

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

Manav Prabhakar, Jwalandhar Girnar, Arpan Kusari*

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

Abstract

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

Code —

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

Introduction

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

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

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

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

Verre fracturé, Caméras défaillantes : Simulation d'échantillons adversariaux basés sur la physique pour les systèmes de conduite autonome

Manav Prabhakar, Jwalandhar Girnar, Arpan Kusari* Institut de recherche sur les transports de l'Université du Michigan 2901 Baxter Road, Salle 202, Ann Arbor, MI-48103 {prmanav, jwala, kusari}@umich.edu

Résumé

Alors que de nombreuses recherches se sont récemment concentrées sur la génération d'échantillons adversariaux basés sur la physique, une catégorie critique mais souvent négligée provient des défaillances physiques au sein des caméras embarquées—composants essentiels aux systèmes de perception des véhicules automatisés. Les défaillances de la caméra, qu'elles soient dues à des contraintes externes provoquant une panne matérielle ou à des défauts de composants internes, peuvent directement compromettre la sécurité et la fiabilité des systèmes de conduite autonome. Premièrement, nous motivons l'étude en utilisant deux expériences réelles distinctes pour démontrer que les défaillances du verre entraînent effectivement l'échec des modèles de réseaux neuronaux basés sur la détection. Deuxièmement, nous développons une étude basée sur la simulation en utilisant le processus physique de la rupture du verre pour créer des scénarios perturbés, représentant une classe réaliste d'échantillons adversariaux basés sur la physique. En utilisant une approche basée sur le modèle par éléments finis (MEF), nous générions des fissures de surface sur l'image de la caméra en appliquant un champ de contrainte défini par des particules dans un maillage triangulaire. Enfin, nous utilisons des techniques de rendu basé sur la physique (PBR) pour fournir des visualisations réalistes de ces fractures physiquement plausibles. Pour évaluer les implications en matière de sécurité, nous appliquons les effets de verre brisé simulés comme filtres d'image à deux jeux de données de conduite autonome—KITTI et BDD100K—ainsi qu'au jeu de données de détection d'image à grande échelle MS-COCO. Nous évaluons ensuite les taux d'échec de détection pour des classes d'objets critiques en utilisant des modèles de détection d'objets basés sur CNN (YOLOv8 et Faster R-CNN) et une architecture basée sur un transformateur avec des Pyramid Vision Transformers. Pour approfondir l'impact distributionnel de ces distorsions visuelles, nous calculons la divergence de Kullback-Leibler (K-L) entre trois distributions de données distinctes, en appliquant divers filtres de verre brisé à un jeu de données personnalisé (capturé à travers un pare-brise fissuré), ainsi qu'aux jeux de données KITTI et Kaggle de chats et chiens. L'analyse de la divergence K-L suggère que ces filtres de verre brisé n'introduisent pas de changements distributionnels significatifs. Notre objectif est de fournir une méthodologie robuste, basée sur la physique, pour générer des échantillons adversariaux qui reflètent les défaillances réelles des caméras, dans le but global d'améliorer la résilience et la sécurité des systèmes de conduite autonome contre de telles menaces physiques.

Code —

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

Introduction

Les caméras sont omniprésentes en tant que capteurs à distance, collectant des données d'un environnement externe non struc-turé et dynamique, souvent dans des conditions difficiles. Une défaillance ou un défaut dans un capteur est une divergence par rapport à l'état fonctionnel dans au moins un paramètre donné du système (van Schrick 1997). Ces défauts peuvent survenir en raison de causes internes (telles que l'usure) ou externes (température, humidité, etc.). Pour les caméras RGB, les causes internes incluent les pixels morts tandis que les causes externes incluent les boîtiers fracturés ou les lentilles extérieures, et la condensation. Ces défaillances soudaines sont difficiles à détecter et impactent négativement les algorithmes de détection d'objets—réduisant la précision et conduisant souvent à des hallucinations comme le montre la Fig. 1. Les défaillances survenant dans un véhicule automatisé (AV) par exemple, peuvent entraîner des problèmes de sécurité critiques entraînant des accidents et, dans certains cas, des décès.

Actuellement, à la connaissance des auteurs, il n'existe pas de méthodes rigoureuses pour générer des défaillances de capteurs basées sur des caméras (Ceccarelli et Secci 2022).

Dans ce travail, nous nous concentrons sur la défaillance du capteur due à des fractures dans tout verre recouvrant une caméra (ou un boîtier de caméra), bien que le processus détaillé dans cet article puisse être utilisé pour n'importe laquelle des défaillances de la caméra listées dans (Ceccarelli et Secci 2022). Ces effets de fracture du verre dans une caméra peuvent être causés par un objet externe frappant la caméra ou à la suite d'une montée soudaine de chaleur et/ou de pression à l'intérieur du boîtier. Dans le langage des réseaux neuronaux, une image capturée dans de telles conditions est considérée comme un échantillon adversarial. Des recherches antérieures (Akhtar et Mian 2018; Carlini et Wagner 2017; Szegedy et al. 2013) montrent que même de petites quantités de corruptions, parfois difficiles à percevoir par l'œil humain, suffisent à tromper complètement les réseaux neuronaux où un changement subtil des entrées peut entraîner un changement radical des sorties. Nous tenons à noter que (Li, Schmidt et Kolter 2019) ont fourni un paradigme d'attaque adversariale basé sur une caméra physique, qui constitue le travail le plus proche dans ce domaine. Ils ont présenté une modification de l'image en utilisant une superposition d'un autocollant translucide soigneusement conçu qui a conduit à une mauvaise classification.

Pour comprendre l'effet de ces fractures sur les images résultantes de la caméra, nous avons mené deux expériences distinctes : l'une

*Auteur correspondant Copyright © 2026, Association pour l'avancement de l'intelligence artificielle (www.aaai.org). Tous droits réservés.

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

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

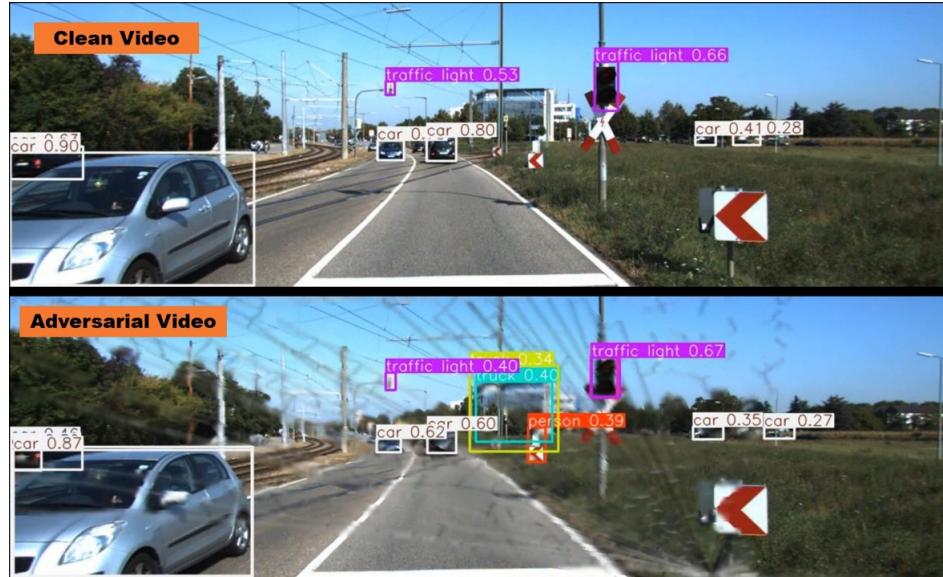


Figure 1: A qualitative comparison of clean vs adversarial video generated using our simulation and rendering method on KITTI. This frame shows false positives, and reduced confidence levels for true positives. Refer to the supplementary material for full video.

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

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

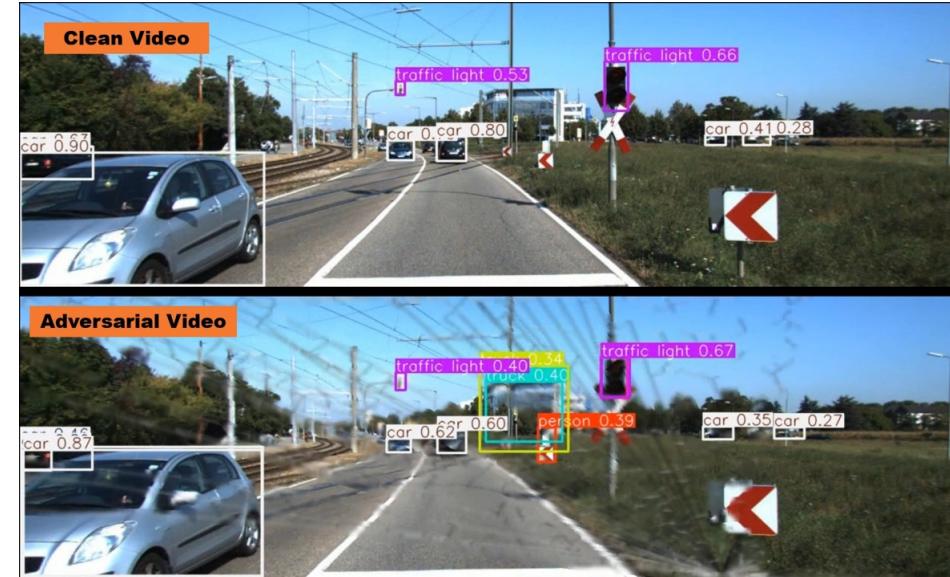


Figure 1 : Une comparaison qualitative entre une vidéo propre et une vidéo adversariale générée à l'aide de notre méthode de simulation et de rendu sur KITTI. Cette image montre des faux positifs et des niveaux de confiance réduits pour les vrais positifs. Référez-vous au matériel supplémentaire pour la vidéo complète.

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

dans un environnement statique intérieur et l'autre dans un environnement extérieur dynamique. La première consistait à fracturer du verre trempé et à le placer devant la caméra (voir Fig. 2(a)) avec un véhicule statique dans la scène pour comprendre comment différents motifs de fracture affectent la qualité et l'apparence de la scène. Nous avons capturé des images à différentes longueurs focales pour évaluer la variabilité de ces corruptions. Cela nous a aidés à répondre à certaines questions qualitatives sur l'apparence visuelle de ces fractures par rapport à leur étendue et leur intensité, motivant notre approche dans la section Plan focal et Simulation d'attaque physique. La configuration expérimentale et les résultats expérimentaux détaillés se trouvent dans la section Expérience statique du Supplémentaire. La deuxième expérience (Fig. 2(b)) consistait à enregistrer une vidéo en extérieur avec des véhicules dynamiques en conditions de lumière du jour en plaçant une caméra MobileEye à côté d'une fissure de pare-brise présentée dans la Fig. 2 (montrée en haut à gauche) et à effectuer une inférence en utilisant YOLOv8 (Jocher, Chaurasia, et Qiu 2023) pour obtenir une compréhension primitive de l'impact de tels scénarios sur les réseaux de détection d'objets. Nous avons observé que le modèle peut facilement détecter le véhicule dans une image propre tandis qu'il souffre d'échecs de détection (en bas à droite) ou génère de faux positifs (en bas à gauche). Fait intéressant, la présence d'une fissure peut également augmenter de manière inattendue la confiance dans la prédiction de la voiture présentant un bord clairement défini (0,92 en bas à gauche contre 0,75 en haut à gauche). Les résultats d'inférence détaillés avec la classe véhicule et personne sont donnés dans la section Expérience dynamique du Supplémentaire.

