On failures of RGB cameras and their effects in autonomous driving applications

Removed for double blind

Abstract—**RGB cameras are arguably one of the most relevant sensors for autonomous driving applications. It is undeniable that failures of vehicle cameras may compromise the autonomous driving task, possibly leading to unsafe behaviors when images that are subsequently processed by the driving system are altered. To support the definition of safe and robust vehicle architectures and intelligent systems, in this paper we define the failures model of a vehicle camera, together with an analysis of effects and known mitigations. Further, we build a software library for the generation of the corresponding failed images and we feed them to the trained agent of an autonomous driving simulator: the misbehavior of the trained agent allows a better understanding of failures effects and especially of the resulting safety risk.**

Keywords—autonomous driving; RGB camera; failures model; driving simulation; image-based applications

I. Introduction

Autonomous driving is attracting growing attention in recent years, with ever-increasing demand and investments from the industry [17]. The objective of an autonomous driving system is to drive by itself without requiring help from a human: the vehicle detects the environment, locates its position, and operates to get to the specified destination safely.

Sensor technology, data-fusion and inference algorithms as Artificial Intelligence and machine learning (AI/ML) applications are the enabling technologies that play a cornerstone role in autonomous driving systems. These are involved in the majority of the essential tasks for safe driving such as sensor-fusion, environment representation, scene understanding, semantic segmentation, tracking, object detection and recognition [15]. Amongst sensors, the RGB (red, green and blue) camera is acknowledged as the most commonly used and an irreplaceable one [15]. Despite cameras have the known disadvantages of strong sensitivity to external illumination and limited field of view, visual recognition systems are amongst the most solid applications of autonomous driving [33]. Vehicle cameras are already exploited in many applications such as traffic sign recognition, lane detection, obstacle detection, etc. [15], [16], [17]. Additional prospective applications are being researched; for example, at intersections, knowing the location of pedestrians and bicyclists can allow the car to make sophisticated precedence decisions [16]. Also, cameras are amongst the cheapest solutions to build autonomous driving systems that are capable of sensing the surroundings [17].

When the images provided by the camera are degraded, fatal accidents may occur. The trained agents of the AI/ML applications responsible for the elaboration of inputs may rely on biased data and consequently lead to wrong (unsafe) decisions.

A considerable amount of works explored how to make the trained agents robust to artificially crafted or accidentally manipulated input images [35], [36], [37], and how to secure a camera from direct attacks that may disrupt the proper behavior of the camera itself [18], [19]. However, few or no works focused on the attentive identification of a realistic and complete set of accidental modifications of images from a failed RGB camera. To the best of our knowledge, existing works consider a credible set of image alterations but without a systematic analysis of what are the possible malfunctions of a camera and consequently without a shared failures model and software libraries for its reproduction.

We observe that cameras failures and their effects on the overall system should be systematically studied to fully understand the resulting safety threats at system-level, and possible mitigations should be identified carefully. Such study would benefit system and software engineers, both for architecting of system and for robustness assessment of image-based AI/ML applications.

From the above statement, the contribution of this paper are defined as follows: i) we create and discuss a failures model for vehicle cameras in the domain of autonomous driving, by analyzing the different failures, their causes and their effects on the system, ii) we review existing mitigations at component-level, iii) we reproduce the failures effects on the produced image through software; iii) we confirm the effects and discuss the associated safety risks using as reference an application for autonomous driving. Technically, we achieve these contributions respectively through: i) the definition of an FMEA (Failure Modes and Effects Analysis, [14]) on the components of an RGB camera, assuming the camera is placed in a vehicle, and an attentive literature review; ii) the development of a failure library in Python, that we made publicly available in [9], to alter images according to the failures model; and iii) the injection of failures in the frontal camera of a vehicle simulated in the Carla autonomous driving simulator [1], where the trained agent from [3] is running. This last item allows discussing the misbehavior of the trained agent, mainly in terms of number of collisions. Most noteworthy, we prove that even camera failures which slightly alter the image may deceive the trained agent, and in a way dependent on weather conditions.

The rest of the paper is organized as follows. In Section II we present the fundamentals that are at the basis of our work. In Section III we detail the identified failures of camera components, their effects on the output image, and possible mitigations. In Section IV and Section V respectively we implement the failures model in a set of Python modules, and we inject them in the camera of a simulated vehicle, showing their impact and analyzing failures risks. Finally, in Section

VI we discuss related works and in Section VII we summarize the paper contribution and we define our future works.

