird.

Indem wir unseren Rendering-Ansatz mit drei beliebten Open-Source-Datensätzen kombinieren - KITTI (Geiger et al. 2013), BDD100K (Yu et al. 2020) und MS-COCO (Lin et al. 2014), sind wir in der Lage, effizient adversarische Bilder zu generieren. Ein gängiger Prozess zum Testen der generierten adversarischen Bilder besteht darin, die Anzahl der falsch-positiven/negativen über den Bildraum zu ermitteln. In unserem Fall können wir jedoch aufgrund des lokalen adversarischen Effekts nicht einfach auf eine bildbasierte Messung vertrauen. Daher verwenden wir die adversarischen Bilder (ähnlich der unteren linken Abbildung in Abb. 2) und extrahieren die Objekte

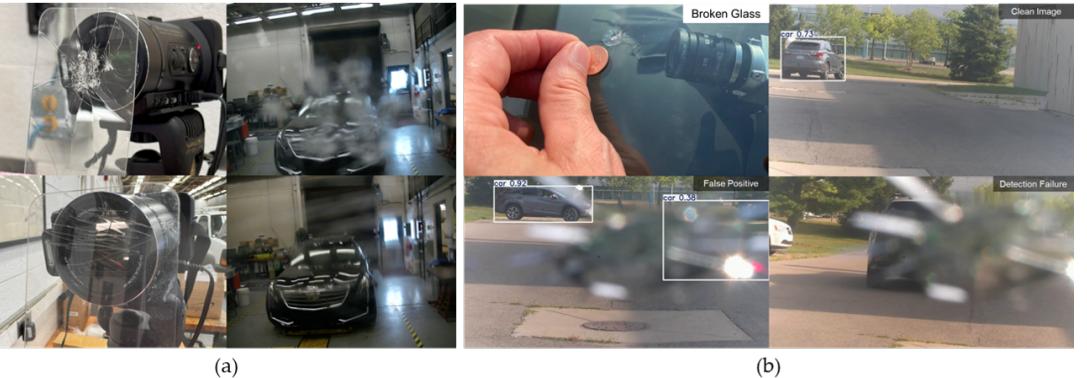


Figure 2: (a) Indoor static experiment. Left: Camera with 2 different fractured tempered glass patterns; right - images of the vehicle under the different fractures. (b) Outdoor dynamic experiment. Top left - a coin sized windshield crack; top right - clean image with the vehicle detected using YOLOv8; 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

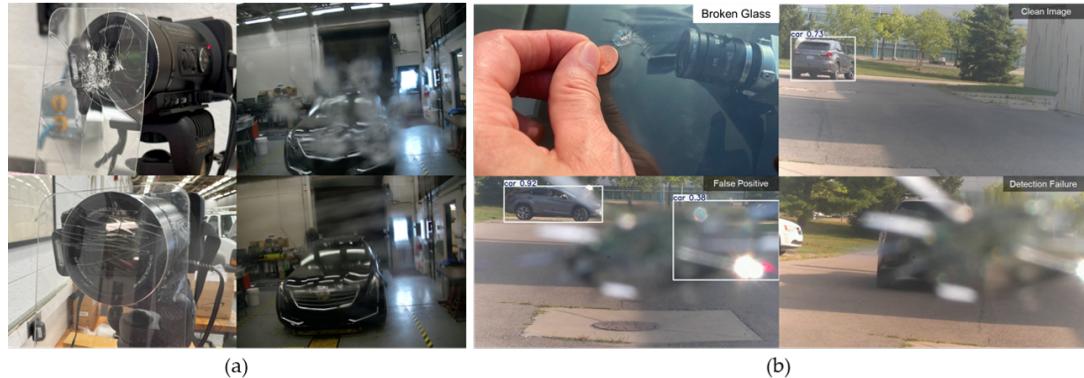


Abbildung 2: (a) Statisches Experiment im Innenbereich. Links: Kamera mit 2 verschiedenen Mustern von gebrochenem Sicherheitsglas; rechts - Bilder des Fahrzeugs unter den verschiedenen Brüchen. (b) Dynamisches Experiment im Außenbereich. Oben links - ein münzgroßer Riss in der Windschutzscheibe; oben rechts - sauberes Bild mit dem Fahrzeug, das mit YOLOv8 erkannt wurde; unten links - falsch positives Ergebnis durch den Riss; unten rechts - Erkennungsfehler durch das Glas. Weitere Beispiele aus diesen Experimenten sind im Ergänzungsmaterial enthalten.

die sich in dem Bereich befinden, in dem der Bruch existiert, unter Verwendung der Ground-Truth-Bounding-Boxen. Wir nutzen dann YOLOv8, Faster R-CNN (Ren et al. 2016) und Pyramid Vision Transformer (PVTv2) (Wang et al. 2022), um den Prozentsatz der Objekte zu ermitteln, die fehlschlagen, wenn die adversarialen Filter angewendet werden. Wir bieten auch Ablationsstudien an, um die verteilungsbedingten Unterschiede zwischen den drei Bildersätzen zu verstehen: Experimentell gesammelte echte zerbrochene Glasbilder, online gesammelte echte zerbrochene Glasbilder und die generierten Bilder. Wir berechnen die Kullbeck-Liebler-Divergenz für diese Bildverteilungen, um die Ähnlichkeit der generierten Bilder mit den echten zerbrochenen Glasbildern zu beweisen. Wir verwenden Katzenbilder aus dem Kaggle Cats and Dogs Datensatz als Kontrolle, um den Unterschied zwischen Bildverteilungen (PK) zu verstehen.

Die wesentlichen Beiträge des Papiers lassen sich wie folgt zusammenfassen:

- Wir bieten eine neuartige Methode zur Abstraktion von Glasbrüchen durch eine Kombination von Spannungsverbreitungsmethoden und minimalen Spannbäumen, um physikalisch fundierte gebrochene Glasstrukturen zu erzeugen.
- Wir präsentieren einen PBR-Ansatz, um ein realistisches Rendering von Kameraausfällen zu ermöglichen, das mit jeder Art von bestehenden Computer-Vision-Datensätzen - sowohl Bilder als auch Videos - verwendet werden kann.
- Unsere Simulations- und Rendering-Pipelines sind skalierbar und recheneffizient ($\approx 1.6s$), sodass sie sowohl von der Wissenschaft als auch von der Industrie zur Verbesserung der Robustheit und des Schutzes vor Verteilungen für eine Vielzahl von Anwendungen genutzt werden können.

Hintergrund

Physikbasierte adversarielle Muster

Das Problem des adversarialen Beispiels kann wie folgt definiert werden: Für ein Modell M , das eine Eingabeprobe X korrekt in ihre vorgesehene Klasse klassifiziert, d.h. $M(X) = y_{true}$, führt das Hinzufügen eines Fehlers ϵ zur Eingabeprobe X zu einer veränderten Probe \hat{X} , sodass $M(\hat{X}) \neq y_{true}$. Somit führt die Injektion des Fehlers ϵ zu einem adversarialen Beispiel, das das Modell zum Scheitern bringt.

Obwohl die Idee der adversarialen Manipulation des Modells im Kontext des maschinellen Lernens schon vor einiger Zeit identifiziert wurde (Dalvi et al. 2004), lag der Fokus im letzten Jahrzehnt eindeutig auf den adversarialen Angriffen auf neuronale Netzwerke (Szegedy et al. 2013; Goodfellow, Shlens und Szegedy 2014). In diesen Arbeiten zeigten die Forscher, dass eine kleine gezielte Injektion von Rauschen, die für das menschliche Auge fast nicht wahrnehmbar ist, die Labels vollständig veränderte (Szegedy et al. 2013) und umgekehrt Bilder erzeugt werden konnten, die für Menschen völlig unkenntlich aussahen, aber von den DNNs perfekt klassifiziert wurden (Nguyen, Yosinski und Clune 2015).

Während diese adversariellen Muster das Modell auf mögliche Fehler untersuchen, fehlt ihnen jeglicher physikalischer Realismus bei ihrer Erstellung und sie benötigen Zugriff auf das Modell. Um dies zu adressieren, hat sich die jüngste Forschung darauf konzentriert, physikalisch relevante adversarielle Muster zu entwickeln. Einer der ersten Vorstöße in diesem Bereich wurde von (Kurakin, Goodfellow und Bengio 2018) unternommen, die die Genauigkeit der Modelle in der physischen Welt ins Visier nahmen, indem sie verrauschte Bilder von einer Handy-Kamera einspeisten, die das Modell dazu brachten, einen großen Teil der Muster falsch zu klassifizieren. In ähnlicher Weise demonstrierten (Eykholt et al. 2018), dass echte Verkehrsschilder mit einfachen, strategisch platzierten physischen Aufklebern so gestört werden können, dass sie selbst bei Blickwinkeländerungen die modernsten DL-Algorithmen nahezu perfekt täuschen. Andere Forscher haben adversarielle Bilder (Kong et al. 2020), Translucent Patches auf der Kamera (Zolfi et al. 2021) oder künstliche LiDAR-Oberflächen (Tu et al. 2020) platziert, um Muster zu erzeugen, die Objektdetektoren täuschen. Während diese vorherige Forschung Physik zur Erzeugung der Muster verwendet, stammen sie nicht aus der Modellierung eines rigorosen physikalischen Prozesses, und wir zielen darauf ab, diese Lücke in dieser Arbeit zu schließen.