Nous avons ensuite cherché des images réelles de verre brisé en ligne (Sec. Images de fracture de verre réel du Supplémentaire) mais n'avons pas réussi à constituer un ensemble de données suffisamment grand pour permettre une approche basée sur les données pour la défense contre ces conditions adverses. De plus, nous avons expérimenté avec des outils CGI comme Maya et Blender pour

créer de tels effets, mais ils manquent de flexibilité, de contrôle, d'échelle et de physique pour simuler ces conditions. L'option de simulation la plus proche dans la littérature existante est ArcSim (Pfaff et al. 2014). Cependant, leurs sorties de simulation haute résolution sont extrêmement lentes (≈ 20 heures), ce qui rend difficile la mise à l'échelle. En conséquence, nous avons orienté nos efforts vers la création d'un pipeline de simulation évolutif pour générer des fractures pouvant être utilisées pour faire progresser la pile de perception.

Pour une fracture de verre, le point principal, la force et l'angle d'incidence peuvent être aléatoires, mais la propagation et le motif résultant suivent un processus intrinsèquement physique (soit linéaire, soit radial). Nous construisons donc une simulation de fracture basée sur des particules dans un maillage triangulaire généré aléatoirement et effectuons la propagation du stress à travers le maillage. Notre simulation nous permet de produire les fractures au sein d'un maillage triangulaire à chaque état temporel discret δt . Nous utilisons OpenCV pour convertir le maillage donné en une image de motif de verre brisé correspondant. Nous utilisons ensuite le rendu basé sur la physique (PBR) (Pharr, Jakob, et Humphreys 2023) pour rendre de manière réaliste les fractures de surface en utilisant la fonction de distribution de la réflectance bidirectionnelle (BRDF) en calculant la quantité de lumière réfléchie à partir d'un point donné sur une surface en raison de la source ou des sources de lumière qui y sont incidentes.

En combinant notre approche de rendu avec trois ensembles de données open source populaires - KITTI (Geiger et al. 2013), BDD100k (Yu et al. 2020) et MS-COCO (Lin et al. 2014), nous sommes capables de générer des images adversariales de manière efficace. Un processus courant pour tester les images adversariales générées est de trouver le nombre de faux positifs/négatifs à travers l'espace d'image. Cependant, dans notre cas, en raison de l'effet adversarial étant local, nous ne pouvons pas nous fier simplement à une mesure basée sur l'image. Nous utilisons donc les images adversariales (similaires à la figure en bas à gauche de la Fig. 2) et extrayons les objets

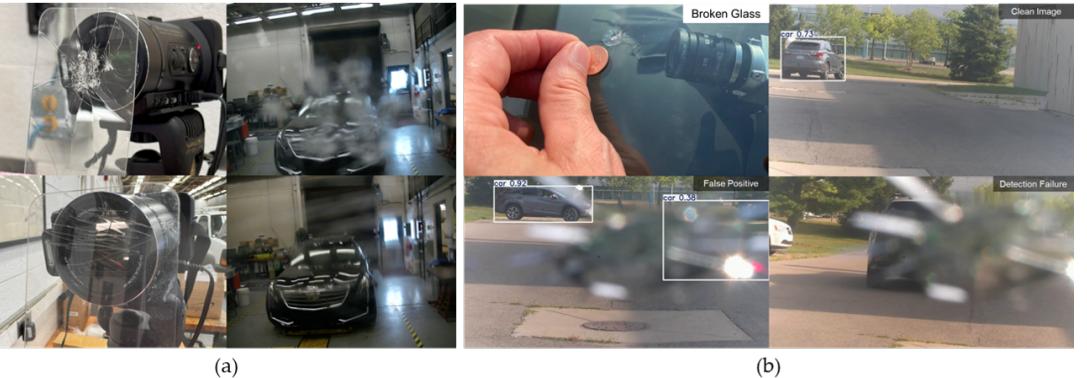


Figure 2: (a) Indoor static experiment. Left: Camera with 2 different fractured tempered glass patterns; right - images of the vehicle under the different fractures. (b) Outdoor dynamic experiment. Top left - a coin sized windshield crack; top right - clean image with the vehicle detected using YOLOv8;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

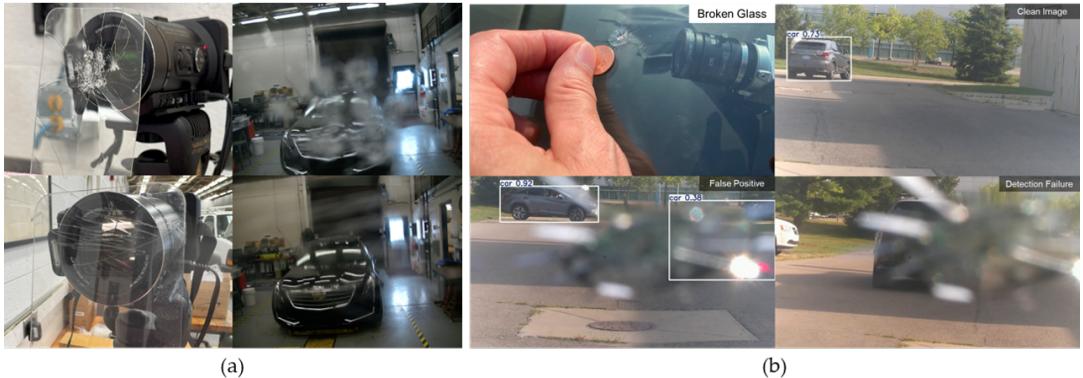


Figure 2 : (a) Expérience statique en intérieur. À gauche : Caméra avec 2 motifs différents de verre trempé fracturé ; à droite - images du véhicule sous les différentes fractures. (b) Expérience dynamique en extérieur. En haut à gauche - une fissure de la taille d'une pièce de monnaie sur le pare-brise ; en haut à droite - image nette avec le véhicule détecté à l'aide de YOLOv8 ; en bas à gauche - faux positif à travers la fissure ; en bas à droite - échec de détection à travers le verre. D'autres exemples de ces expériences sont fournis dans le matériel supplémentaire.

qui se trouvent dans la région où la fracture existe en utilisant les boîtes englobantes de vérité terrain. Nous utilisons ensuite YOLOv8, Faster R-CNN (Ren et al. 2016) et Pyramid Vision Transformer (PVTv2) (Wang et al. 2022) pour trouver le pourcentage d'objets qui échouent lorsque les filtres adversariaux sont appliqués. Nous fournissons également des études d'ablation pour comprendre les différences de distribution entre les trois ensembles d'images : images de verre brisé réelles collectées expérimentalement, images de verre brisé réelles collectées en ligne et les images générées. Nous calculons la divergence de Kullbeck-Liebler (K-L) pour ces distributions d'images afin de prouver la similarité des images générées avec les images de verre brisé réelles. Nous utilisons des images de chats du jeu de données Kaggle Cats and Dogs comme contrôle pour comprendre la différence entre les distributions d'images (PK).

Les principales contributions de l'article peuvent être résumées comme suit :

- Nous proposons une nouvelle manière d'abstraire la fracture du verre grâce à une combinaison de méthodes de propagation du stress et d'arbres couvrants minimaux, pour générer des motifs de verre brisé physiquement cohérents.
- Nous présentons une approche PBR pour faciliter un rendu réaliste des défaillances de la caméra qui peut être utilisé avec tout type de jeux de données de vision par ordinateur existants - à la fois images et vidéos.
- Nos pipelines de simulation et de rendu sont évolutifs et efficaces sur le plan computationnel ($\approx 1.6s$), permettant leur utilisation par le milieu académique et l'industrie pour améliorer la robustesse et la protection hors distribution pour une large gamme d'applications.

Contexte

Échantillons adversariaux basés sur la physique

Le problème de l'échantillon adversarial peut être défini comme suit : pour un modèle M qui classe correctement un échantillon d'entrée X dans sa classe désignée, c'est-à-dire $M(X) = y_{true}$, l'ajout d'une erreur ϵ à l'échantillon d'entrée X résulte en un échantillon modifié \hat{X} tel que $M(\hat{X}) \neq y_{true}$. Ainsi, l'injection de l'erreur ϵ résulte en un échantillon adversarial qui fait échouer le modèle.

Bien que l'idée de manipulation adversariale du modèle ait été identifiée dans le contexte de l'apprentissage automatique il y a déjà un certain temps (Dalvi et al. 2004), au cours de la dernière décennie, l'accent a été mis principalement sur les attaques adversariales contre les réseaux neuronaux (Szegedy et al. 2013; Goodfellow, Shlens, and Szegedy 2014). Dans ces articles, les chercheurs ont montré qu'une petite injection ciblée de bruit, presque imperceptible à l'œil humain, changeait complètement les étiquettes (Szegedy et al. 2013) et, inversement, des images pouvaient être générées qui semblaient totalement méconnaissables pour les humains mais qui avaient des classifications parfaites par les DNNs (Nguyen, Yosinski, et Clune 2015).

Bien que ces échantillons adversariaux testent le modèle pour d'éventuelles défaillances, ils manquent de réalisme physique dans leur génération et nécessitent un accès au modèle. Pour remédier à cela, certaines recherches récentes ont visé à construire des échantillons adversariaux physiquement pertinents. L'une des premières incursions dans ce domaine a été réalisée par (Kurakin, Goodfellow et Bengio 2018) qui ont ciblé la précision des modèles dans le monde physique en fournissant des images bruitées d'un appareil photo de téléphone portable, ce qui a conduit le modèle à classer incorrectement une grande partie des échantillons. Dans le même ordre d'idées, (Eykholt et al. 2018) ont démontré que de véritables panneaux de signalisation peuvent être perturbés avec de simples autocollants physiques placés stratégiquement pour tromper presque parfaitement les algorithmes DL de pointe, même avec des changements de point de vue. D'autres chercheurs ont placé des images adversariales (Kong et al. 2020), des patches translucides sur la caméra (Zolfi et al. 2021) ou des surfaces LiDAR artificielles (Tu et al. 2020) pour générer des échantillons qui trompent les détecteurs d'objets. Bien que ces recherches antérieures utilisent la physique pour générer les échantillons, elles ne proviennent pas de la modélisation d'un processus physique rigoureux et nous visons à combler cette lacune dans ce travail.

théorie du verre fissuré/fracturé

Le sujet de la façon dont le verre se brise et se propage est encore une question de recherche ouverte et controversée, avec plusieurs théories physiques proposées (Rouxel et Brow 2012). Alors que la procédure microscopique de la fissuration du verre est débattue, au niveau macroscopique, la fissuration

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 $\tau_{max} = (\sigma_I - \sigma_{II})/2$ (Coulomb 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

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

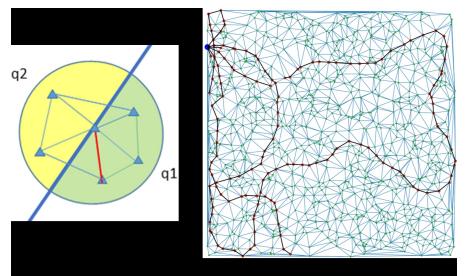


Figure 3: (a) For a splitting plane given in blue, the summed positive stress q_1 and summed negative stress q_2 are compared and the propagation happens on the side with greater summed stress and chosen node which is the closest to the splitting plane (given in red). (b) Shows how we simulate a fracture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

dynamique est bien comprise. (Liu et al. 2021) ont analysé le processus de fissuration des lentilles en verre dans l'application de moulage de précision du verre en utilisant le MEF avec un modèle tridimensionnel dans un logiciel de simulation physique. Les paramètres physiques ont été entrés dans le logiciel et les chemins de fissure ont été analysés à l'aide des résultats de la simulation. Les auteurs ont effectué une simulation de température et de contrainte d'un modèle de maillage tridimensionnel de haute précision du verre moulé. (Iben et O'Brien 2009) ont proposé une méthode pour générer des fractures de surface dans une variété de matériaux, y compris le verre. Comme mentionné dans l'introduction, (Pfaff et al. 2014) ont fourni la simulation de la rupture du verre sous forme de feuille mince, ce qui constitue le travail le plus proche de notre méthode proposée.