II. Background Notions

We present background notions that are at the basis of our work. In Section II.A we define the architectural structure of an RGB camera (simply called camera, from now on) that we use as reference in our work, and in Section II.B we describe the FMEA methodology that we apply to identify camera failures.

A. *Architecture of a Camera*

We consider a camera structured in five components (Figure 1): lens, camera body, Bayer filter, image sensor and ISP (Image Signal Processor) [27]. These five components contribute to the creation of the output image.

Lens. Photographic lenses are devices capable of collecting and reproducing an image [7]. The lens is the component that has the greatest impact on the quality of the images. The photographic lens can be composed of one or more lenses and/or reflectors as systems of concave and convex mirrors, often also combined with diopters. The fundamental factor that distinguishes one lens from another is primarily the focal length, which allows dividing them into macro categories. A second factor that characterizes the lens is brightness. A third distinguishing factor are macros: a macro lens has the ability to focus from infinity to 1:1 magnification that is, the size of the image in real life is the same size as it is reproduced on the sensor. Another relevant factor is the focus: this can be manual or automatic. The lens also contains a minimum of electronics, necessary for the focus motor (when automatic) and for zooming [20], and also they may or may not be stabilized.

Camera Body. The camera body is the container of all the electronics of the camera. The Bayer filter, the image filter and the ISP are here contained. Typically, the functions of the camera body are securing the device and protecting inner components from exposure and contact with the outside. For example, the case protects the sensor from light and other possible sources of damage.

Bayer Filter. The Bayer filter (or Bayer pattern) is a scheme for the arrangement of elements sensitive to different colors layered above sensors, that is used for the acquisition of digital images [71]. The photodiodes in an image sensor are color-blind by nature: they can only register shades of gray. To obtain the color in the image, they are covered with different color filters: red, green and blue (RGB) according to the model designated by the Bayer filter. This filter groups the sensors for the three fundamental colors RGB in cells of 2x2 photosites: each cell contains two green elements, one red element and one blue element (Figure 2). Each pixel is filtered to register only one of the three colors: to obtain a color image, various demosaicing algorithms can be used that interpolate a set of complete red, green and blue values for each pixel. These algorithms use the surrounding pixels to estimate the value for a particular pixel [70].

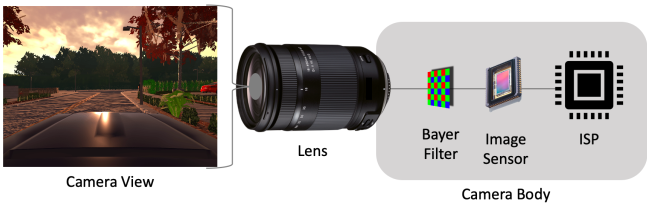


Figure 1. A camera and its components: Lens, and the Camera Body composed of Bayer filter, Image Sensor, Image Signal Processor.

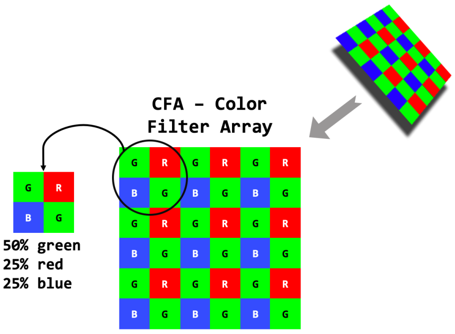


Figure 2. Scheme of a Bayer filter. On the left, a sample cell of 2x2 photosites. In digital imaging, a color filter array (CFA), or color filter mosaic (CFM), is a mosaic of tiny color filters placed over the pixel sensors of an image sensor to capture color information [41].

Image Sensor. The image sensor is the transducer that converts the image into its representation or electrical coding. Essentially, it is a silicon chip capable of capturing and measuring light i.e., the quantity of photons which reach the chip. The sensor generally has a rectangular shape and typically small dimensions, which vary from manufacturer to manufacturer and from model to model. The sensor surface is made up of millions of tiny receivers arranged in a regular grid. These receivers, also called photosites, are the microsensors that carry out the conversion from photons to electrons.