Theorie des gebrochenen/zerbrochenen Glases

Das Thema, wie Glas bricht und wie es sich ausbreitet, ist nach wie vor eine offene Forschungsfrage, die mit mehreren vorgeschlagenen physikalischen Theorien umstritten ist (Rouxel und Brow 2012). Während das mikroskopische Verfahren des Glasbruchs diskutiert wird, ist auf makroskopischer Ebene das Rissverhalten

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 $\vec{p_I p_j}$ where $p_j \in k_i$ as

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

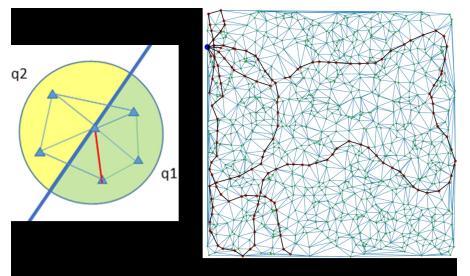


Figure 3: (a) For a splitting plane given in blue, the summed positive stress q_1 and summed negative stress q_2 are compared and the propagation happens on the side with greater summed stress and chosen node which is the closest to the splitting plane (given in red). (b) Shows how we simulate a fracture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

gut verstanden. (Liu et al. 2021) analysierten den Prozess des Glaslinsenbruchs in der Präzisionsglasformungsanwendung unter Verwendung von FEM mit einem dreidimensionalen Modell in einer physikalischen Simulationssoftware. Die physikalischen Parameter wurden in die Software eingegeben und die Risspfade anhand der Simulationsergebnisse analysiert. Die Autoren führten eine Temperatur- und Spannungssimulation eines hochpräzisen dreidimensionalen Netzmodells des geformten Glases durch. (Iben und O'Brien 2009) boten eine Möglichkeit, Oberflächenbrüche in verschiedenen Materialien, einschließlich Glas, zu erzeugen. Wie bereits in der Einleitung erwähnt, bot (Pfaff et al. 2014) die Simulation des Glasbruchs als dünnes Blatt, was die am engsten verwandte Arbeit zu unserer vorgeschlagenen Methode darstellt.

Methodik

Um realistische Glasbrüche zu erzeugen, müssen groß angelegte physikbasierte Simulationen durchgeführt werden, indem die Bruchdynamik auf einem triangulierten Finite-Elemente-Netz mit Glaseigenschaften gelöst wird.

Simulation von gebrochenem Glas

Wir stellen Glas dar, indem wir Partikel verwenden, die aus einer gleichmäßigen Verteilung über eine Ebene entnommen werden, die in Form eines 2D-Netzes mit eingeschränkter Delaunay-Triangulation eingeschränkt ist. Dies entfernt schlecht geformte Dreiecke und vermeidet ungleichmäßige und unrealistische Kanten.

Jedes Partikel p_i hat eine Position x_i und hat nächste Nachbarn k_i innerhalb eines Radius r , die bestehende Kanten mit p_i haben. Mathematisch stellt das Triangulationsnetz \mathcal{M} eine endliche Menge von 2-Simplizes dar, so dass, wenn

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

Die Rissmuster im Glas entstehen durch die Spannung der äußeren Kraft (F) am anfänglichen Einschlagpunkt p_I , indem ein spezifisches Deformationsgesetz (Elastizität und Plastizität) des Glases (G) angenommen wird (Kuna 2013). Wir berechnen dann die Festigkeitsparameter in Form von effektiver Spannung σ_V am Einschlagpunkt (V) als Spannungszustand des Einschlagpunkts. Die kritischen Spannungswerte für die Festigkeit des Glases σ_C werden durch Tests an einfachen Proben mit elementaren Belastungsbedingungen (z.B. Zugversuch) ermittelt. Der Bruch tritt dann auf, wenn die effektive Spannung größer ist als die kritische Spannung, geteilt durch den Sicherheitsfaktor (S):

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

Aus der klassischen Festigkeitstheorie wissen wir, dass das Versagen in den meisten Fällen durch die Hauptspannungen σ_I und σ_{II} für 2D-Elemente kontrolliert wird. Der anfängliche Riss entsteht entweder durch den normal-planaren Riss, bei dem die Bruchflächen senkrecht zur Richtung der höchsten Hauptspannung σ_I (Rankine 1857) liegen, oder durch den Scherplanar-Riss, bei dem die Bruchflächen mit den Schnittflächen der maximalen Scherspannung $\tau_{max} = (\sigma_I - \sigma_{II})/2$ (Coulumb 1776) zusammenfallen. Im Fall von Glas nehmen wir an, dass der anfängliche Bruch senkrecht zur Richtung der maximalen Hauptspannung erfolgt.

Vom anfänglichen Einschlagpunkt p_I aus ist die Spannungsverbreitung durch das Glas instabil, da der Riss abrupt wächst, ohne dass eine Erhöhung der äußeren Belastung erforderlich ist. Ab p_I ,

breite sich die Spannung in der Nachbarschaft des Scheitelpunkts k_i aus, da die Spannung entlang der Richtung $\vec{p_I p_j}$ verläuft, wo $p_j \in k_i$ als

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

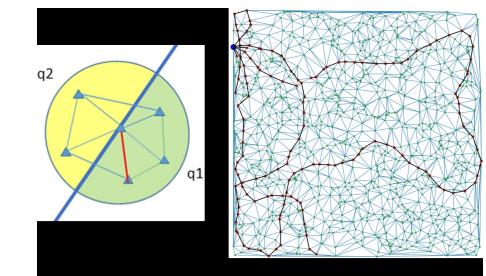


Abbildung 3: (a) Für eine in Blau gegebene Spaltebene werden die aufsummierte positive Spannung q_1 und die aufsummierte negative Spannung q_2 verglichen, und die Ausbreitung erfolgt auf der Seite mit der größeren aufsummierten Spannung und dem gewählten Knoten, der der Spaltebene am nächsten liegt (in Rot angegeben). (b) Zeigt, wie wir einen Bruch in einem Netz simulieren, der von seinem Einschlagpunkt (in Blau markiert) zu den Knoten ausgeht, die Spannungen über ihrer Schwellenfestigkeit erfahren (in Rot markiert).

Mit dem für jede Kante berechneten Stress kann der aufsummierte positive Stress (in Abb. 3 gezeigt) dann wie folgt angegeben werden:

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

für die kontinuierliche Oberfläche Ω , wobei die Indikatorfunktion ist. Der aufsummierte positive Stress für diskrete Simplizes im entsprechenden Bereich A mit Radius R wird wie folgt angegeben

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

Ähnlich wird der aufsummierte negative Stress q_2 berechnet. Dann wählen wir für die größere Größe $\max(|q_1|, |q_2|)$ die entsprechende Kante mit der höchsten Stresskonzentration im gegebenen Segment als die optimale Trennebene, da dies die maximale Stressentlastung bietet. Somit wandert der Stress entlang der Maschenkanten und dissipiert den Stress an jedem Knotenpunkt.

Die rekursive Anwendung der Stressausbreitung wird durchgeführt, bis die Konvergenz des Stresses in allen Zuständen erreicht ist, d.h. $\sigma_p^{(t)} \approx \sigma_p^{(t-1)} \forall p \in V$.

Die Ausbreitung des Stresses in alle Richtungen über alle Knoten führt zu RückrisSEN, wie in (O'Brien und Hodgins 1999) erklärt. Um dies zu vermeiden, propagieren wir nur entlang der Kanten, wo die Stressniveaus maximal sind, führen jedoch ein Stress-Update an allen benachbarten Knoten durch. Wir verwenden dann einen minimalen Spannbaum (MST) auf einem Netz, das mit diesen gestressten Knoten erstellt wurde. Wir kombinieren diesen MST mit unserem anfänglichen Stressausbreitungsfeld entlang der Kanten, um das endgültige Rissmuster zu berechnen. Der MST ist eine effektive Abstraktion, da er die Knoten verbindet, die näher beieinander liegen und sich im Hochstressfeld befinden, während Redundanzen entfernt werden.

Unser Berechnungsprozess der Spannungsverteilung ist in Algorithmus 1 in der Ergänzung definiert.

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,

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

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

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

Then, L_r can be defined as

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

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

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

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

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

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

where \hat{n}_i is the unit surface normal of the crack.

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

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

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,

Physikalisch basiertes Rendering

Sobald wir die Brüche auf der Mesh-Ebene erzeugt haben, besteht unser nächstes Ziel darin, eine visuelle Darstellung dieser Brüche zu erstellen. Wie alle PBR-Techniken basiert unsere Methode auf der Mikrofacet-Theorie, die besagt, dass jede Oberfläche durch winzige, perfekt reflektierende Spiegel, sogenannte Mikrofacetten, beschrieben werden kann (Pharr, Jakob, and Humphreys 2023).

In Übereinstimmung mit der Mikrofacet-Theorie und der Energieerhaltung verwenden wir die Reflexionsgleichung,

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

wobei $L_o(x, \omega_o, \lambda, t)$ die gesamte spektrale Strahldichte der Wellenlänge λ ist, die entlang der Richtung ω_o zum Zeitpunkt t von einer bestimmten Position x nach außen gerichtet ist. ω_o ist die Richtung des ausgehenden Lichts. t ist die Zeit. L_e ist die emittierte spektrale Strahldichte und L_r ist die reflektierte spektrale Strahldichte.

Sei I_1 die bidirektionale Reflexionsverteilungsfunktion,

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

und sei I_2 die spektrale Strahldichte, die von Richtung ω_i zum Zeitpunkt t nach innen zu x kommt.

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

Dann kann L_r definiert werden als

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

wobei Ω die um die Oberflächennormale \mathbf{n} zentrierte Einheits-Halbkugel über ω_i ist, sodass $\omega_i \cdot n > 0$.

Durch die Abstraktion der Reflexionsgleichung wollen wir ein visuelles Rendering unseres gebrochenen Glasgitters erstellen. Wir haben $L_e = 0$, da Glas kein Licht emittiert. Nun, um L_r zu berechnen, betrachten wir jeden Riss zwischen den Knoten als Mikrofacet. Dann können wir L_r für jeden Riss definieren als:

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

Gegeben die Einheitsvektoren $(\hat{\omega}_\alpha)$ und $(\hat{\omega}_\theta)$, die den Azimut- (α) und Zenit- (θ) Winkel entsprechen, berechnen wir die mittlere Energie, die auf den Riss trifft, als

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

wobei \hat{n}_i die Einheits-Oberflächennormale des Risses ist.

Sei (I_r, I_g, I_b) die mittlere Intensität der Lichtquelle. Dann wird die Rissintensität, I_c , definiert als

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

Fokalebene und Physische Angriffssimulation Während wir in der Lage sind, realistische Brüche zu simulieren, besteht der Hauptanwendungsfall unserer Arbeit darin, simulierte Beispiele zu erzeugen, die auf bestehenden Datensätzen (KITTI, BDD100K, MS-COCO) überlagert sind, und sie mit dem realen Straßendatensatz zu vergleichen, den wir erstellt haben.