Méthodologie

Générer des défaillances réalistes du verre nécessite de créer des simulations à grande échelle basées sur la physique en résolvant la dynamique de fracture sur un maillage d'éléments finis triangulé avec les propriétés du verre.

Simulation de verre brisé

Nous représentons le verre en utilisant des particules échantillonées à partir d'une distribution uniforme répartie sur un plan contraint sous la forme d'un maillage 2D en utilisant la triangulation de Delaunay contrainte. Cela élimine les triangles mal formés et évite les bords inégaux et irréalistes.

Chaque particule p_i a une position x_i et a des voisins les plus proches k_i dans un rayon r qui ont des arêtes existantes avec p_i . Mathématiquement, le maillage de triangulation \mathcal{M} représente un ensemble fini de 2-simplices tel que si

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

Les motifs de fissures dans le verre se produisent en raison du stress causé par la force externe (F) au point d'impact initial p_I en supposant une loi de déformation spécifique (élasticité et plasticité) du verre (G) (Kuna 2013). Nous calculons ensuite les paramètres de résistance sous forme de contrainte effective σ_V au point d'impact (V) comme l'état de contrainte du point d'impact. Les valeurs de contrainte critique pour la résistance du verre σ_C sont trouvées en utilisant des tests sur des échantillons simples avec des conditions de chargement élémentaires (par exemple, test de traction). La fracture se produit alors lorsque la contrainte effective est supérieure à la contrainte critique divisée par le facteur de sécurité (S) :

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

D'après la théorie classique de la résistance des matériaux, nous savons que la rupture dans la plupart des cas est contrôlée par les contraintes principales σ_I et σ_{II} pour les éléments 2D. La fissure initiale se produit soit par la fissure normale-plan qui se situe perpendiculairement à la direction de la contrainte principale la plus élevée σ_I (Rankine 1857), soit par la fissure de cisaillement-plan où les faces de fracture coïncident avec les plans d'intersection de la contrainte de cisaillement maximale $\tau_{max} = (\sigma_I - \sigma_{II})/2$ (Coulomb 1776). Dans le cas du verre, nous supposons que la fracture initiale se produit perpendiculairement à la direction de la contrainte principale maximale.

Depuis le point d'impact initial p_I , la propagation du stress à travers le verre est instable car la fissure se développe brusquement sans qu'il soit nécessaire d'augmenter la charge externe. Depuis p_I ,

le stress se propage dans le voisinage du sommet k_i alors que le stress le long de la direction $\vec{p_I p_j}$ où $p_j \in k_i$ alors

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

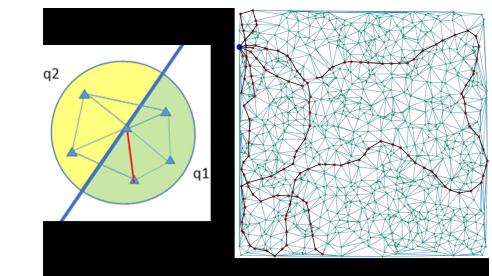


Figure 3 : (a) Pour un plan de séparation donné en bleu, la contrainte positive totale q_1 et la contrainte négative totale q_2 sont comparées et la propagation se produit du côté avec la contrainte totale la plus élevée et le noeud choisi qui est le plus proche du plan de séparation (indiqué en rouge). (b) Montre comment nous simulons une fracture dans un maillage à partir de son point d'impact (marqué en bleu) vers les noeuds subissant une contrainte au-delà de leur résistance seuil (marqué en rouge).

Avec le stress calculé pour chaque arête, le stress positif total (montré dans la Fig. 3) peut alors être donné comme suit :

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

pour la surface continue Ω où se trouve la fonction indicatrice. Le stress positif total pour les simples discrets dans la zone correspondante A de rayon R est donné comme

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

De même, le stress négatif total q_2 est calculé. Ensuite, pour une magnitude plus grande $\max(|q_1|, |q_2|)$, nous choisissons l'arête correspondante avec la plus haute concentration de stress dans le segment donné comme le plan de séparation optimal, car cela fournit le maximum de soulagement de stress. Ainsi, le stress se propage le long des arêtes du maillage, dissipant le stress à chaque noeud.

L'application récursive de la propagation du stress est exécutée jusqu'à la convergence du stress dans tous les états, c'est-à-dire $\sigma_p^{(t)} \approx \sigma_p^{(t-1)} \forall p \in V$.

La propagation du stress dans toutes les directions à travers tous les noeuds entraîne des fissures en retour comme expliqué dans (O'Brien et Hodgins 1999). Pour l'éviter, nous ne propagons que le long des arêtes où les niveaux de stress sont maximaux, mais effectuons une mise à jour du stress sur tous les noeuds voisins. Nous utilisons ensuite un arbre couvrant minimal (ACM) sur un maillage créé à l'aide de ces noeuds stressés. Nous combinons cet ACM avec notre champ de propagation de stress initial le long des arêtes pour calculer le motif final de fissure. L'ACM est une abstraction efficace car il connecte les noeuds qui sont plus proches les uns des autres et dans le champ de stress élevé tout en éliminant les redondances.

Notre processus de calcul de la propagation du stress est défini dans l'algorithme 1 dans le Supplémentaire.

Physically-based rendering

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

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

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

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

Let I_1 be the bidirectional reflectance distribution function,

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

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

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

Then, L_r can be defined as

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

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

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

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

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

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

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

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

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

Focal Plane and Physical attack simulation While we are able to simulate realistic fractures, the primary use case for our work is to be able to generate simulated examples overlayed on existing datasets (KITTI, BDD100K, MS-COCO) and compare them with the real on-road dataset that we created.

Any captured image will exhibit sharp features of the objects in its focal plane. The glass enclosure covering the camera is extremely close and is thus not part of the focal

plane. When the crack happens, the light rays bounce unevenly along the crack and creates a blur (example provided in Fig. 4). We create a binary mask based on the crack pattern and then blur the fractures overlayed on the image. This produces a far-focus image. For a short-focus image, we blur the image and focus on the foreground i.e. the crack.

Experimentation

Dataset

We benchmark two types of broken glass pattern - real and simulated - on three popular open-source datasets - KITTI (Geiger et al. 2013), BDD100k (Yu et al. 2020) and MS-COCO (Lin et al. 2014). The first two represent specific autonomous driving domain while the last one is a general purpose image dataset. The real broken glass pattern images are collected from FreePik website¹ and represent the baseline in our case. We collected 65 images in total and expanded them to a set of 10,000 images via image augmentation using random shifts, image flips and cropping techniques. We also generate 10,000 images using our physics simulator. We then overlay these cracked glass patterns using our PBR pipeline onto every validation image in the datasets and collect the aggregate results. We use three model architectures YOLOv8, Faster R-CNN and PVTv2 model with pretrained weights to generate object detection results.

Implementation

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

Results and Discussion

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

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

¹<https://www.freepik.com>

Rendu basé sur la physique

Une fois que nous avons générés les fractures au niveau du maillage, notre objectif suivant est de créer un rendu visuel de ces fractures. Comme toutes les techniques de PBR, notre méthode est basée sur la théorie des micro-facettes qui stipule que toute surface peut être décrite par de minuscules miroirs parfaitement réfléchissants appelés micro-facettes (Pharr, Jakob, et Humphreys 2023).

Conformément à la théorie des micro-facettes et à la conservation de l'énergie, nous utilisons l'équation de réflectance,

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

où $L_o(x, \omega_o, \lambda, t)$ est la radiance spectrale totale de longueur d'onde λ dirigée vers l'extérieur le long de la direction ω_o à l'instant t , depuis une position particulière x . ω_o est la direction de la lumière sortante. t est le temps. L_e est la radiance spectrale émise et L_r est la radiance spectrale réfléchie.

Soit I_1 la fonction de distribution bidirectionnelle de réflectance,

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

et soit I_2 la radiance spectrale entrant vers x depuis la direction ω_i à l'instant t .

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

Alors, L_r peut être défini comme

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

où Ω est l'hémisphère unitaire centré autour de la normale de surface \mathbf{n} sur ω_i tel que $\omega_i \cdot \mathbf{n} > 0$.

En abstrayant l'équation de réflectance, nous visons à créer un rendu visuel de notre maillage de verre brisé. Nous avons $L_e = 0$ car le verre n'émet pas de lumière. Maintenant, pour calculer L_r , nous considérons toute fissure entre les noeuds comme une micro-facette. Ensuite, nous pouvons définir L_r pour chaque fissure comme :

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

Étant donné les vecteurs unitaires $(\hat{\omega}_\alpha)$ et $(\hat{\omega}_\theta)$ correspondant respectivement aux angles d'azimut (α) et de zénith (θ), nous calculons l'énergie moyenne incidente sur la fissure comme

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

où \hat{n}_i est la normale de surface unitaire de la fissure.

Soit (I_r, I_g, I_b) l'intensité moyenne de la source lumineuse. Alors l'intensité de la fissure, I_c est définie comme

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

Plan focal et simulation d'attaque physique Bien que nous soyons capables de simuler des fractures réalistes, l'utilisation principale de notre travail est de pouvoir générer des exemples simulés superposés sur des ensembles de données existants (KITTI, BDD100k, MS-COCO) et de les comparer avec l'ensemble de données réelles sur route que nous avons créé.

Toute image capturée présentera des caractéristiques nettes des objets dans son plan focal. L'enceinte en verre recouvrant la caméra est extrêmement proche et n'est donc pas partie du plan focal.

Lorsque la fissure se produit, les rayons lumineux rebondissent de manière inégale le long de la fissure et créent un flou (exemple fourni dans la Fig. 4). Nous créons un masque binaire basé sur le motif de la fissure, puis floutons les fractures superposées sur l'image. Cela produit une image à mise au point éloignée. Pour une image à mise au point rapprochée, nous floutons l'image et nous concentrons sur le premier plan, c'est-à-dire la fissure.

Expérimentation

Jeu de données

Nous évaluons deux types de motifs de verre brisé - réel et simulé - sur trois ensembles de données open-source populaires - KITTI (Geiger et al. 2013), BDD100k (Yu et al. 2020) et MS-COCO (Lin et al. 2014). Les deux premiers représentent un domaine spécifique de conduite autonome tandis que le dernier est un ensemble de données d'images à usage général. Les images de motifs de verre brisé réels sont collectées sur le site FreePik¹ et représentent la référence dans notre cas. Nous avons collecté un total de 65 images et les avons étendues à un ensemble de 10,000 images via l'augmentation d'images en utilisant des décalages aléatoires, des retournements d'images et des techniques de recadrage. Nous générions également 10,000 images en utilisant notre simulateur physique. Nous superposons ensuite ces motifs de verre fissuré en utilisant notre pipeline PBR sur chaque image de validation des ensembles de données et collectons les résultats agrégés. Nous utilisons trois architectures de modèles YOLOv8, Faster R-CNN et PVTv2 avec des poids pré-entraînés pour générer des résultats de détection d'objets.

Mise en œuvre

Notre modèle de simulation est développé en échantillonnant aléatoirement 10^4 particules à partir d'une distribution spatiale uniforme dans le cadre donné sur un CPU. Un arbre KD du package Python SciPy (Virtanen et al. 2020) utilisant les paramètres par défaut est construit pour trouver les voisins les plus proches approximatifs de chaque particule. Une triangulation de Delaunay est ensuite exécutée sur les particules pour créer un maillage triangulaire contraint. Nous utilisons une force d'impact de 500 unités avec un point d'impact aléatoire et un vecteur d'impact aléatoire. La propagation du stress se produit jusqu'à ce qu'un seuil de 300 unités soit atteint. Le PBR est effectué sur CPU en implémentant les méthodes décrites dans la section précédente en utilisant OpenCV et Python.