Each individual receiver is able to supply an electrical charge proportional to the number of photons that hit it. The detected charge is then converted by a special analog-digital conversion circuit into a numerical value. Each of the values obtained from the photosites will constitute a pixel of the obtained image. There are currently two types of image sensors on the market: CCD (Charge Coupled Device) and CMOS (Complementary Metal Oxide Semi- conductor). Both are based on the concept of converting the charge of each photosites into a digital format using an ADC (Analog to Digital Converter), but differ in how the information is processed. In fact, for CCD sensors, the information on the charge is taken from the photosites row by row and stored in a register whose content is passed to an amplifier and subsequently to the ADC. After the row has been fully processed, it is eliminated from the exit register and the next row that undergoes the same treatment is loaded. Instead in CMOS sensors, together with photosites a series of transistors have also been integrated which perform an amplification and conversion of the charge into voltage. Using a matrix structure, it is possible to individually select each photosites through its [*row*, *column*] coordinates, and then send the voltage to an ADC that performs the conclusive digital conversion [6], [42].

ISP. The Image Signal Processor is a type of specialized media processor or Digital Signal Processor (DSP), used for image processing in digital cameras as well as other devices [42]. Its functions are multiple and fundamental for the final result after a first acquisition of the image, such as: demosaicing, correction of the image sensor, noise reduction, image sharpness correction, resizing the image, lens distortion correction, chromatic aberration correction, image compression and JPEG encoding, video compression, audio processing / compression / encoding and more [42].

B. Principles of FMEA

The Failure Mode and Effects Analysis (FMEA, [14]) is a widely-used reliability management technique designed to identify potential failures of a component or a process, understand the effects of these failures, assess the risk associated with these failure modes and ultimately classify problems in terms of importance [4]. This failure identification allows to choose and implement corrective actions to address the most serious potential failures. Usually performed analytically, an FMEA is composed of four stages [4], [5]: i) identify all known or potential failure modes of a system; ii) confirm the *causes* and *effects* of each failure; iii) rank the recognized failures by their *risk*, defined as a combination between the probability of occurrence of a failure and the severity of the latter; iv) take remedial actions for the highest risky failures.

The assumption underlying the application of the FMEA is the principle according to which the risk is related not only to the probability that a failure occurs, but also the seriousness of its consequences and ability to avoid or mitigate it. Ultimately, FMEA provides a knowledge base of the failures model and information on corrective actions that can be used as a resource for future troubleshooting activities [4], [5], [14], which matches the objectives of our work and it is the reason why this technique was selected.

III. Analysis of Camera Failures

We consider a frontal camera of a vehicle organized in the five components discussed in Section II.A, and we assume that output images are then processed by image-based AI/ML applications. We exercise the FMEA on the components of the camera. The application of the FMEA identifies the failures model of the camera components, the cause of such failures, and their resulting effect at camera-level i.e., on the output image. Further, we complement this list with a literature review on camera failures, to assure that no relevant failure modes are left out. With respect to the usual analytical application of FMEA, due to the absence of reference data, we could not associate credible ratings on the risk matched to the individual failures (this is in-line with acknowledged limitations of risk ratings in FMEA [4]). However, we mitigate this gap through Section IV and Section V, via simulations of the identified failures.