Jedes aufgenommene Bild wird scharfe Merkmale der Objekte in seiner Fokalebene aufweisen. Das Glasgehäuse, das die Kamera abdeckt, ist extrem nah und gehört daher nicht zur Fokalebene.

Wenn der Riss entsteht, prallen die Lichtstrahlen ungleichmäßig entlang des Risses ab und erzeugen eine Unschärfe (Beispiel in Abb. 4). Wir erstellen eine Binärmaske basierend auf dem Rissmuster und verwischen dann die überlagerten Brüche auf dem Bild. Dies erzeugt ein Bild mit Fernfokus. Für ein Bild mit Nahfokus verwischen wir das Bild und konzentrieren uns auf den Vordergrund, d.h. den Riss.

Experimentieren

Datensatz

Wir bewerten zwei Arten von zerbrochenem Glas - real und simuliert - auf drei beliebten Open-Source-Datensätzen: KITTI (Geiger et al. 2013), BDD100K (Yu et al. 2020) und MS-COCO (Lin et al. 2014). Die ersten beiden repräsentieren spezifische Domänen des autonomen Fahrens, während der letzte ein allgemeiner Bilddatensatz ist. Die Bilder mit realen zerbrochenen Glasstrukturen werden von der FreePik-Website¹ gesammelt und stellen in unserem Fall die Basislinie dar. Insgesamt haben wir 65 Bilder gesammelt und sie durch Bildaugmentation mit zufälligen Verschiebungen, Bildspiegelungen und Zuschneideoperationen auf einen Satz von 10,000 Bildern erweitert. Wir erzeugen auch 10,000 Bilder mit unserem Physiksimulator. Anschließend überlagern wir diese zerbrochenen Glasstrukturen mit unserer PBR-Pipeline auf jedes Validierungsbild in den Datensätzen und sammeln die aggregierten Ergebnisse. Wir verwenden drei Modellarchitekturen: YOLOv8, Faster R-CNN und PVTv2-Modell mit vorgeübten Gewichten, um Objekterkennungsergebnisse zu erzeugen.

Implementierung

Unser Simulationsmodell wird entwickelt, indem 10^4 Partikel zufällig aus einer gleichmäßigen räumlichen Verteilung im gegebenen Rahmen auf einer CPU entnommen werden. Ein KD-Baum aus dem SciPy-Python-Paket (Virtanen et al. 2020) mit Standardparametern wird konstruiert, um die ungefähren nächsten Nachbarn jedes Partikels zu finden. Anschließend wird eine Delaunay-Triangulation auf den Partikeln durchgeführt, um ein eingeschränktes dreieckiges Netz zu erstellen. Wir verwenden eine Einschlagkraft von 500 Einheiten mit einem zufälligen Einschlagspunkt und einem zufälligen Einschlagsvektor. Die Spannungsverbreitung erfolgt, bis ein Schwellenwert von 300 Einheiten erreicht ist. Das PBR wird auf der CPU durchgeführt, indem die im vorherigen Abschnitt beschriebenen Methoden unter Verwendung von OpenCV und Python implementiert werden.

Ergebnisse und Diskussion

Ein wesentlicher Unterschied zu den meisten früheren Arbeiten zu adversarialen Beispielen besteht darin, dass unsere generierten adversarialen Muster nicht alle Pixel in einem Bild universell beeinflussen. Daher muss der Vergleich nur für den Bildbereich durchgeführt werden, in dem das Muster existiert. Zu diesem Zweck erstellen wir eine Binärmaske jedes Musters und geben die Ergebnisse der Objekte aus, die nur in diesem Muster existieren.

Tabelle 1 zeigt die Ergebnisse der durchschnittlichen Präzision (AP) unter den adversarialen Bildern, die mit den beiden Arten von Rissmustern (online gesammelt und simuliert) für verschiedene Klassen erzeugt wurden. Für KITTI sinkt die AP der anderen Klassen erwartungsgemäß mit dem Rückgang der AP, der dem Prozentsatz der vom Lkw besetzten Bildfläche entspricht, wobei der Lkw die größte Abnahme verzeichnet. Für BDD100K mit PVTv2-B0 sehen wir, dass der Rückgang der AP bei den simulierten Bildern am größten ist, aber insgesamt,

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-

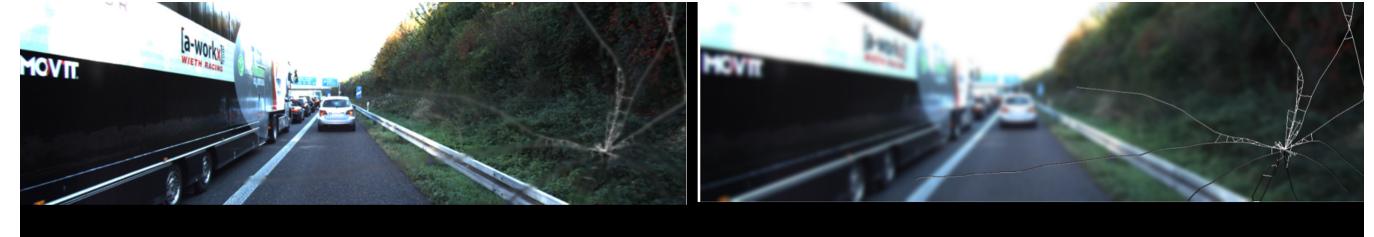


Abbildung 4: (a) Zeigt das simulierte Bild mit der Straße und den Fahrzeugen in der Fokalebene (PBR und Fernfokus). (b) Bezeichnet das simulierte Rissmuster in der Fokalebene (PBR und Nahfokus).

Tabelle 1: Durchschnittliche Präzision (in Prozent) verschiedener Klassen in KITTI, BDD100k und MS-COCO unter verschiedenen adversarialen Bildern. x gibt die Überlagerungsbeziehung zwischen Datensatz und Glasriss-Typ an. Sauber x Datensatz - bezieht sich direkt auf die speziellen Bilder ohne adversarische Probe. RO x Datensatz - bezieht sich auf echte Bilder von online gesammeltem, rissigem Glas, die auf saubere Bilder überlagert wurden. Sim x Datensatz - bezieht sich auf simulierte Rissmuster, die auf saubere Bilder überlagert wurden.

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

Der Trend wird beibehalten, wobei die Fußgängerklasse den stärksten Rückgang zeigt. Für MS-COCO haben wir den AP für die Superkategorien aggregiert: Person, Fahrzeuge und Essen. Dies liegt daran, dass viele Objekte in MS-COCO eine kleinere Fläche im Bildrahmen einnehmen, was es schwierig macht, aussagekräftige Ergebnisse aus allen Kategorien zu erhalten. Ein sehr interessantes Ergebnis ist, dass die Fußgängerklasse unter den realen gebrochenen Glasmustern einen mehrfachen Anstieg des AP aufweist. Während dieser Trend kontraintuitiv erscheinen mag, stimmt er mit den Ergebnissen in Abb. 2 überein, wo das Vertrauen des Autos aufgrund einer Kante steigt. Dies zeigt tatsächlich, dass der AP stark vom Rissmuster abhängt, was es äußerst wichtig macht, Verteidigungsmethoden zu entwickeln, um diese adversarialen Angriffe zu mildern.

Ablationsstudien

Unsere Ergebnisse zeigen, dass die simulierten Bilder einen ähnlichen adversarialen Effekt wie die realen Bilder erzielen. Daher ist eine wichtige Ablationsstudie für uns zu verstehen, wie nah die simulierten Rissmuster an den realen Rissmustern und den online gesammelten Mustern sind. Wir bilden 5 Verteilungen

- Realer On-Road-Datensatz (dargestellt in Abb. 2)

- Rissmuster, die online gesammelt wurden (Abb. 5 oben links)
- Simulierte Rissmuster (Abb. 5 unten links)
- Simulierte Rissmuster überlagert auf KITTI (Abb. 5 unten rechts)
- Rissmuster, die online gesammelt wurden, überlagert auf KITTI (Abb. 5 oben rechts)

Wir berechnen nun die K-L-Divergenz zwischen all diesen Verteilungen, um zu ermitteln, wie ähnlich sie einander sind (siehe Abb. 6). Um eine Kontrolle zu bieten, vergleichen wir KITTI mit Bildern von Katzen aus dem Kaggle-Datensatz, was eine K-L-Divergenz von 2,434 ergibt. In dieser Skala haben die PBR-Bilder von zerbrochenem Glas einen Unterschied von 0,36 zu den echten zerbrochenen Glasmustern, während die zerbrochenen Glasfilter, die auf KITTI-Bilder überlagert sind, eine ähnliche K-L-Divergenz aufweisen.

Abb. 7 zeigt eine Analyse der Rechenzeit für jedes unserer Module und über verschiedene Anzahlen von Partikeln. Wir führen diese Analyse über 100 Durchläufe durch, wobei wir zufällige Einschlagpunkte, Einschlagswinkel und Netzstrukturen mit einer festen Anzahl von Partikeln generieren. Der Unterschied in der Rechenzeit für verschiedene Durchläufe kann dem Einschlagpunkt und dem Ein-

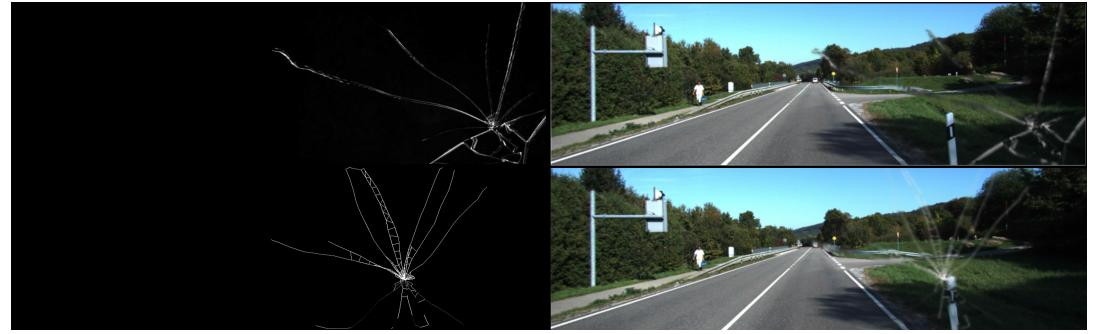


Figure 5: Top left - Crack pattern collected online on Freepik; top right - online crack pattern overlayed on KITTI; bottom left - simulated crack pattern with PBR; bottom right - simulated crack pattern overlayed on KITTI.