Résultats et Discussion

Un changement majeur par rapport à la plupart des travaux précédents sur les exemples adversariaux est que nos motifs adversariaux générés n'affectent pas universellement tous les pixels d'une image. Par conséquent, la comparaison doit être effectuée uniquement pour la région de l'image où le motif existe. À cette fin, nous créons un masque binaire de chaque motif et produisons les résultats des objets qui existent uniquement dans ce motif.

Le Tableau 1 montre les résultats de la précision moyenne (AP) sous les images adversariales générées en utilisant les deux types de motifs de fissures (collectés en ligne et simulés) pour différentes classes. Pour KITTI, l'AP des autres classes diminue comme prévu avec la diminution de l'AP correspondant au pourcentage d'image occupé par la classe camion enregistrant la plus forte baisse. Pour BDD100K avec PVTv2-B0, nous constatons que la baisse de l'AP est la plus importante dans les images simulées mais globalement,

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-

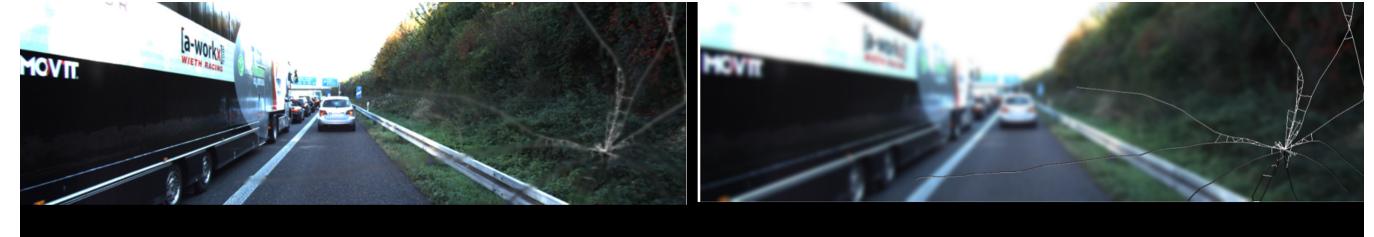


Figure 4 : (a) Montre l'image simulée avec la route et les véhicules dans le plan focal (PBR et mise au point lointaine). (b) désigne le motif de fissure simulé dans le plan focal (PBR et mise au point courte).

Tableau 1 : Précision moyenne (en pourcentage) des différentes classes dans KITTI, BDD100k et MS-COCO sous différentes images adversariales. x fournit la relation de superposition entre le jeu de données et le type de fissure de verre. Clean x Dataset - se réfère directement aux images particulières sans aucun échantillon adversarial. RO x Dataset - se réfère aux images réelles de verre fissuré collectées en ligne superposées sur des images propres. Sim x Dataset - se réfère aux motifs de fissures simulés superposés sur des images propres.

Jeu de données	Seuil IoU	Catégorie	Jeux de données propres x	Jeux de données RO x	Sim x Ensemble de données
KITTI (YOLOv8)	0.5	Piéton	25.64	69.72	17.84
		Camion	12.39	3.59	3.76
		Car	58.99	50.7	57.73
	0.75	Piéton	6.83	33.88	6.02
		Camion	11.29	2.67	2.79
		Car	31.25	23.85	30.15
BDD100k (PVTv2)	0.5	Piéton	66.47	54.33	25.95
		Camion	61.97	52.83	52.02
		Car	80.37	70.14	56.78
	0.75	Piéton	27.06	22.72	10.60
		Camion	47.03	38.23	42.52
		Car	46.23	45.97	42.99
MS-COCO (Faster R-CNN)	0.5	Personne	0.035	0.024	0.024
		Véhicules	2.14	1.45	1.87
		Food	35.34	28.07	30.65
	0.75	Personne	0.032	0.022	0.023
		Véhicules	1.56	1.05	1.07
		Food	24.59	18.85	22.00

la tendance se maintient avec la classe des piétons montrant la baisse la plus marquée. Pour MS-COCO, nous avons agrégé l'AP pour les super-catégories : personne, véhicules et nourriture. Cela est dû au fait que de nombreux objets dans MS-COCO occupent une plus petite surface dans le cadre de l'image, rendant difficile l'obtention de résultats significatifs pour toutes les catégories. Un résultat très intrigant est que la classe des piétons connaît une augmentation multiple de l'AP sous les motifs de verre brisé réels. Bien que cette tendance puisse sembler contre-intuitive, elle résonne avec les résultats de la Fig. 2 où la confiance de la voiture augmente en raison d'un bord. Cela montre en fait que l'AP dépend fortement du motif de fissure, rendant extrêmement important de créer des méthodologies de défense pour atténuer ces attaques adversariales.

Études d'ablation

Nos résultats indiquent que les images simulées obtiennent un effet adversarial similaire à celui des images réelles. Ainsi, une étude d'ablation importante pour nous est de comprendre à quel point les motifs de fissures simulés se rapprochent des motifs de verre fissuré réels et de ceux collectés en ligne. Nous formons 5 distributions

- Jeu de données réels sur route (illustré dans la Fig. 2)

- Motifs de fissures collectés en ligne (Fig. 5 en haut à gauche)
- Motifs de fissures simulés (Fig. 5 en bas à gauche)
- Motifs de fissures simulés superposés sur KITTI (Fig. 5 en bas à droite)
- Motifs de fissures collectés en ligne superposés sur KITTI (Fig. 5 en haut à droite)

Nous calculons maintenant la divergence K-L entre toutes ces distributions pour déterminer à quel point elles se ressemblent (voir Fig. 6). Afin de fournir un contrôle, nous comparons KITTI à des images de chats du jeu de données Kaggle, ce qui donne une divergence K-L de 2,434. Sur cette échelle, les images PBR de verre brisé ont une différence de 0,36 par rapport aux motifs réels de verre brisé, tandis que les filtres de verre brisé superposés sur les images KITTI ont une divergence K-L si millaire.

La Fig. 7 montre une analyse du temps de calcul pour chacun de nos modules et pour différents nombres de particules. Nous effectuons cette analyse sur 100 exécutions, générant des points d'impact aléatoires, des angles d'impact et une structure de maillage avec un nombre fixe de particules. La différence de temps de calcul pour différentes exécutions peut être attribuée au point d'impact et à l'angle d'impact.

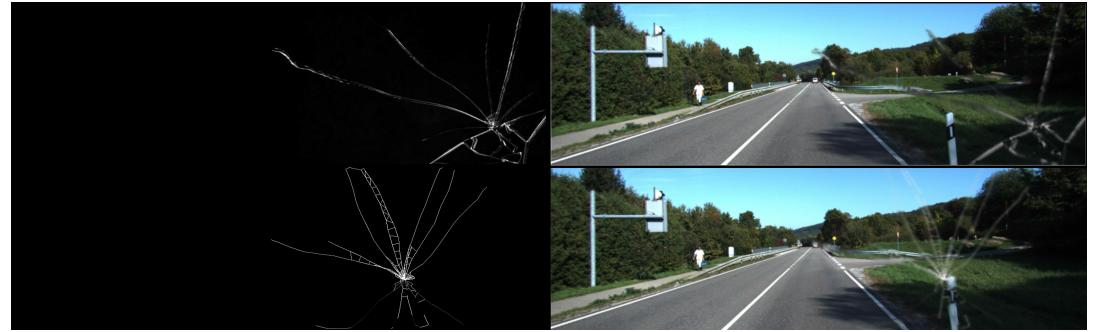


Figure 5: Top left - Crack pattern collected online on Freepik; top right - online crack pattern overlayed on KITTI; bottom left - simulated crack pattern with PBR; bottom right - simulated crack pattern overlayed on KITTI.

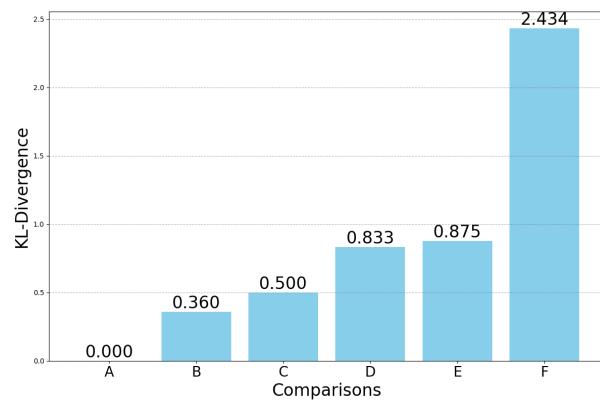


Figure 6: K-L divergence of different pairs of image distributions. Datasets: RC - Real on-road dataset (see Fig. 2), KITTI and Cats. Filters: RO - Real (collected online) and Sim - Simulated. K-L divergence between (x - overlay relation): A - (Sim x KITTI) vs (Sim x KITTI); B - (Sim vs RO); C - (Clean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

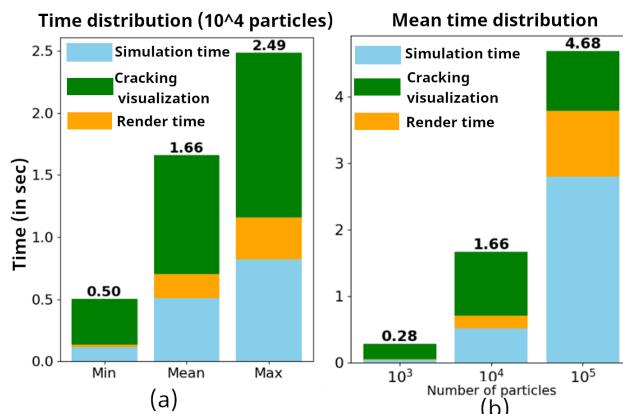


Figure 7: (a) Mean time taken by different modules of our pipeline across 100 runs. (b) The minimum, maximum and mean time taken by different modules across 100 runs for a 10^4 particle mesh. For these plots, we showcase the time taken for simulation (simulation time), converting the mesh to glass (cracking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

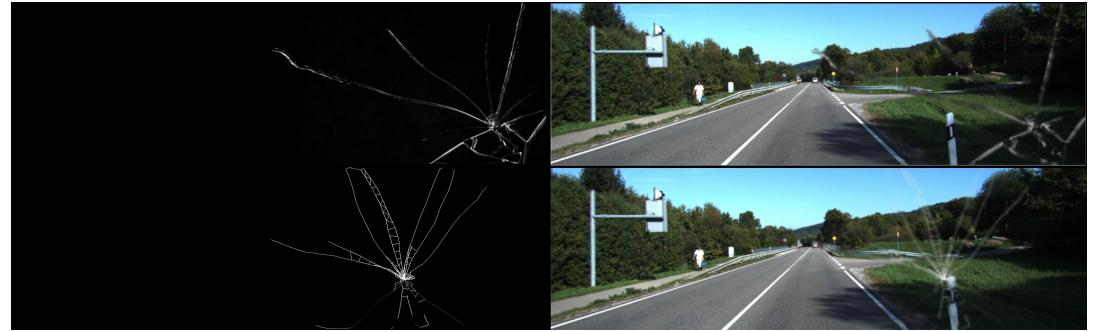


Figure 5 : En haut à gauche - Motif de fissure collecté en ligne sur Freepik ; en haut à droite - motif de fissure en ligne superposé sur KITTI ; en bas à gauche - motif de fissure simulé avec PBR ; en bas à droite - motif de fissure simulé superposé sur KITTI.