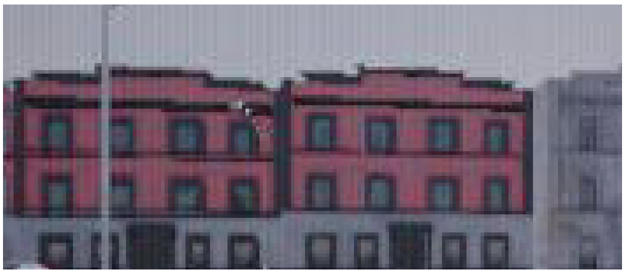


Figure 3. Example of banding: vertical and horizontal lines are introduced

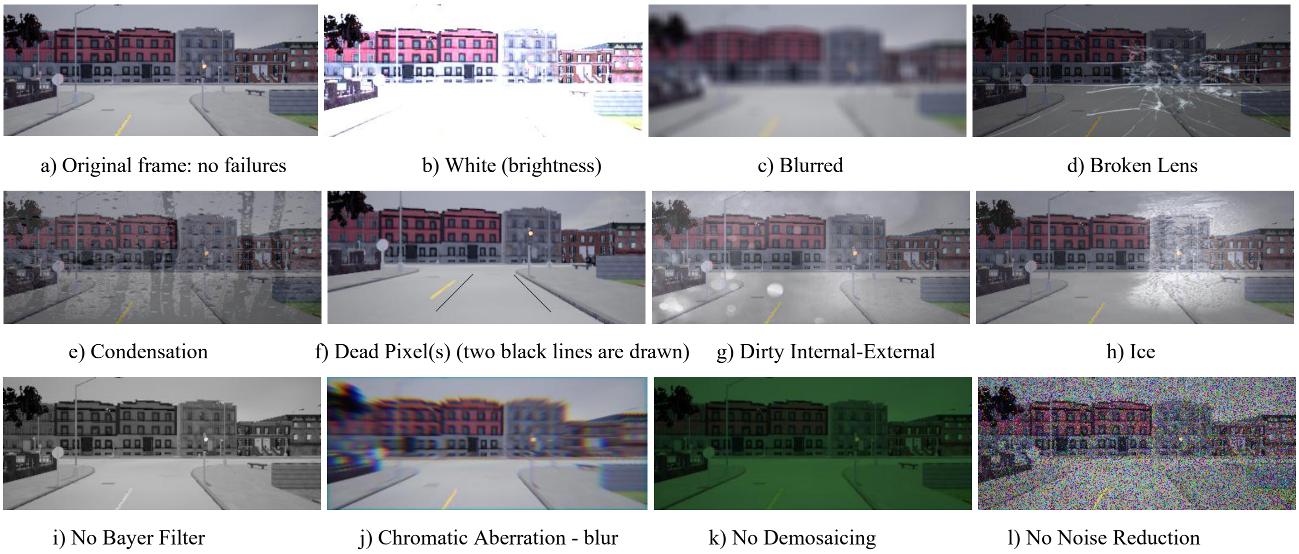
In Section III.A we list the failures in alphabetic order; we assign an evocative name to each failure, that we will use in the rest of the paper. In Section III.B we summarize the failures and we report mitigations from the state of the art.

A. List of identified failures

Banding. In this failure, many parallel horizontal lines in the background are visible in the produced image. The lines are more visible when looking at the darker colors, although they can be also perceived on the lighter ones. Another type of banding can be seen on images where the plain background color degrades, from the point of view of tones, in lighter colors: this is typically called dithering [59]. Figure 3 shows the Banding failure injected into Figure 4a. Only a portion of the banded image is shown, to make the banding effects clearly visible to the naked eye. Banding failures manifest in the image sensor.

Black, White (brightness). These failures represent the brightness alteration, from its minimum (black image) to its maximum (white image) limits, that can happen with the breakdown of fundamental components of the lens such as the shutter, the diaphragm or the iris (Figure 4b). For example, if the shutter has a malfunction that does not allow the entrance of the correct amount of light through the lens, brightness could be altered, from entirely black to entirely white. These brightness failures manifest in the lens component.

Blurred. Blur may occur if the image captured by the device is not in focus (Figure 4c) [69]. Especially in autonomous driving, it is of fundamental importance that the



a) Original frame: no failures
b) White (brightness)
c) Blurred
d) Broken Lens
e) Condensation
f) Dead Pixel(s) (two black lines are drawn)
g) Dirty Internal-External
h) Ice
i) No Bayer Filter
j) Chromatic Aberration - blur
k) No Demosaicing
l) No Noise Reduction

Figure 4. Output frames of the frontal camera for 11 of the different failures of the camera components.
The failures are applied to a frame acquired from the frontal camera of the Carla [1] autonomous driving simulator.

captured images are of good quality and therefore in focus. This is clear if we think that the image-based AI/ML applications make decisions regarding the vehicle's movement, based on the information content of the camera images. Blurred failures manifest in the lens component.

Brackish/Salt-Water. The brackish phenomenon is common in coastal areas and puts a strain on the durability of the materials if not treated with suitable products and/or not maintained over time [6]. The corrosive power of water and air given by the substantial percentage of salt may damage the lens and also the camera body itself, to the extent that external agents may enter the circuits [43]. In the worst scenario, this can affect image acquisition. The image may be altered in various ways; we refer to the description of all the other failures effects for a complete characterization of Brackish/Salt-Water effect. This failure manifests in the lens and camera body.

Bright Lines. This failure is rather rare with the current knowledge and technology. The produced images could show bright vertical and/or horizontal lines, also clearly distinguishable with the naked eye. The cause of these lines is due to the use of LIDARs: this laser technology emits a light intensity (not visible to the human eye) that can seriously damage the camera's image sensor. This failure may manifest in the image sensor.