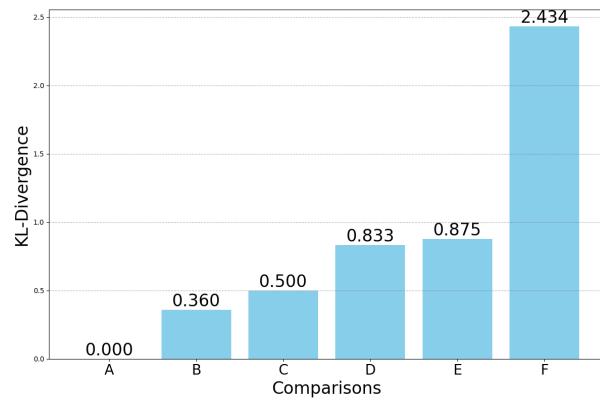


Figure 6: K-L divergence of different pairs of image distributions. Datasets: RC - Real on-road dataset (see Fig. 2), KITTI and Cats. Filters: RO - Real (collected online) and Sim - Simulated. K-L divergence between (x - overlay relation): A - (Sim x KITTI) vs (Sim x KITTI); B - (Sim vs RO); C - (Clean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

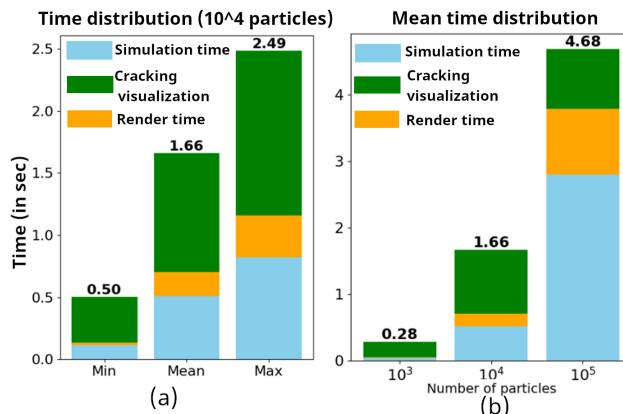


Figure 7: (a) Mean time taken by different modules of our pipeline across 100 runs. (b) The minimum, maximum and mean time taken by different modules across 100 runs for a 10^4 particle mesh. For these plots, we showcase the time taken for simulation (simulation time), converting the mesh to glass (cracking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

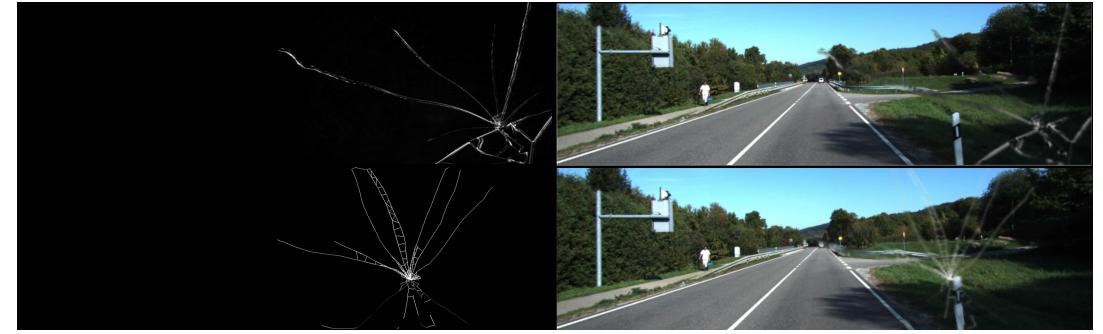


Abbildung 5: Oben links - Rissmuster online auf Freepik gesammelt; oben rechts - online Rissmuster überlagert auf KITTI; unten links - simuliertes Rissmuster mit PBR; unten rechts - simuliertes Rissmuster überlagert auf KITTI.

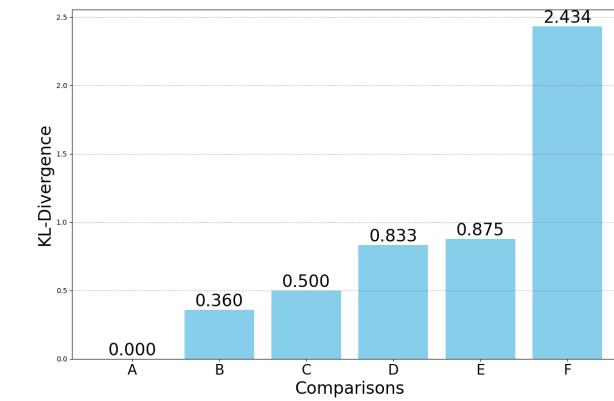


Abbildung 6: K-L-Divergenz verschiedener Paare von Bildverteilungen. Datensätze: RC - Reales On-Road-Datensatz (siehe Abb. 2), KITTI und Katzen. Filter: RO - Real (online gesammelt) und Sim - Simulierte. K-L-Divergenz zwischen (x - Überlagerungsbeziehung): A - (Sim x KITTI) vs (Sim x KITTI); B - (Sim vs RO); C - (Sauberes RC vs KITTI); D - (Beschädigtes RC) vs (RO x KITTI); E - (Beschädigtes RC) vs (Sim x KITTI); F - KITTI vs Katzen.

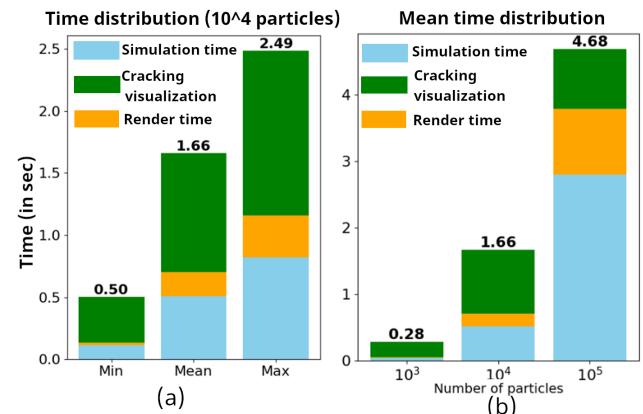


Abbildung 7: (a) Durchschnittliche Zeit, die von verschiedenen Modulen unserer Pipeline über 100 Durchläufe benötigt wird. (b) Die minimale, maximale und durchschnittliche Zeit, die von verschiedenen Modulen über 100 Durchläufe für ein 10^4 Partikelgitter benötigt wird. In diesen Diagrammen zeigen wir die für die Simulation benötigte Zeit (Simulationszeit), die Umwandlung des Gitters in Glas (Rissvisualisierung) und schließlich das Rendering (Renderzeit).

schlagswinkel zugeschrieben werden. Die Visualisierung der Risse und die Renderzeit variieren ebenfalls aufgrund unterschiedlich großer Masken, die durch verschiedene Bruchmuster entstehen. Wir variieren auch die Anzahl der Partikel und beobachten, wie die Laufzeit exponentiell mit der Zunahme der Partikel steigt. Alle diese Durchläufe wurden auf Bildern aus dem KITTI-Datensatz mit den Abmessungen von $(375 \times 1242 \times 3)$ gerendert.

Fazit und zukünftiger Umfang

Wir haben eine neuartige Klasse von adversarialen Fehlern eingeführt, die aus dem physikalischen Prozess von Fehlern in der Kamera resultieren. In diesem Papier stellen wir einen Ansatz vor, um ein realistisches zerbrochenes Glas-Muster aus einer physikalischen Simulation zu erzeugen und dieses anschließend in bestehende Bilddatensätze mittels physikalisch basierter Darstellung einzubetten. Wir zeigen, dass die simulierten adversarialen Bilder zu erheblichen Fehlern in der Objekterkennung führen können.

In dieser Arbeit befassen wir uns mit Black-Box-Angriffen, die aus realen, natürlich vorkommenden physikalischen Phänomenen resultieren, die nicht künstlich entwickelt wurden, um spezifische Modellschwachstellen auszunutzen.

Wir gehen davon aus, dass keine Kenntnisse über die Modellattribute, Gewichte oder Architektur vorliegen, wodurch Angriffe auf verschiedene Modelle übertragbar sind. Physische adversarielle Methoden (Translucent Patch, RP2) können alle als Verdeckungen entweder der Kamera oder der erfassten Objekte bezeichnet werden. Die Adversarialität ergibt sich aus der Wirkung der Modellinference aufgrund dieser Verdeckungen. Unsere PBR-Pipeline mischt die Risse mit den Quellbildern als durchscheinende, verschwommene Muster, die den latenten Raumcodierung beeinflussen, anstatt eine direkte Verdeckung zu verursachen, was zu falschen Erkennungen führt.

Während diese Arbeit speziell eine physikbasierte Methode zur Erzeugung von zerbrochenen Glasmustern einführt, umfassen Kameraausfälle auch andere Effekte wie Sonnenblendung, Überbelichtung, Unterbelichtung, Kondensation usw. Unsere zukünftige Arbeit wird sich darauf konzentrieren, ein adversariales Toolkit für die realistische Erzeugung dieser Effekte mithilfe von Physik zu erstellen und sie anschließend auf bestehende Bilddatensätze und Auto-Simulationsplattformen zu platzieren, um die weitere Forschung in diesem Bereich der partiellen Kameraausfälle zu fördern.

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 règles des maximis et minimis à quelques problèmes de statique relatifs, à 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.