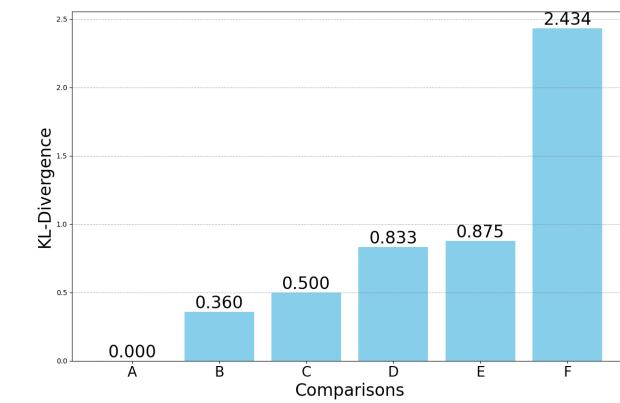


Figure 6 : Divergence K-L de différentes paires de distributions d'images. Jeux de données : RC - Jeu de données réel sur route (voir Fig. 2), KITTI et Chats. Filtres : RO - Réel (collecté en ligne) et Sim - Simulé. Divergence K-L entre (x - relation de superposition) : A - (Sim x KITTI) vs (Sim x KITTI) ; B - (Sim vs RO) ; C - (RC propre vs KITTI) ; D - (RC cassé) vs (RO x KITTI) ; E - (RC cassé) vs (Sim x KITTI) ; F - KITTI vs Chats.

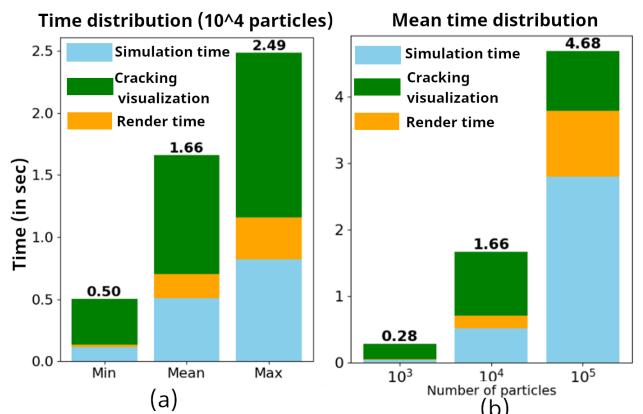


Figure 7 : (a) Temps moyen pris par différents modules de notre pipeline sur 100 exécutions. (b) Le temps minimum, maximum et moyen pris par différents modules sur 100 exécutions pour un maillage de particules 10^4 . Pour ces graphiques, nous présentons le temps pris pour la simulation (temps de simulation), la conversion du maillage en verre (visualisation de la fissuration) et enfin le rendu (temps de rendu).

L'angle d'impact. La visualisation des fissures et le temps de rendu varient également en raison des masques de tailles différentes formés par les motifs de fracture variables. Nous faisons également varier le nombre de particules et observons comment le temps d'exécution augmente de manière exponentielle avec l'augmentation du nombre de particules. Toutes ces exécutions ont été rendues sur des images du jeu de données KITTI avec des dimensions de $(375 \times 1242 \times 3)$.

Conclusion et perspectives futures

Nous avons introduit une nouvelle classe d'échecs adversariaux résultant du processus physique de défaillances dans la caméra. Dans cet article, nous proposons une approche pour générer un motif de verre brisé réaliste à partir d'une simulation physique et l'intégrer ensuite dans des ensembles de données d'images existants en utilisant le rendu basé sur la physique. Nous montrons que les images adversariales simulées peuvent entraîner des erreurs significatives dans la détection d'objets.

Dans ce travail, nous abordons les attaques adversariales en boîte noire provenant de phénomènes physiques réels et naturels, non artificiellement conçus pour exploiter des vulnérabilités spécifiques du modèle.

Nous ne supposons aucune connaissance des attributs, des poids ou de l'architecture du modèle, garantissant que les attaques sont transférables entre divers modèles. Les méthodes adversariales physiques (Patch Translucide, RP2) peuvent toutes être qualifiées d'occlusions soit de la caméra, soit des objets capturés. L'adversarialité provient de l'effet de l'inférence du modèle dû à ces occlusions. Notre pipeline PBR mélange les fissures avec les images sources sous forme de motifs translucides et flous, impactant l'encodage de l'espace latent plutôt que de provoquer une occlusion directe, entraînant des détections incorrectes.

Bien que ce travail introduise une méthode basée sur la physique pour la génération de motifs de verre brisé spécifiquement, les défaillances de la caméra englobent d'autres effets tels que l'éblouissement solaire, la surexposition, la sous-exposition, la condensation, etc. Notre travail futur se concentrera sur la création d'une boîte à outils adversariale pour la génération réaliste de ces effets en utilisant la physique et, par la suite, les placer sur des ensembles de données d'images existantes et des plateformes de simulation de voitures pour promouvoir davantage de recherches dans ce domaine des défaillances partielles de la caméra.

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

Références

- Akhtar, N.; et Mian, A. 2018. Menace des attaques adversariales sur l'apprentissage profond en vision par ordinateur : Une enquête. *Ieee Access*, 6 : 14410–14430.
- Carlini, N.; et Wagner, D. 2017. Les exemples adversariaux ne sont pas facilement détectés : Contournement de dix méthodes de détection. Dans *les Actes du 10e atelier ACM sur l'intelligence artificielle et la sécurité*, 3–14.
- Ceccarelli, A.; et Secci, F. 2022. Défaillances des caméras RGB et leurs effets dans les applications de conduite autonome. *IEEETransactions on Dependable and SecureComputing*, Coulomb, 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.; et Verma, D. 2004. Classification adversariale. Dans *les Actes de la dixième c onférence internationale ACM SIGKDD sur la découverte de connaissances et l'exploration de données*, 99–108. Eykholt, K.; Evtimov, I.; Fernandes, E.; Li, B.; Rahmati, A.; Xiao, C.; Prakash, A.; Kohno, T.; et Song, D. 2018. Attaques physiques robustes sur la classification visuelle par apprentissage profond. Dans *les Actes de la conférence IEEE sur la vision par ordinateur et la reconnaissance de formes*, 1625–1634. Geiger, A.; Lenz, P.; Stiller, C.; et Urtasun, R. 2013. La vision rencontre la robotique : Le jeu de données KITTI. *International Journalof RoboticsResearch(IJRR)*. Goodfellow, I. J.; Shlens, J.; et Szegedy, C. 2014. Expliquer et exploiter les exemples adversariaux. *arXiv preprintarXiv:1412.6572*. Iben, H. N.; et O'Brien, J. F. 2009. Génération de motifs de fissures de surface. *GraphicalModels*, 71(6): 198–208. Jocher, G.; Chaurasia, A.; et Qiu, J. 2023. Ultralytics YOLO. Kong, Z.; Guo, J.; Li, A.; et Liu, C. 2020. Physgan : Génération d'exemples adversariaux résilients dans le monde physique pour la conduite autonome. Dans *les Actes de la conférence IEEE/CVF sur la vision par ordinateur et la reconnaissance de formes*, 14254–14263. Kuna, M. 2013. Éléments finis en mécanique de la rupture. *Solidmechanics andits applications*, 201: 153–192. Kurakin, A.; Goodfellow, I. J.; et Bengio, S. 2018. Exemples adversariaux dans le monde physique. *Dans la sécurité et la sûreté de l'intelligence artificielle*, 99–112. Chapman and Hall/CRC. Li, J.; Schmidt, F.; et Kolter, Z. 2019. Autocollants de caméra adversariaux : Une attaque physique basée sur la caméra contre les systèmes d'apprentissage profond. Dans *la conférence internationale sur l'apprentissage automatique*, 3896–3904. PMLR. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; et Zitnick, C. L. 2014. Microsoft coco : Objets communs dans le contexte. Dans *Computer Vision–ECCV 2014 : 13e Conférence européenne, Zurich, Suisse, 6–12 septembre 2014, Actes, Partie V 13*, 740–755. Springer.
- Remarques sur la terminologie dans le domaine de la supervision, de la détection de défauts et du diagnostic. *Volumes des Actes de l'IFAC*, 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.; et les Contributateurs de SciPy 1.0. 2020. SciPy 1.0 : Algorithmes fondamentaux pour le calcul scientifique en Python. *Nature Methods*, 17 : 261–272. Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; et Shao, L. 2022. Pvt v2 : Améliorations des bases avec le transformateur de vision pyramidale. *Médias visuels computationnels*, 8(3) : 415–424.

Yu, F.; Chen, H.; Wang, X.; Xian, W.; Chen, Y.; Liu, F.; Madhavan, V.; and Darrell, T. 2020. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2636–2645.

Zolfi, A.; Kravchik, M.; Elovici, Y.; and Shabtai, A. 2021. The translucent patch: A physical and universal attack on object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 15232–15241.

Yu, F.; Chen, H.; Wang, X.; Xian, W.; Chen, Y.; Liu, F.; Madhavan, V.; et Darrell, T. 2020. BDD100k : Un ensemble de données de conduite diversifié pour l'apprentissage multitâche hétérogène. Dans les Actes de la conférence IEEE/CVF sur la vision par ordinateur et la reconnaissance de formes, 2636–2645. Zolfi, A.; Kravchik, M.; Elovici, Y.; et Shabtai, A. 2021. Le patch translucide : Une attaque physique et universelle sur les détecteurs d'objets. Dans les Actes de la conférence IEEE/CVF sur la vision par ordinateur et la reconnaissance de formes, 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).

Algorithme de propagation du stress

Algorithme 1 décrit la procédure pour simuler la propagation du stress à travers un matériau suite à un événement d'impact. L'algorithme prend en entrée la localisation de l'impact (pt), l'amplitude de la force d'impact (F), le vecteur de direction de l'impact (v), et l'arête parente (PE) associée au site d'impact. Il utilise également un rayon du plus proche voisin R pour déterminer l'ensemble des emplacements candidats pour la propagation du stress.

Algorithme 1 : Propagation du stress

```

1:  $pt \leftarrow$  Point d'impact
2:  $F \leftarrow$  Force d'impact
3:  $PE \leftarrow$  Arête parente
4:  $v \leftarrow$  Vecteur d'impact
5:  $R \leftarrow$  Rayon du plus proche voisin
6:
7: procédure PROPAGERSTRESS( $Pt, F, V, PE$ )
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:  PROPAGERSTRESS( $Pt, F, V, PE$ )
16: fin de la procédure
```

Tout d'abord, il utilise une structure de données en arbre KD pour interroger efficacement tous les points (frontières) dans un rayon donné R du point d'impact. Pour chaque frontière, il calcule un vecteur directionnel unitaire du point d'impact vers la frontière (NN). Il projette ensuite le vecteur d'impact v sur cette direction pour obtenir la similarité cosinus $\cos(\theta)$, capturant la relation angulaire entre la direction de l'impact et la direction de propagation candidate. Pour chaque candidat, la valeur résultante est utilisée, avec la force d'impact, pour calculer la contrainte correspondante à ce point. L'algorithme sélectionne ensuite le candidat avec la valeur de contrainte maximale. Le vecteur d'impact v et l'arête parente PE sont mis à jour pour correspondre à cette nouvelle direction. Le processus est répété de manière récursive, permettant à l'onde de contrainte simulée de se propager de manière itérative à travers le matériau le long du chemin de transfert de contrainte le plus élevé.

Cette approche vise à imiter la manière dont la contrainte d'un point d'impact est le plus susceptible de se propager à travers un matériau—suivant préférentiellement des chemins définis par la proximité géométrique et l'alignement mécanique avec l'impact initial.

Le résultat final de la simulation est la réalisation du maillage sous forme d'image qui correspond au motif de lentille brisée (image finale de la Fig. 8).