Broken Lens. One or more internal or external lenses may break, for example because of mechanical stresses due to vehicle jolts or the impact with gravel throw-up by the tires of nearby vehicles. The camera regularly outputs the image, but it will include one additional line (in case of a scratch) or more complicated patterns (Figure 4d), and obviously in case no further components breaks because of the mechanical stress. This failure manifests in the lens component.

Broken VR. In this case, the malfunction of the component that deals with the reduction of vibrations (VR) is considered. This is located in the lens and it is common on many camera models. Its malfunction causes out of focus images: for this reason, the effects of this failure are similar to the Blurred ones. Broken VR manifests in the lens component.

Condensation. When the outside air temperature drops sharply, condensation may appear on the lenses. Condensation, or humidity, degrades the images (Figure 4e) and, if it penetrates inside the device, also the electronical parts [72]. The image is acquired in the correct way, but each image may have heavy changes due to halos on the lenses [72]. If humidity penetrates inside the camera body, it may cause malfunctions and it may also preclude the entire operability of the device. Condensation failure manifests in the lens and camera body.

Dead Pixel. In this failure, the output images have one or more defects of a pixel size. We call this failure Dead Pixel, because it has the same visual effect of the common failure that can be noticed on LCD displays, when a pixel stops working properly and it appears as a black spot on the screen. The single broken pixel may not preclude the good interpretation of the captured images by the AI/ML applications, despite a deliberately modified pixel may actually do so [39]. However, several dead pixels (e.g., a limit case is in Figure 4f) have higher chances to drop the accuracy of the AI/ML application that uses the image. The dead pixel failure manifests in the image sensor.

Dirty Internal - Dirty External. These two failures (Figure 4g) can be discussed together, as they both concern debris of various kinds and sizes (most typically, dust and dirt) which deposits on the internal or external lenses [72]. The most significant difference between the two lenses is that the external dirt can be removed, usually by cleaning the first lens of the objective (i.e., the most external lens). Instead, removing the internal dirt requires more time and, sometimes, specialized personnel [44]. Dirty-related failures manifest in the lens component.

Electrical Overload. The excessive and dangerous temperature increase of the conductors due to an electrical overload could damage, and most likely break, the electronical parts in the camera body. A device with this generic electrical problem may stop working or find itself in a state where images cannot be processed [44]; as effect, the images are produced incorrectly or, most likely, not produced at all. Electrical overload may manifest in the camera body.

Flare. The structure itself of the lenses group creates flare. Flare is due to the reflection of the sun or other light sources on the lenses. The resulting image shows spots of various colors, usually more than one and concentrated on an imaginary line. Obviously, these spots can cover details in the images: the subsequent processing steps, even if the image is captured correctly, will be influenced by the presence of these spots. Flare is difficult to eliminate completely also with the modern technology, that sets a series of lenses slightly spaced one from each other, and each with a specific task. Flare failure manifests in the lens component.

Heat. This type of failure relates to the heat that the lens or camera body can suffer in their operational life. In extreme cases, excessive heat could lead to the evaporation of the lubricating liquids of the moving parts (e.g., zoom). As a result, the use of the zoom (if present) and of other sub-components that are intended to make the image as clear as possible (e.g., the focus tools) may be precluded. As for Brackish/Salt-Water, the image may be altered in various ways and we refer to the description of all the other failures effects for a complete characterization of heat effects. The heat failure manifests in the lens and camera body.

Ice. Ice can be the cause of several camera malfunctions. In fact, this can break the external materials of camera lens and camera body. Furthermore, the external lens can be covered with a blanket of ice that prevents the acquisition of images (Figure 4h). The Ice failure manifests in the lens and the camera body.

No Action. A failed ISP does not respond and therefore the processing of the acquired image does not take place: the image remains in raw format, without any type of processing. As effect, the camera does not transmit anything to the other processing components of the vehicle. This failure manifests in the ISP.

No Bayer Filter. With this failure we want to highlight that, without a Bayer Filter properly functioning, the produced image would result in wrong colors (Figure 4i; wrong colors, in this case, mean that greyscale images are produced). The following phases would unavoidably process chromatically-wrong images. This failure manifests in the Bayer filter.