Referenzen

- Akhtar, N.; und Mian, A. 2018. Bedrohung durch adversariale Angriffe auf Deep Learning in der Computer Vision: Eine Umfrage. *Ieee Access*, 6: 14410–14430.
- Carlini, N.; und Wagner, D. 2017. Adversarielle Beispiele sind nicht leicht zu erkennen: Umgehung von zehn Erkennungsmethoden. In *InProceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 3–14.
- Ceccarelli, A.; und Secci, F. 2022. Ausfälle von RGB-Kameras und ihre Auswirkungen auf Anwendungen im autonomen Fahren. *IEEETransactions on Dependable and SecureComputing*. Coulumb, C.-A. 1776. Essai sur une application des règles des maximis et minimis à quelques problèmes de statique relatifs, à l'architecture. *Mem. Acad. Roy. Div. Sav.*, 7: 343–387. Dalvi, N.; Domingos, P.; Mausam; Sanghai, S.; und Verma, D. 2004. Adversarielle Klassifikation. 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.; und Song, D. 2018. Robuste physische Angriffe auf die visuelle Klassifikation durch Deep Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1625–1634. Geiger, A.; Lenz, P.; Stiller, C.; und Urtasun, R. 2013. Vision trifft Robotik: Der KITTI-Datensatz. *International Journal of RoboticsResearch(IJRR)*. Goodfellow, I. J.; Shlens, J.; und Szegedy, C. 2014. Erklärung und Nutzung adversarier Beispiele. *arXiv preprintarXiv:1412.6572*. Iben, H. N.; und O’Brien, J. F. 2009. Erzeugung von Oberflächenrissmustern. *GraphicalModels*, 71(6): 198–208. Jocher, G.; Chaurasia, A.; und Qiu, J. 2023. Ultralytics YOLO. Kong, Z.; Guo, J.; Li, A.; und Liu, C. 2020. Physgan: Erzeugung physisch-weltresistenter adversarier Beispiele für autonome Fahren. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14254–14263. Kuna, M. 2013. Finite Elemente in der Bruchmechanik. *Solid Mechanics and Its Applications*, 201: 153–192. Kurakin, A.; Goodfellow, I. J.; und Bengio, S. 2018. Adversarielle Beispiele in der physischen Welt. In *Artificial Intelligence Safety and Security*, 99–112. Chapman and Hall/CRC. Li, J.; Schmidt, F.; und Kolter, Z. 2019. Adversarielle Kamerasticker: Ein physischer, kamerabasierter Angriff auf Deep-Learning-Systeme. In *International Conference on Machine Learning*, 3896–3904. PMLR Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; und 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. 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.; und 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.; und Shao, L. 2022. Pvt v2: Verbesserte Baselines mit Pyramid Vision Transformer. *Computational visualmedia*, 8(3): 415–424.

Liu, Y.; Xing, Y.; Li, C.; Yang, C.; und Xue, C. 2021. Analyse des Linsenbruchs in der Präzisionsglasformung mit der Finite-Elemente-Methode. *Applied Optics*, 60(26): 8022–8030. Nguyen, A.; Yosinski, J.; und Clune, J. 2015. Tiefe neuronale Netzwerke lassen sich leicht täuschen: Hochsichere Vorhersagen für nicht erkennbare Bilder.

In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 427–436. O'Brien, J. F.; und Hodgins, J. K. 1999. Grafische Modellierung und Animation von sprödem Bruch. In *Proceedings of ACM SIGGRAPH 1999*, 137–146. ACM Press/Addison-Wesley Publishing Co. Pfaff, T.; Narain, R.; De Joya, J. M.; und O'Brien, J. F. 2014. Adaptives Reißen und Brechen dünner Blätter. *ACM Transactions on Graphics (TOG)*, 33(4): 1–9. Pharr, M.; Jakob, W.; und Humphreys, G. 2023. *Physically based rendering: From theory to implementation*. MIT Press.

PK, A. ????. Kaggle Katzen und Hunde Mini-Datensatz.

h
t
t
p
s://
w

www.kaggle.com/datasets/aleemaparakatta/cats-and-dogs-mini-dataset. Zugriff: 2024-09-30. Rankine, W. J. M. 1857. II. Über die Stabilität von lockerem Erdreich. *Philosophical transactions of the Royal Society of London*, (147): 9–27. Ren, S.; He, K.; Girshick, R.; und Sun, J. 2016. Faster R-CNN: Hin zu Echtzeit-Objekterkennung mit Region-Vorschlagsnetzwerken.

IEEETransactions on pattern analysis and machine intelligence, 39(6): 1137–1149.

Rouxel, T.; und Brow, R. K. 2012. Der Fluss und Bruch von fortschrittlichen Gläsern—ein Überblick. *International Journal of Applied Glass Science*, 3(1): 1–2. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; und Fergus, R. 2013. Faszinierende Eigenschaften von neuronalen Netzwerken. *arXiv preprintarXiv:1312.6199*.

Tu, J.; Ren, M.; Manivasagam, S.; Liang, M.; Yang, B.; Du, R.; Cheng, F.; und Urtasun, R. 2020. Physisch realisierbare adversarielle Beispiele für Lidar-Objekterkennung. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 13716–13725. van Schrick, D.

1997. Bemerkungen zur Terminologie im Bereich der Überwachung, Fehlererkennung und 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.; und SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamentale Algorithmen für wissenschaftliches Rechnen in Python. *Nature Methods*, 17: 261–272.

Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; und Shao, L. 2022. Pvt v2: Verbesserte Baselines mit Pyramid Vision Transformer. *Computational visualmedia*, 8(3): 415–424.

Yu, F.; Chen, H.; Wang, X.; Xian, W.; Chen, Y.; Liu, F.; Madhavan, V.; und 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.; und 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.; und Darrell, T. 2020. BDD100K: Ein vielfältiger Fahrdatensatz für heterogenes Multitask-Lernen. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2636–2645.

Zolfi, A.; Kravchik, M.; Elovici, Y.; und Shabtai, A. 2021. Der Translucent Patch: Ein physischer und universeller Angriff auf Objektdetektoren. 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:    $frontiers \leftarrow KDTtree - queryRadius(R)$ 
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).

Algorithmus der Spannungsverbreitung

Algorithmus 1 beschreibt das Verfahren zur Simulation der Verbreitung von Spannung durch ein Material nach einem Einschlagereignis. Der Algorithmus nimmt als Eingaben den Ort des Einschlags (pt), die Größe der Einschlagkraft (F), den Einschlagsrichtungsvektor (v) und die mit der Einschlagstelle verbundene Elternkante (PE). Er verwendet auch einen Nächster-Nachbar-Radius R , um die Menge der Kandidatenorte für die Spannungsverbreitung zu bestimmen.

Algorithmus 1: Spannungsverbreitung

```

1:  $pt \leftarrow$  Einschlagpunkt
2:  $F \leftarrow$  Einschlagkraft
3:  $PE \leftarrow$  Elternkante
4:  $v \leftarrow$  Einschlagsvektor
5:  $R \leftarrow$  Nächster-Nachbar-Radius
6:
7: Prozedur STRESSVERBREITUNG( $Pt, F, V, PE$ )
8:    $frontiers \leftarrow KDTtree - queryRadius(R)$ 
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:  PROPAGIERSTRESS( $Pt, F, V, PE$ )
16: Prozedur beenden
```

Zuerst verwendet es eine KD-Baum-Datenstruktur, um effizient alle Punkte (Grenzflächen) innerhalb eines gegebenen Radius R vom Einschlagpunkt abzufragen. Für jede Grenzfläche berechnet es einen Einheitsrichtungsvektor vom Einschlagpunkt zur Grenzfläche (NN). Dann projiziert es den Einschlagsvektor v auf diese Richtung, um die Kosinusähnlichkeit $\cos(\theta)$ zu erhalten, die die Winkelbeziehung zwischen der Einschlagsrichtung und der Kandidaten-Ausbreitungsrichtung erfasst. Für jeden Kandidaten wird der resultierende Wert zusammen mit der Einschlagkraft verwendet, um die entsprechende Spannung an diesem Punkt zu berechnen. Der Algorithmus wählt dann den Kandidaten mit dem maximalen Spannungswert aus. Der Einschlagsvektor v und die Elternkante PE werden aktualisiert, um dieser neuen Richtung zu entsprechen. Der Prozess wird rekursiv wiederholt, sodass die simulierte Spannungswelle iterativ durch das Material entlang des Pfades der größten Spannungsübertragung propagiert.

Dieser Ansatz zielt darauf ab, nachzuahmen, wie sich Spannung von einem Einschlagpunkt am wahrscheinlichsten durch ein Material ausbreitet—bevorzugt entlang von Pfaden, die sowohl durch geometrische Nähe als auch durch mechanische Ausrichtung mit dem ursprünglichen Einschlag definiert sind.

Das endgültige Ergebnis der Sim ist die Darstellung des Netzes als Bild, das dem Muster einer zerbrochenen Linse entspricht (Endbild von Abb. 8).

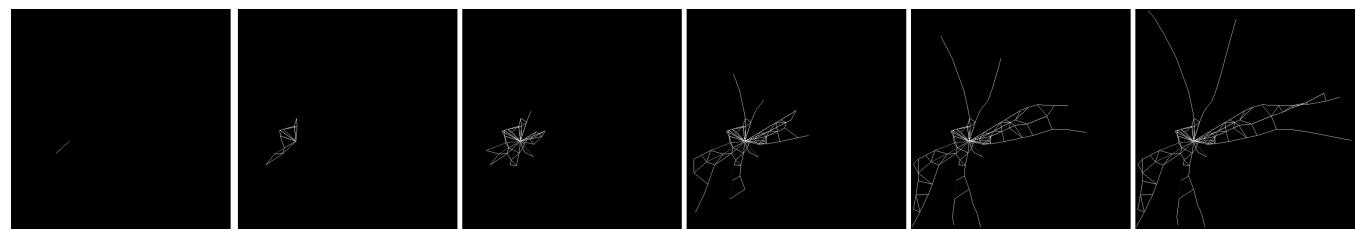
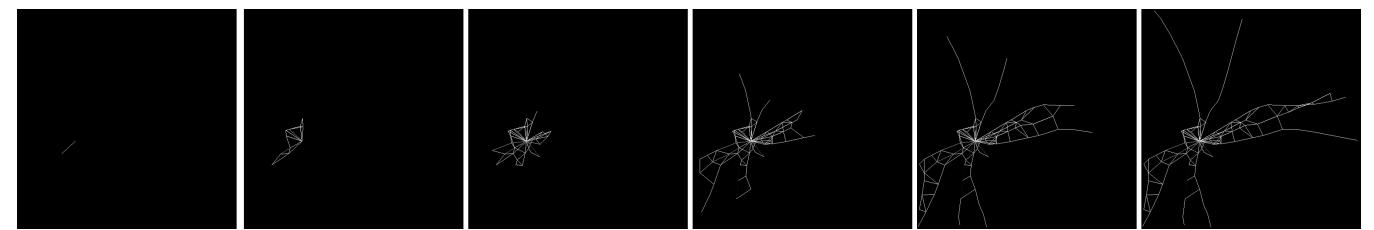


Figure 8: An animation of fracturing of a lens simulated by setting the stress field and applying PBR.