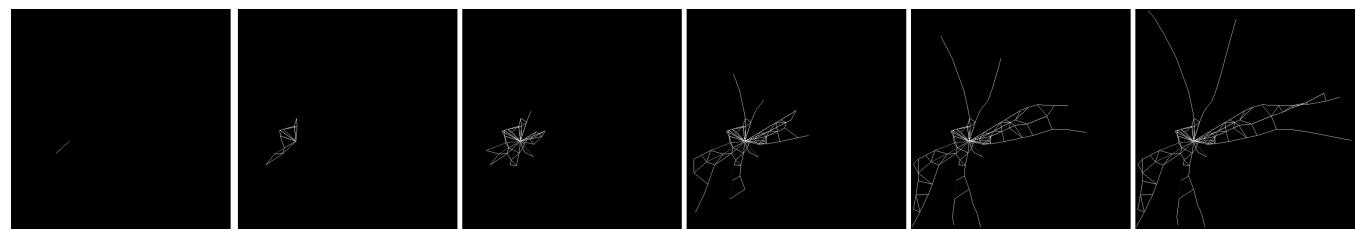
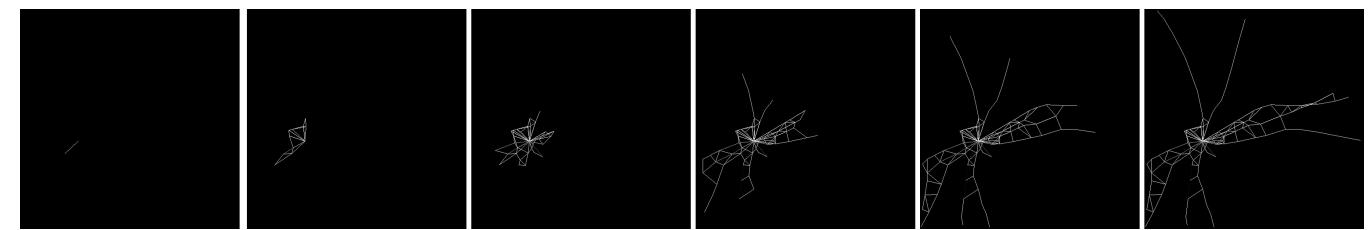


Figure 8: An animation of fracturing of a lens simulated by setting the stress field and applying PBR.



Figu^{re} 8 : Une animation de la fracture d'une lentille simulée en définissant le champ de contrainte et en appliquant PB R.

Static Experiment

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

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

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

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

Increased AP for pedestrians in KITTI

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



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

Expérience Statique

Afin de comprendre l'effet de ces fractures sur les images résultantes, nous menons d'abord une expérience statique en intérieur comme référencé dans la Section Introduction. Nous utilisons diverses plaques de verre trempé pour cette expérience, que nous brisons aléatoirement à l'aide d'un petit marteau avec un ou plusieurs points de rupture. Ensuite, nous plaçons un appareil photo hybride 36 MP JVC GC-PX10 monté sur un trépied avec une pince devant le verre trempé.

La Fig. 9(a) montre l'installation détaillée avec le support de 1 a caméra et le verre trempé maintenu en place avec une pince. La Fig. 9(b) montre l'image capturée par la caméra et la Fig. 9(c) montre le véhicule unique placé comme l'objet principal capturé par la caméra à travers le verre trempé. La scène est éclairée par des lumières fluorescentes au plafond.

La Fig. 10 montre certains des motifs de fractures/rayures s ur le verre trempé. Ces motifs ont été intentionnellement randomisés, en utilisant plusieurs points focaux et différents niveaux de force pour imiter la nature imprévisible et variée des dommages réels sur le verre. En appliquant diverses forces, nous avons pu produire un éventail de fractures et de rayures, allant de fines abrasions de surface à des fractures plus prononcées. Cette approche a été choisie pour reproduire de près les types de dommages que les surfaces en verre peuvent rencontrer dans des conditions réelles—tels que ceux causés par des impacts, des débris ou des facteurs de stress environnementaux—assurant ainsi la pertinence et le réalisme de notre configuration expérimentale. Ces motifs de dommages représentatifs nous permettent d'analyser plus efficacement l'influence des imperfections du verre sur la performance des capteurs et les algorithmes de détection d'objets.

Deux motifs de fracture différents et leurs images résultantes sont présentés dans la Fig. 11 et la Fig. 12. Nous tenons à noter que nous avons considérablement varié les focales des caméras pour comprendre comment les images apparaissent en mise au point proche et lointaine. Les résultats montrent que même des motifs de rayures mineures apparaissent dans l'image de sortie, tandis qu'un motif de multi-fractures beaucoup plus fort peut brouiller presque toute l'image. Cette expérience fournit l'intuition sur laquelle notre cadre de simulation et de visualisation est construit.

Augmentation de l'AP pour les piétons dans KITTI

Nous tenons à souligner que l'augmentation de l'AP pour la classe des piétons était quelque chose qui nous a surpris au départ. Cependant, une analyse approfondie et qualitative nous a aidés à comprendre que cela se produisait en raison des fissures dans le verre, ce qui facilitait la classification des piétons par le modèle grâce à des contours améliorés autour d'eux. Ce n'était pas un artefact de contour, mais plutôt la fissure du verre agissant comme une bordure supplémentaire séparant clairement le piéton de l'arrière-plan. Un résultat similaire a également été observé dans [1] où l'AP global a été augmenté dans les images adversariales.

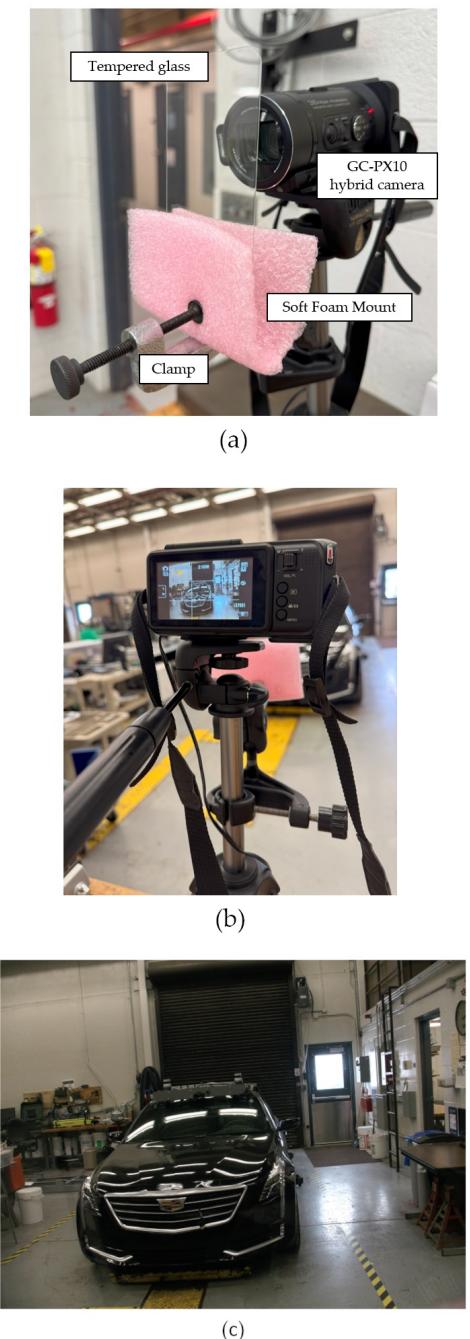


Figure 9 : Configuration expérimentale pour la collecte d'images affectées par des couches extérieures rayées/cassées pour une caméra. (a) montre l'ensemble de la configuration pour prendre des images adversariales. (b) montre la position de la caméra par rapport à la scène capturée. (c) montre la scène capturée par la caméra.



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



Figure 10 : Quelques motifs de fractures/éraflures sur le verre que nous avons utilisé pour collecter les images. (a) Une force vive appliquée perpendiculairement à la surface du verre, produisant des fractures se propageant radialement. (b) et (c) reproduisent un verre avec des éraflures

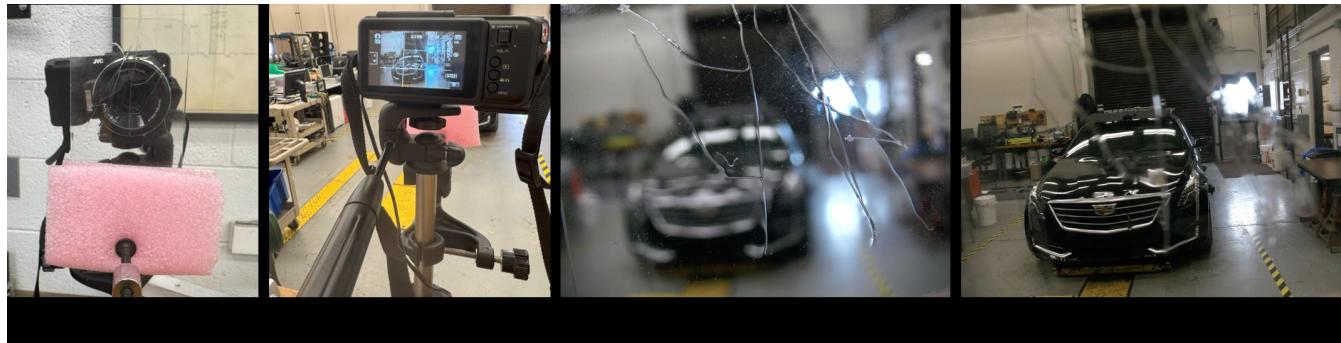


Figure 11: (a) Shows the scratched pattern placed in front of the camera, (b) shows the camera POV. (c) shows the image captured by the camera (short-focus). (d) shows the image captured by the camera (far-focus)

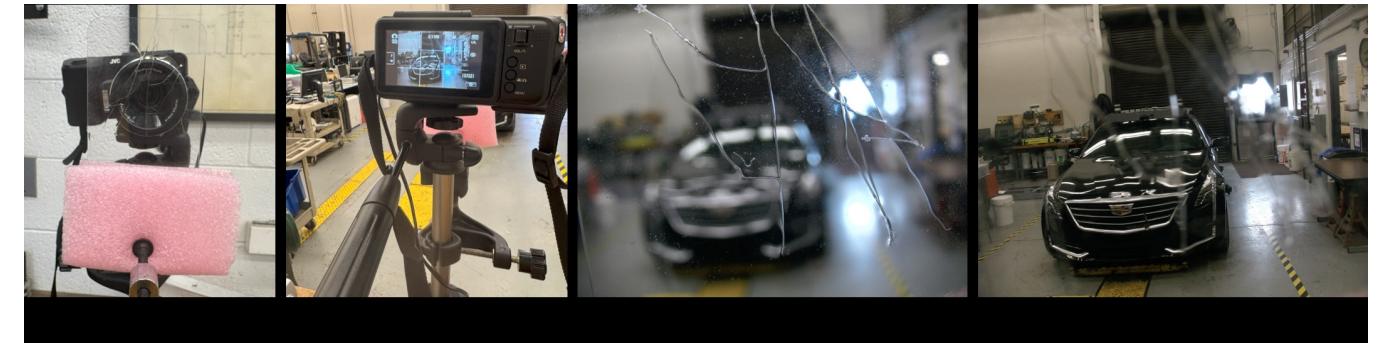


Figure 11 : (a) Montre le motif éraflé placé devant la caméra, (b) montre le point de vue de la caméra. (c) montre l'image capturée par la caméra (mise au point courte). (d) montre l'image capturée par la caméra (mise au point longue)