No Chromatic Aberration Correction - Incomplete Chromatic Aberration Correction. This failure refers to the case that the ISP fails to (fully or partially) apply the removal of chromatic aberration on the acquired image. In optics, axial chromatic aberration is a defect in the formation of the image, due to the different refractive values of the light wavelengths that passes through the optical instrument [67]. It is a defect which may affect all optical lens systems, to varying degrees. This failure leads to images with colored halos on the edges of the subjects: the images show "fringes" of various colors (mostly purple) and a sort of general blur (Figure 4j). This failure manifests in the ISP.

No Demosaicing - Incomplete Demosaicing. In No Demosaicing, we consider the case in which the image is acquired in RAW format (Figure 4k). This means that the demosaicing process has not been carried out and therefore the image is presented with each pixel containing a red, green or blue value. In this case the Bayer array has not yet been interpreted and the image is more pixelated than normal. The prevalence of the greenish hue, also visible in Figure 4k, is due to the high percentage of green photosites (it is double the percentage of the red and blue photosites) in the Bayer filter, as visible in Figure 2. This failure manifests in the ISP.

No Lens Distortion Correction - Incomplete Lens Distortion Correction. This failure may affect only vehicles which mount cameras with wide-angle lenses [21], [73], which tend to deform the image. In fact, with these lenses, the captured image appears as mapped around a sphere that is more protruding towards the observer, in the center of the image. If the conversion to normal proportions (natural symmetric) is not successful, the image may freeze in this processing phase and the system may be in a stalled state. Alternatively, if the output image is incorrectly processed, the AI/ML applications may use images with surrounding objects of distorted proportions and shapes. This failure manifests in the ISP.

No Noise Reduction - Incomplete Noise Reduction. The device captures the image, but during the processing phases (noise reduction) there is an error that prevents the correct removal of the noise e.g., Figure 4l. This failure manifests in the ISP.

No Sharpness - Incomplete Sharpness. In this case, the processing of the captured images fails during the sharpness correction phase. This affects the ability of a camera to identify and define the separation limit between two contiguous areas that have different brightness and/or color (Figure 5, left side). This failure manifests in the ISP.

Rain. It refers to the case in which there are small spots on the camera lens, mostly white, due to the deposit of water drops on the external lens [72] (Figure 5, right side). The elimination of these stains can be considered trivial when the vehicle is parked and without rainfall. However, given the weather variability that a vehicle can encounter, we cannot only consider such simple case. This failure manifests in the lens.

Sand. Because of sand, there may be a possible corrosion of the external sub-components of the lens (percentage of salt in the sand), with the consequent introduction of external agents inside the device, and to the extent that the camera may not capture exact images. In fact, sand could block subcomponents which have the purpose of making the image as clear as possible (tools for focusing, zooming, etc.). The effect on the output image is similar to Dirty Internal-External. This failure manifests in the lens and camera body.

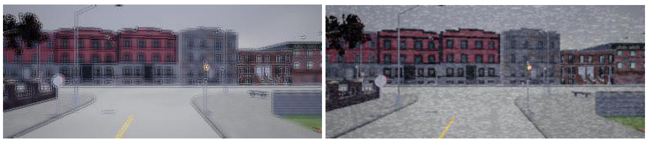


Figure 5. Output frames of the frontal camera for No Sharpness - Incomplete Sharpness (left) and Rain (right) failures.

Spots. This failure occurs when small particles of dust (or other type of material) settle on the Image Sensor. This deposit means that small spots, or shadows, are visible above the light colors on the output image. Such shadows are mostly circular in shape and are very common for amateur photography, particularly when using multiple lenses. In fact, while operating on the camera for maintenance, external agents of imperceptible size may enter and settle on the exposed Image Sensor [74]. The failure effect is similar to the Dirty Internal-External in Figure 4g. This failure manifests in the image sensor.

Water. If water enters the lens or the camera body, the electrical components can fail and most likely no longer acquire images, or acquire them without any content [72]. This failure manifests in the lens and camera body.

Wind. We consider those parts of the component with cavities that, due to the force of the wind (while the vehicle is in motion or parked), could lead to minimal external damage and the subsequent infiltration of various agents inside the camera. The acquisition of images could therefore be incorrect: lenses could move and images shifted or cut, etc. This can be considered very rare for a vehicle camera. This failure manifests in the lens and camera body.

B. *Summary of the Analysis*

Table I recaps the failures model and the involved camera components. An acronym is matched to each failure; it will be used in the rest of the paper. Further, we add a discussion on possible mitigations implementable in the camera.