Figu^{Abb. 8: Eine Animation des Bruchs einer Linse, simuliert durch das Setzen des Spannungsfeldes und die Anwendung R.}
von PB

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

Statisches Experiment

Um die Auswirkungen dieser Brüche auf die resultierenden Bilder zu verstehen, führen wir zunächst ein statisches Experiment in Innenräumen durch, wie im Abschnitt Einführung erwähnt. Für dieses Experiment verwenden wir verschiedene gehärtete Glasscheiben, die wir zufällig mit einem kleinen Hammer an einem oder mehreren Bruchpunkten zerbrechen. Dann platzieren wir eine 36 MP JVC GC-PX10 Hybridkamera, die mit einer Klemme auf einem Stativ vor dem gehärteten Glas montiert ist.

Abb. 9(a) zeigt den detaillierten Aufbau mit der Kamerahalterung und dem gehärteten Glas, das mit einer Klemme fixiert ist. Abb. 9(b) zeigt das von der Kamera aufgenommene Bild und Abb. 9(c) zeigt das einzelne Fahrzeug, das als Hauptobjekt durch das gehärtete Glas von der Kamera erfasst wird. Die Szene wird mit Deckenleuchtstofflampen beleuchtet.

Abb. 10 zeigt einige der Bruch-/Kratzer-Muster auf dem gehärteten Glas. Diese Muster wurden absichtlich randomisiert, indem mehrere Brennpunkte und unterschiedliche Kraftstärken verwendet wurden, um die unvorhersehbare und vielfältige Natur von Glasschäden in der realen Welt nachzuhahmen. Durch die Anwendung unterschiedlicher Kraftstärken konnten wir ein Spektrum von Brüchen und Kratzern erzeugen, das von feinen Oberflächenabnutzungen bis zu ausgeprägteren Brüchen reicht. Dieser Ansatz wurde gewählt, um die Arten von Schäden, die Glasoberflächen unter realen Bedingungen erleiden können—wie durch Stöße, Trümmer oder Umwelteinflüsse verursacht—möglichst genau zu replizieren und so die Relevanz und Realitätsnähe unseres experimentellen Aufbaus sicherzustellen. Diese repräsentativen Schadensmuster ermöglichen es uns, den Einfluss von Glasunvollkommenheiten auf die Sensorleistung und Objekterkennungsalgorithmen effektiver zu analysieren.

Zwei verschiedene Bruchmuster und ihre resultierenden Bilder sind in Abb. 11 und Abb. 12 dargestellt. Wir möchten darauf hinweisen, dass wir die Brennweiten der Kamera erheblich variiert haben, um zu verstehen, wie die Bilder bei Nah- und Fernfokus aussehen. Die Ergebnisse zeigen, dass selbst geringfügige Kratzmuster im Bildausgang sichtbar werden, während ein viel stärkeres Mehrfachbruchmuster fast das gesamte Bild verwischen kann. Dieses Experiment liefert die Intuition, auf der unser Simulations- und Visualisierungsrahmen basiert.

Erhöhte AP für Fußgänger in KITTI

Wir möchten darauf hinweisen, dass die erhöhte AP für die Fußgängerklasse etwas war, das selbst uns zunächst überrascht hat. Eine sorgfältige qualitative Tiefenanalyse half uns jedoch zu verstehen, dass dies aufgrund der Gläserreiss geschah, die es dem Modell erleichterten, Fußgänger zu klassifizieren, da die Kanten um sie herum verstärkt wurden. Dies war kein Kantenartefakt, sondern vielmehr der Gläserriss, der als zusätzliche Kantenbegrenzung fungierte und den Fußgänger klar vom Hintergrund trennte. Ein ähnliches Ergebnis wurde auch in [1] beobachtet, wo die gesamte AP in adversarialen Bildern erhöht wurde.



Abbildung 9: Experimenteller Aufbau zur Erfassung von Bildern, die durch zerkratzte/ gebrochene Außenschichten einer Kamera beeinträchtigt sind. (a) zeigt den gesamten Aufbau zur Aufnahme von adversarialen Bildern. (b) zeigt die Position der Kamera in Bezug auf die aufgenommene Szene. (c) zeigt die Szene, die von der Kamera erfasst wird.



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



Abbildung 10: Einige Bruch-/Kratzer-Muster auf dem Glas, das wir zur Bildersammlung verwendet haben. (a) Eine scharfe Kraft, die senkrecht zur Glasoberfläche angewendet wird, erzeugt radial verlaufende Brüche. (b) und (c) replizieren ein Glas mit Kratzern.

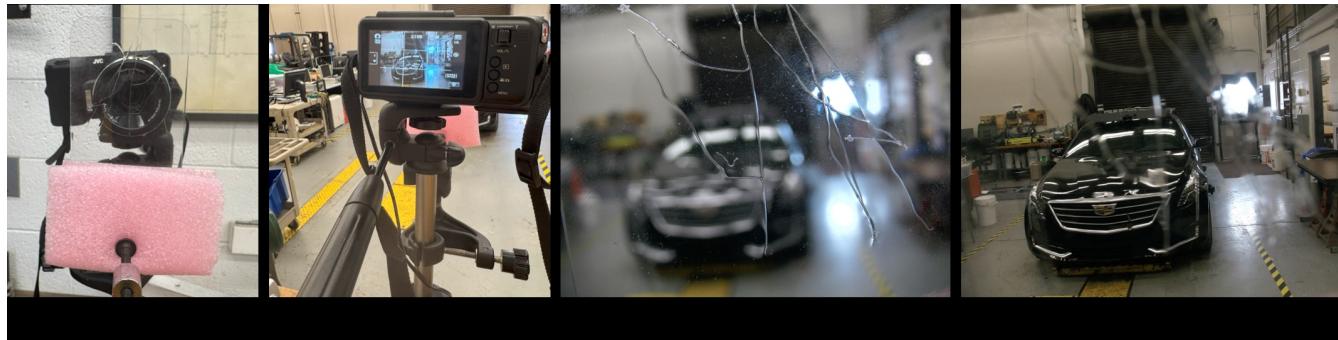


Figure 11: (a) Shows the scratched pattern placed in front of the camera, (b) shows the camera POV. (c) shows the image captured by the camera (short-focus). (d) shows the image captured by the camera (far-focus)

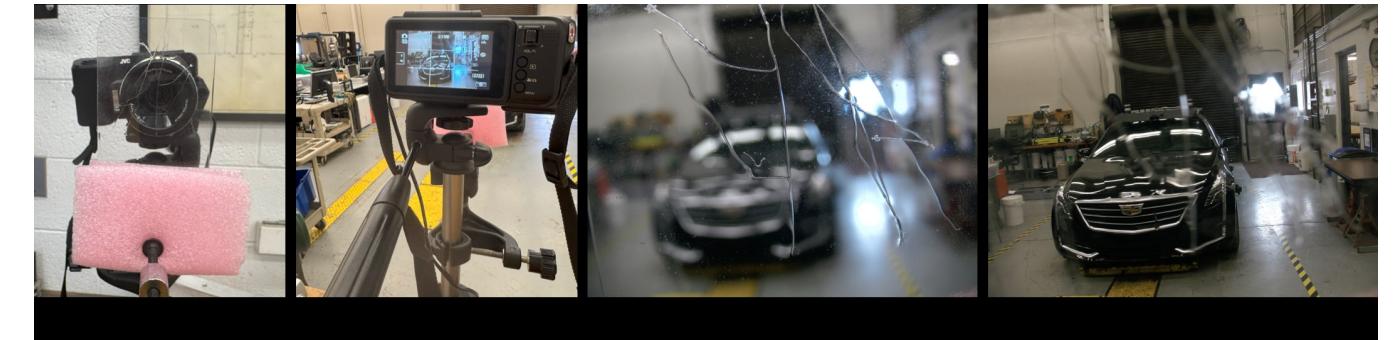


Abbildung 11: (a) Zeigt das Kratzmuster vor der Kamera, (b) zeigt die Kamera-Perspektive. (c) zeigt das von der Kamera aufgenommene Bild (Kurzfokus). (d) zeigt das von der Kamera aufgenommene Bild (Fernfokus).

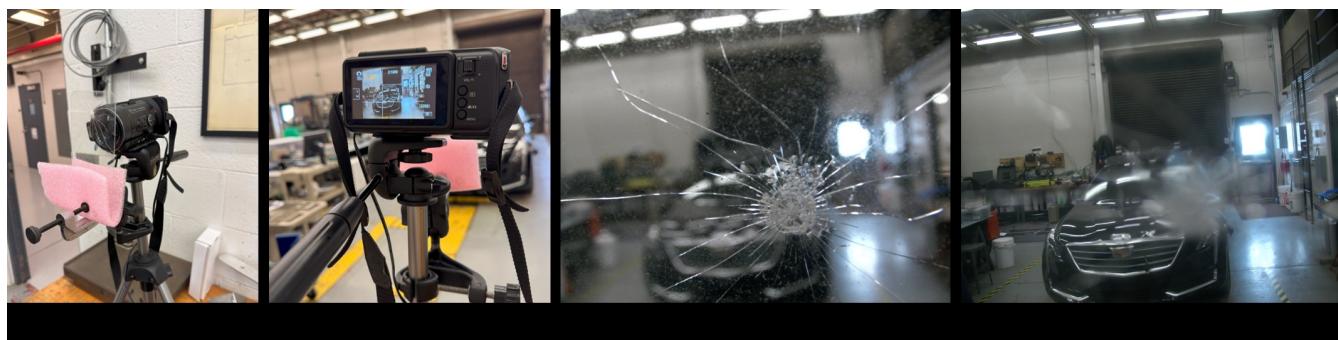


Figure 12: (a) Shows the broken glass pattern placed in front of the camera, (b) shows the camera POV. (c) shows the image captured by the camera (short-focus). (d) shows the image captured by the camera (far-focus)