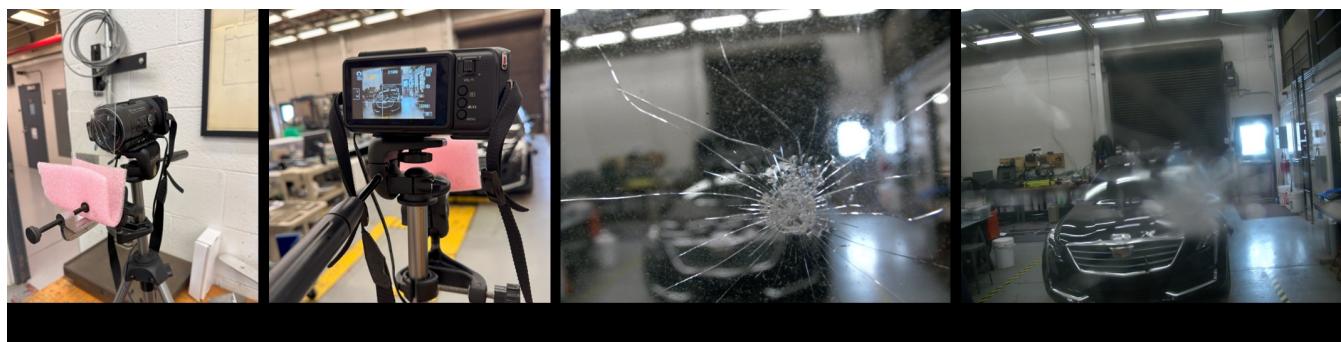


Figure 12: (a) Shows the broken glass pattern in front of the camera, (b) shows the camera POV. (c) shows the image captured by the camera (short-focus). (d) shows the image captured by the camera (far-focus)

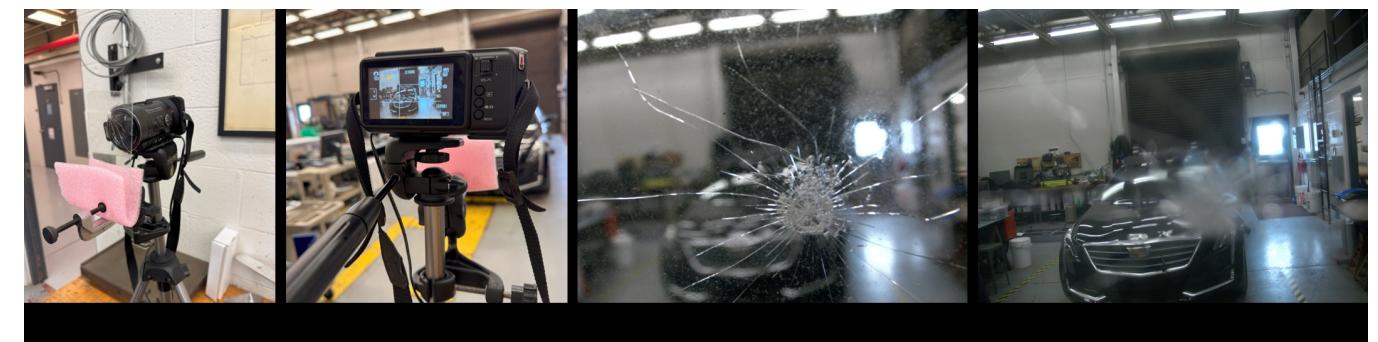


Figure 12 : (a) Montre le motif de verre brisé devant la caméra. (b) montre le point de vue de la caméra. (c) montre l'image capturée par la caméra (mise au point courte). (d) montre l'image capturée par la caméra (mise au point longue)

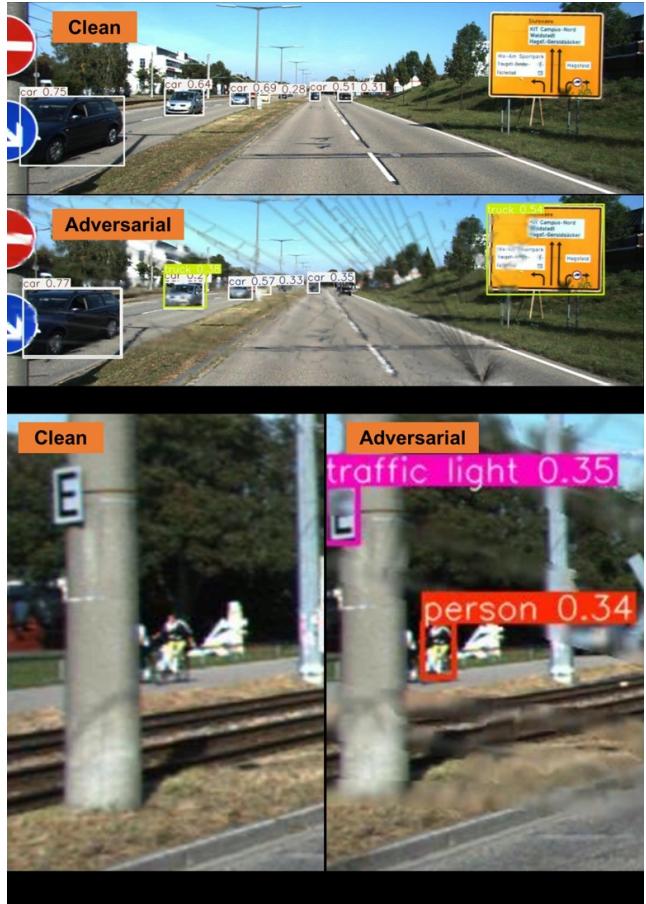


Figure 13: (a) Top - detections on a clean image; bottom - detections on an adversarial image.(b) YoLo fails to detect the person (c) Glass cracks allows the model to detect the person.

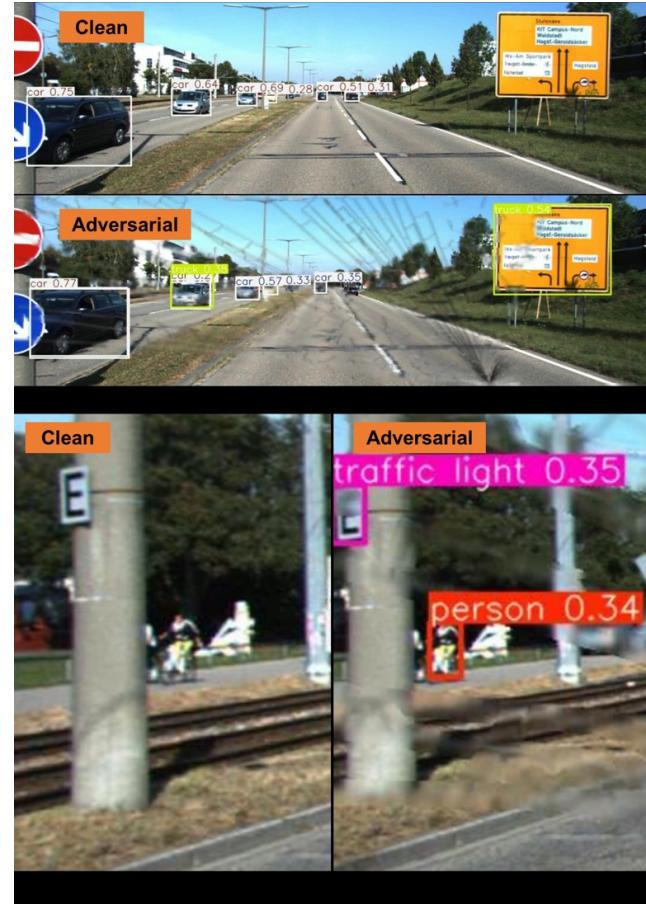


Figure 13 : (a) En haut - détections sur une image propre ; en bas - détections sur une image adversariale. (b) YoLo ne parvient pas à détecter la personne (c) Les fissures du verre permettent au modèle de détecter la personne.

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).

Expérience Dynamique

Dans cette section, nous décrivons l'expérience dynamique mentionnée dans la Section Introduction. Nous réalisons cette expérience pour comprendre la perturbation temporelle introduite par une fissure. Nous utilisons une fissure de pare-brise d'un véhicule et plaçons une petite caméra sur le tableau de bord derrière la fissure. Ensuite, nous photographions deux objets dynamiques - un véhicule et un piéton - alors qu'ils se déplacent dans la scène. La Fig. 14 fournit quelques images spécifiques avec l'inférence de YOLOv8 pour la classe véhicule. Nous montrons qu'avec la fissure, le véhicule reste indétecté dans la plupart des images. De plus, presque chaque image contient un faux positif. En conséquence, nous présentons la Fig. 15 comme les images avec une personne marchant dans la scène. Nous montrons qu'elle fournit par intermittence une détection et parfois avec une mauvaise classe (planche de surf).



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.



Figure 14 : Images spécifiques des cadres pris avec la fissure du pare-brise avec l'inférence YOLOv8 pour la classe de véhicule. A - faux positif sans objet dans la scène ; B - pas d'inférence sur le véhicule ; C - pas d'inférence sur le véhicule ; D - première détection sur le véhicule ; E - deux détections différentes sur le même véhicule ; F - zone de délimitation incorrecte.

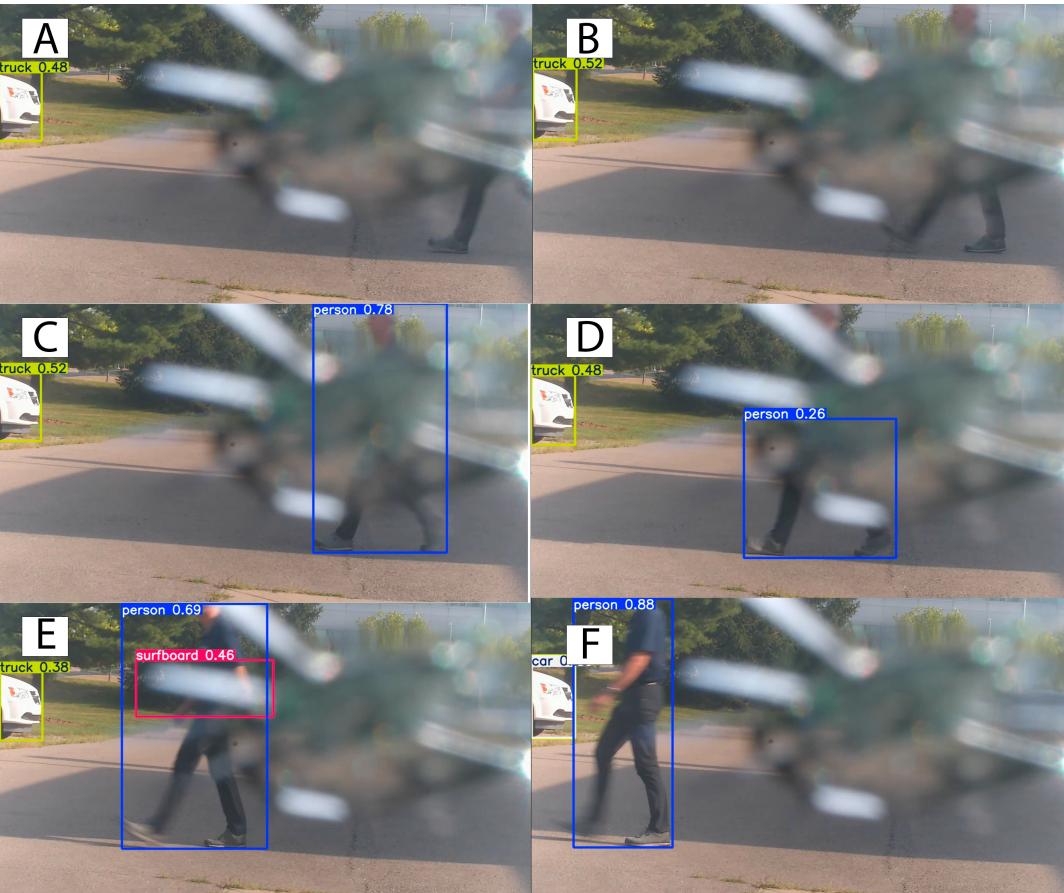


Figure 15: Specific frames of the images taken with the windshield crack with YOLOv8 inference for the person class. A - first entry of person in scene with no detection; B - no inference of person; C - first detection of person; D - partial detection of person; E - detection of person with other class;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