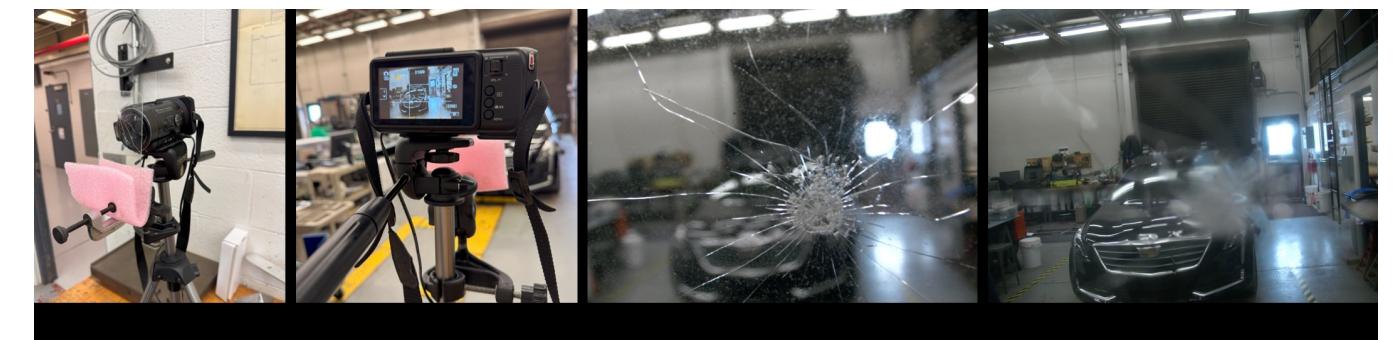


Abbildung 12: (a) Zeigt das zerbrochene Glas-Muster vor der Kamera, (b) zeigt die Kamera-Perspektive. (c) zeigt das von der Kamera aufgenommene Bild (Kurzfokus). (d) zeigt das von der Kamera aufgenommene Bild (Fernfokus).

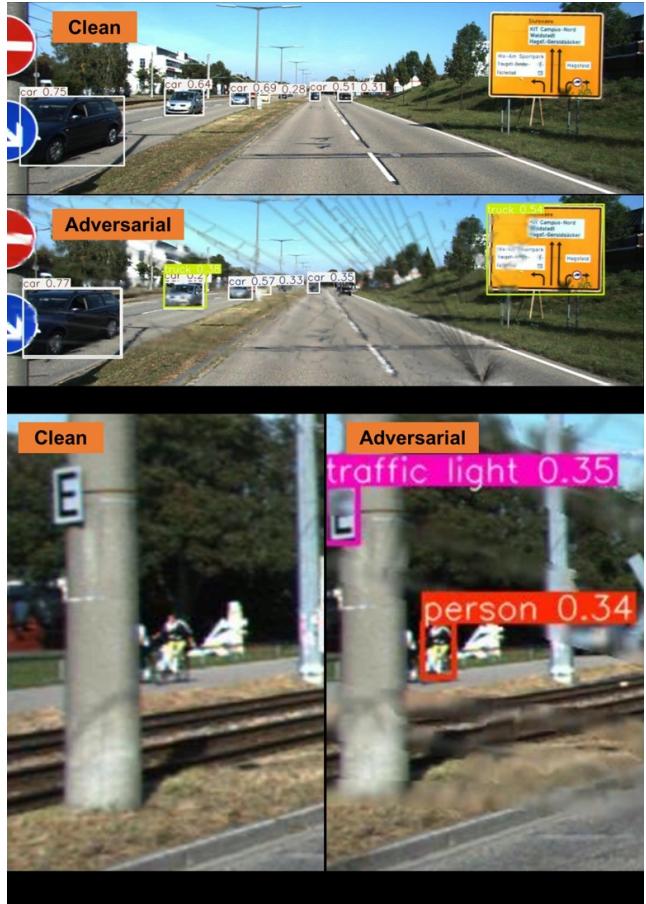


Figure 13: (a) Top - detections on a clean image; bottom - detections on an adversarial image.(b) YoLo fails to detect the person (c) Glass cracks allows the model to detect the person.

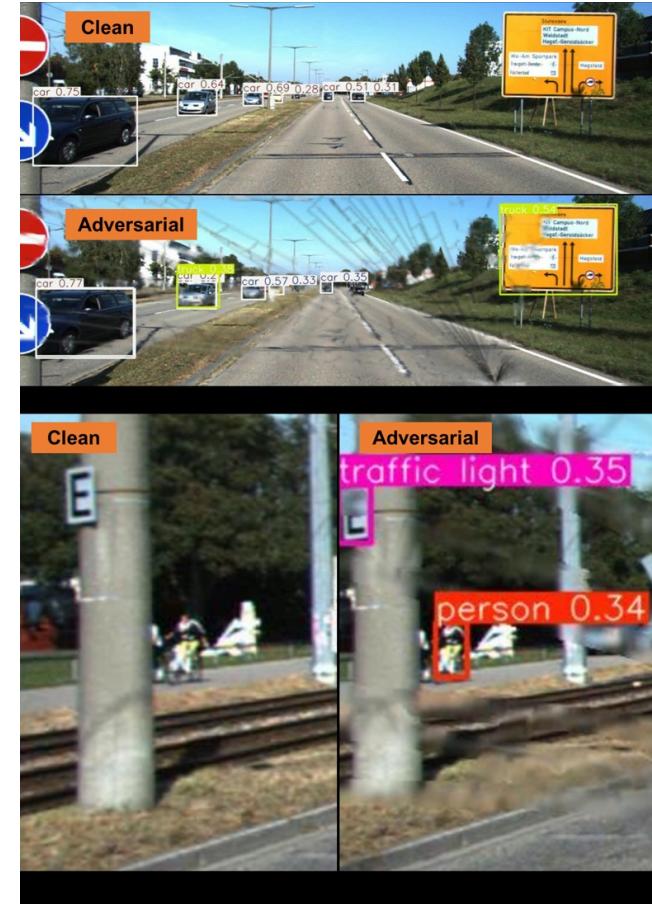


Abbildung 13: (a) Oben - Erkennungen auf einem sauberen Bild; unten - Erkennungen auf einem adversarialen Bild. (b) YoLo erkennt die Person nicht (c) Glasrisse ermöglichen es dem Modell, die Person zu erkennen.

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).

Dynamisches Experiment

In diesem Abschnitt beschreiben wir das dynamische Experiment, das im Abschnitt Einführung erwähnt wird. Wir führen dieses Experiment durch, um die zeitliche Störung zu verstehen, die durch einen Riss eingeführt wird. Wir verwenden einen Windschutzscheibenriss eines Fahrzeugs und platzieren eine kleine Kamera auf dem Armaturenbrett hinter dem Riss. Dann fotografieren wir zwei dynamische Objekte - ein Fahrzeug und einen Fußgänger, während sie sich durch die Szene bewegen. Abb. 14 zeigt einige spezifische Bildrahmen mit Schlussfolgerungen von YOLOv8 für die Fahrzeugklasse. Wir zeigen, dass das Fahrzeug mit dem Riss in den meisten Rahmen unentdeckt bleibt. Zusätzlich enthält fast jeder Rahmen ein falsch positives Ergebnis. Entsprechend präsentieren wir Abb. 15 als die Rahmen mit einer Person, die in der Szene geht. Wir zeigen, dass es intermittierend Erkennungen liefert und gelegentlich mit einer falschen Klasse (Surfbrett).



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.



Abbildung 14: Bestimmte Bilderrahmen der mit dem Windschutzscheibenriss aufgenommenen Bilder mit YOLOv8-Inferenz für die Fahrzeugklasse. A - Falsch positiv ohne Objekt in der Szene; B - keine Inferenz auf Fahrzeug; C - keine Inferenz auf Fahrzeug; D - erste Erkennung auf Fahrzeug; E - zwei verschiedene Erkennungen auf demselben Fahrzeug; F - falscher Bereich des Begrenzungsrahmens.



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.

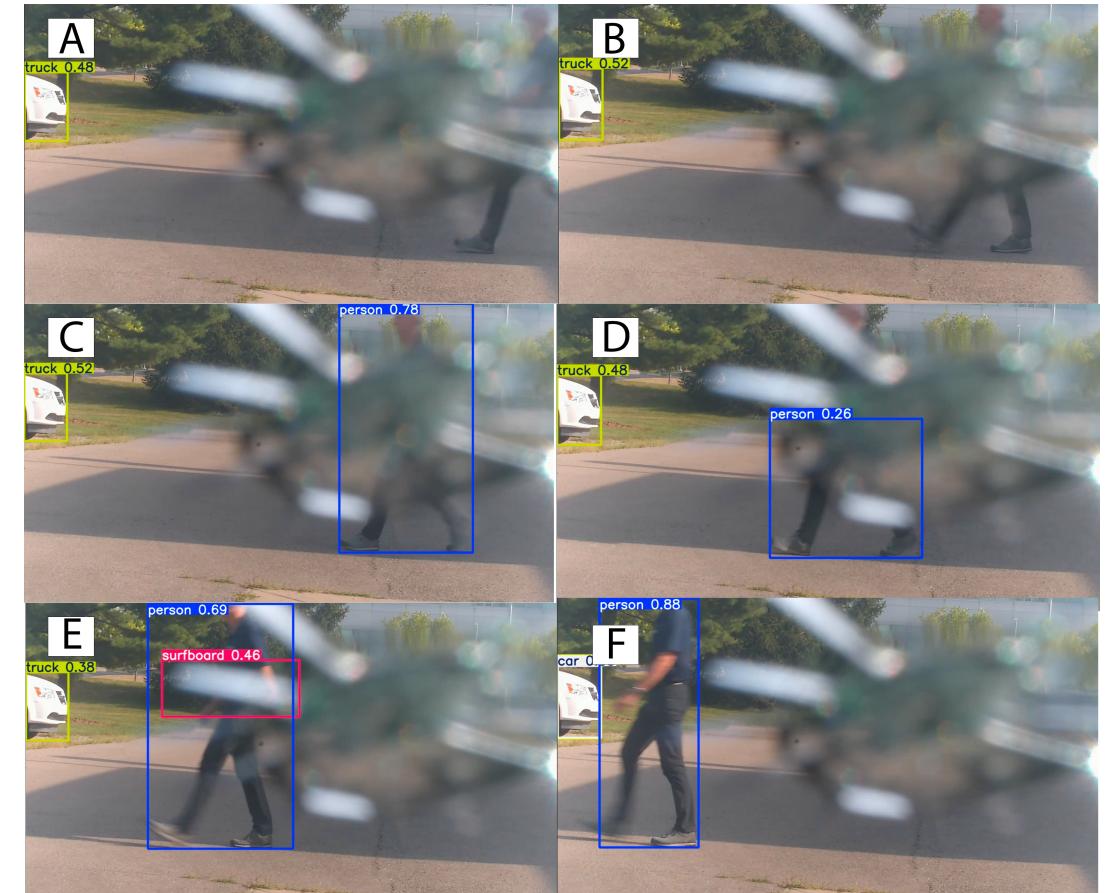


Abbildung 15: Spezifische Bilderrahmen der mit dem Riss in der Windschutzscheibe aufgenommenen Bilder mit YOLOv8-Inferenz für die Klasse Person. A - erster Eintritt der Person in die Szene ohne Erkennung; B - keine Inferenz der Person; C - erste Erkennung der Person; D - teilweise Erkennung der Person; E - Erkennung der Person mit anderer Klasse; F - vollständige Erkennung der Person.

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 Glasbruchbilder

Wir präsentieren ein Beispiel der Glasbruchbilder, die von der FreePik-Website gesammelt und auf den KITTI-Datensatz überlagert wurden, zusammen mit der YOLOv8-Inferenz (Abb. 16). Wir zeigen, dass der Bruch einige Erkennungen entfernt und das Erkennungsvertrauen anderer verringert.

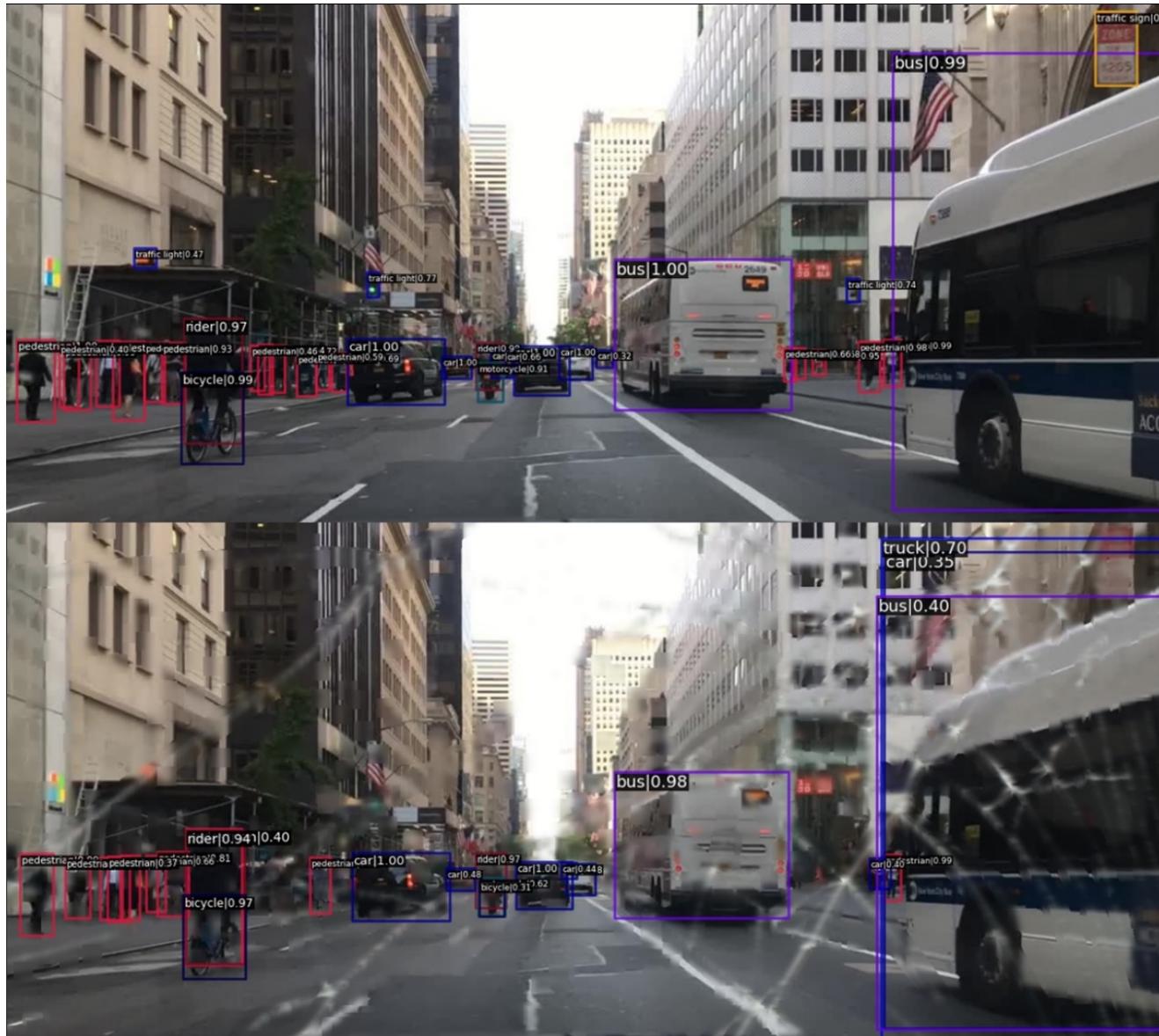


Figure 16: Top - Inference of PTv2 on a clean image from BDD100k. Bottom - Inference for a real broken glass image overlaid on BDD100k for comparison. We see two extra false positives in on the right side (truck, car) and several false negatives for the pedestrian class on the left of the adversarial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

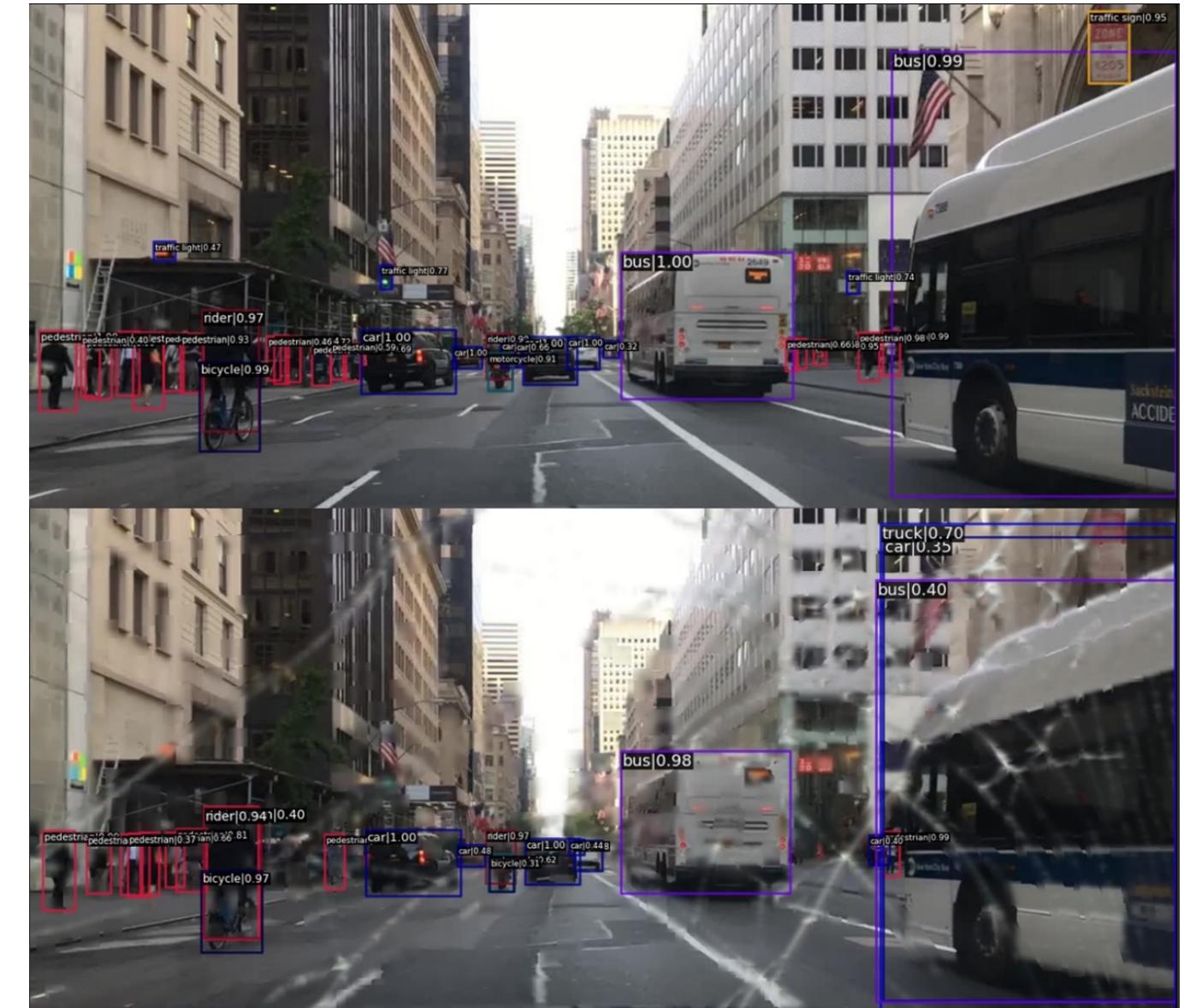


Abbildung 16: Oben - Inferenz von PTv2 auf einem sauberen Bild aus BDD100K. Unten - Inferenz für ein echtes Bild mit zerbrochenem Glas, das zur Vergleichszwecken auf BDD100K überlagert wurde. Wir sehen zwei zusätzliche Fehlalarme auf der rechten Seite (Lkw, Auto) und mehrere Fehlklassifizierungen für die Fußgängerklasse auf der linken Seite des adversarialen Bildes.