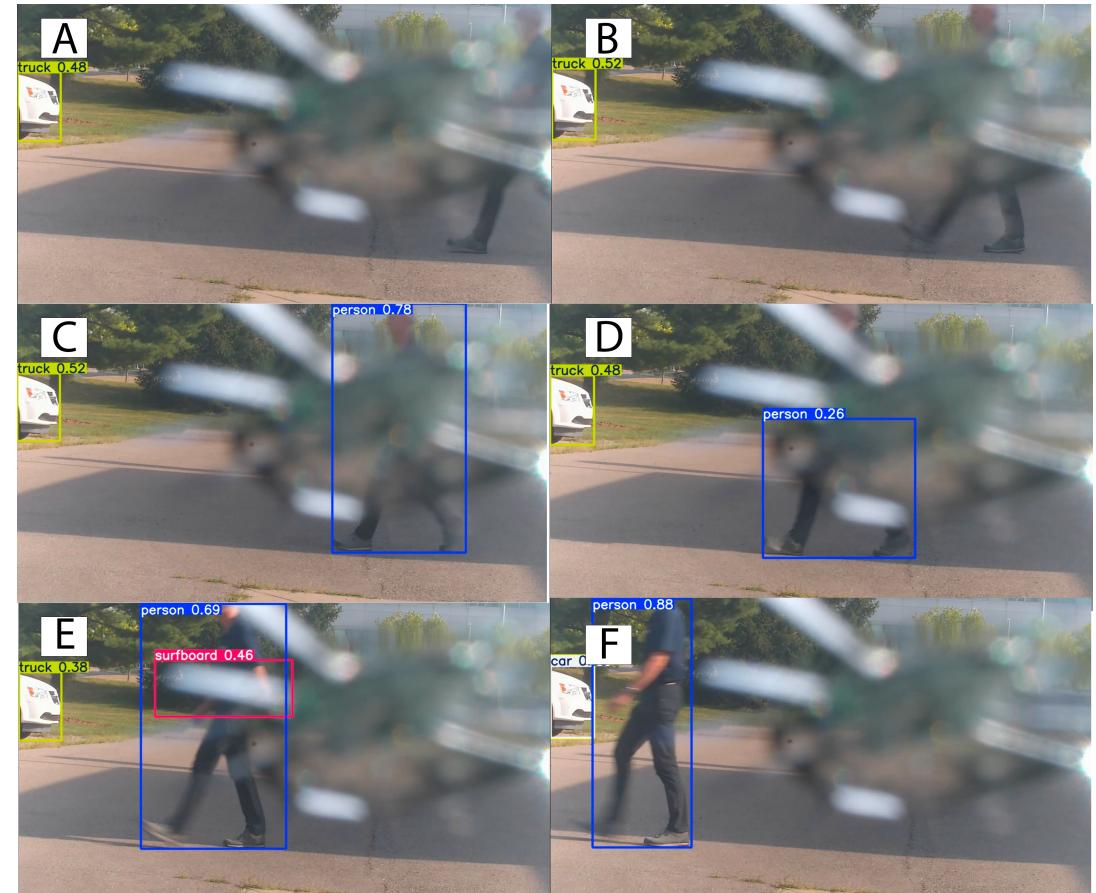


Figure 15 : Images spécifiques prises avec la fissure du pare-brise avec l'inférence YOLOv8 pour la classe personne. A - première entrée de la personne dans la scène sans détection ; B - aucune inférence de la personne ; C - première détection de la personne ; D - détection partielle de la personne ; E - détection de la personne avec une autre classe ; F - détection complète de la personne.

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.

Images réelles de fractures de verre

Nous présentons un exemple d'images de fractures de verre collectées sur le site FreePik, superposées sur le jeu de données KITTI avec l'inférence YOLOv8 (Fig. 16). Nous montrons que la fracture supprime certaines détections et diminue la confiance de détection d'autres.

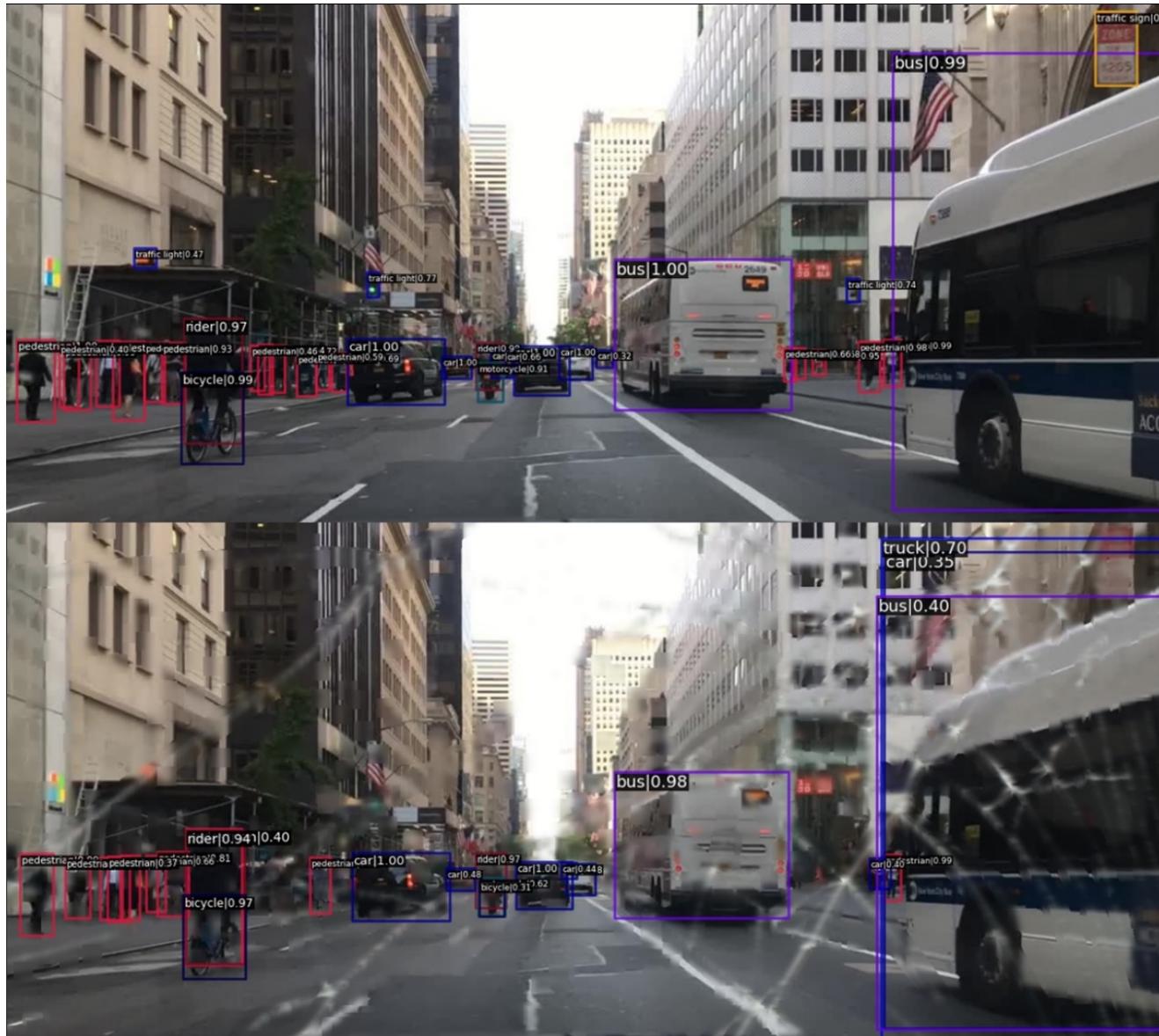


Figure 16: Top - Inference of PTv2 on a clean image from BDD100k. Bottom - Inference for a real broken glass image overlaid on BDD100k for comparison. We see two extra false positives in on the right side (truck, car) and several false negatives for the pedestrian class on the left of the adversarial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

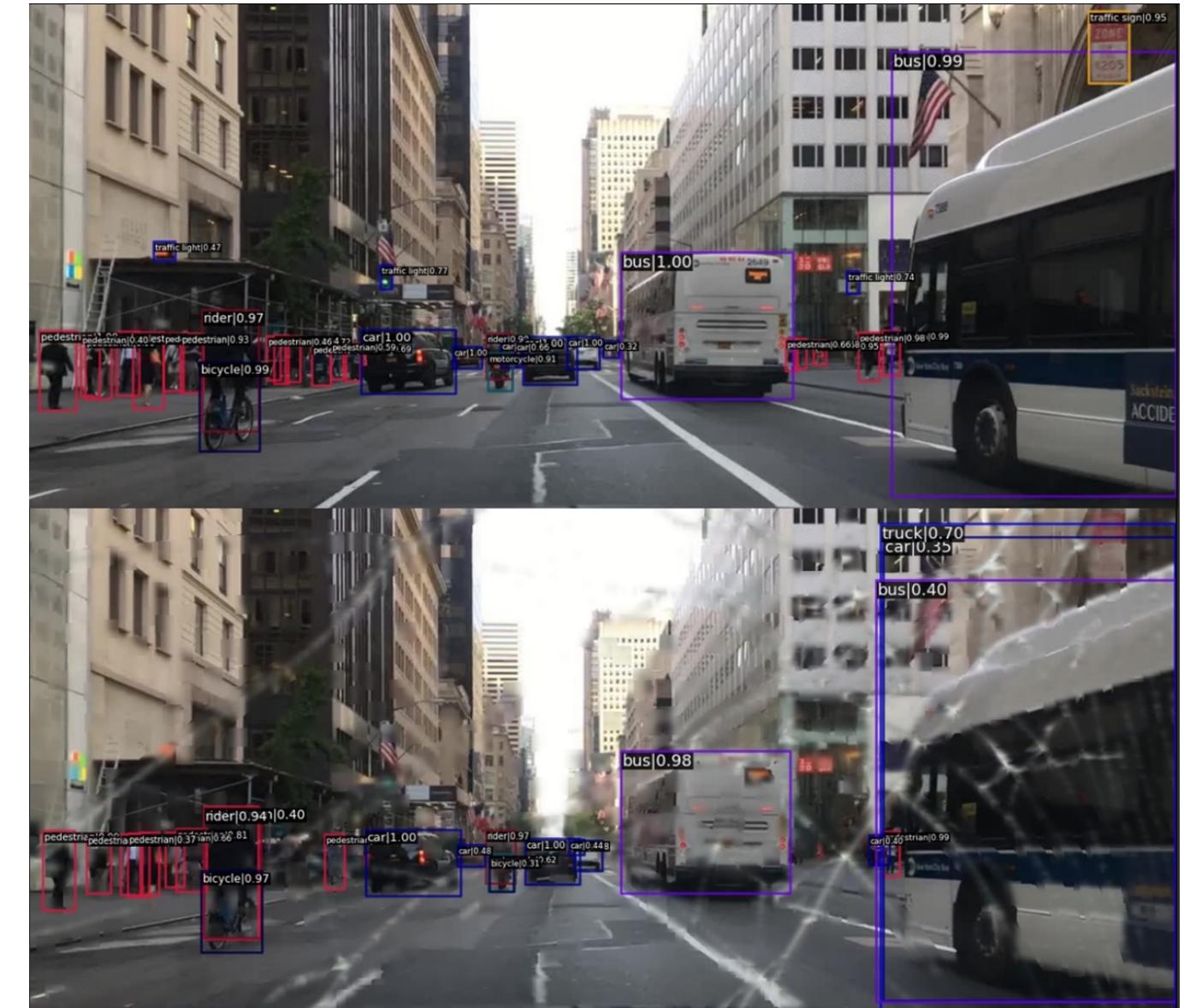


Figure 16 : Haut - Inférence de PTv2 sur une image propre de BDD100k. Bas - Inférence pour une image réelle de verre brisé superposée sur BDD100k pour comparaison. Nous observons deux faux positifs supplémentaires sur le côté droit (camion, voiture) et plusieurs faux négatifs pour la classe piéton sur la gauche de l'image adversariale.