IV. Failures Injection and Test Campaign

To better understand the effects and the safety risk of the identified failures, we reproduce them in software and we inject them in the vehicle camera of an autonomous driving simulator. All the code we developed is available at [9].

A. *Autonomous Driving with Carla simulator*

We opt for the Open Urban Driving Simulator Carla (Car Learning to Act, [1]) to create a vehicle that is autonomously driving in a town using the camera as its unique input sensor.

Carla has been implemented as an open-source layer over the Unreal Engine 4 (UE4, [2]) to support training, prototyping, and validation of autonomous driving models, including both perception and control. Carla includes urban layouts, several vehicle models, buildings, pedestrians, street signs, etc. The simulation platform supports flexible setup of sensor suites, and in particular we will exploit the camera sensor, that allows acquiring images from the frontal camera of the vehicle at a specified Frame Per Second (FPS) rate. In the fixed-time step simulation mode that we will use, the FPS determines the temporal duration of each simulation step; we set FPS=10 i.e., a simulation step corresponds of 100 milliseconds of simulated time. Carla provides information on the simulated vehicles as position, orientation, speed, acceleration, collisions and traffic violations. Weather conditions and time of day can also be specified.

TABLE I. FAILURES MODEL, ACRONYMS, FAILURES APPORTIONMENT TO THE CAMERA COMPONENTS, AND DISCUSSION ON MITIGATIONS.

Failure name	Acronym	Comp.	Identified mitigations
Black	BLA	LENS	Black, white (and brightness) alterations can be detected easily, but it is difficult to recover the image (aside brightness, which can be corrected to a certain extent at post-processing).
White	WHI		
Blurred	BLUR		Various methods of removing or correcting image blurring exist [45], [46]. For example, Cai et al. [45] formulates the blind blurring as a new joint optimization problem, which simultaneously maximizes the sparsity of the blur kernel and the sparsity of the clear image.
Broken lens	BRLE		A broken lens may be detected through image processing, but it may be difficult to re-create a clean image.
Broken VR	BRVR		Methods similar to blur removal [45], [46] are applicable for this failure.
Dirty external	DIRTY		Image processing solutions can remove localized rain and dirt effects from a single image [48]. For example, physics-based methods [47] can remove dust and dirt effects from digital photographs and videos.
Dirty internal			
Flare	FLA		Lens artifacts like lens flare and ghosting can be reduced or removed at post-processing [49] from a single input image to restore the correct image.
Rain	RAIN		For the mitigation of this failure, we refer to the same observations of DIRTY and to [48].
Condensation	COND	LENS-CAMERA BODY	Various works e.g., [50], [51], [52], address the problem of avoiding or removing condensation inside cameras.
Heat	HEAT		Against desert and very cold temperatures, there are commercial solutions such as the video surveillance camera AXIS Q60-C PTZ [28] that meets the military standard MIL-STD-810G [60].
Sand	SAND		This failure needs to be prevedented with proper casing of the camera.
Ice	ICE		Camera devices to heat lenses [53] prevent or at least reduce condensation of moisture.
Water	WAT		This failure needs to be prevedented with proper casing of the camera.
Brackish/Salt-water	BRAC		Approaches exist specifically to prevent corrosion of the surfaces in contact with seawater, brackish water, or fresh water [54].
Wind	WIND		These failures need to be prevented e.g., with proper casing of the camera, reliable control circuits and reliable sensors.
Electrical overload	ELOVER		
No Bayer filter	NBAYF	BAYER FILTER	
Spots	SPO	SENSOR	Many image processing methods remove blemishes e.g., Zamfir et al. [55] detect blemishes and compute their physical appearance (size, shape, position and transparency) in an image as a function of camera settings, while Steinberg et al. [56] process images to automatically correct dust.
Banding	BAND		There are various ways to reduce visual effects of banding e.g., by applying dithering patterns [59].
Bright lines	BRIGLI		This failure is generally mitigated by the current knowledge and technology, and it is now rare.
Dead pixel	DEAPIX		It is possible to detect dead pixels without complicated mathematic computations, so that the detection method can be implemented in the embedded device [57]. Circuits to correct dead pixels can be fabricated on a single integrated chip [58].
No action	NOACT	ISP	It may be easy to detect this failure at system level, but it is not possible to recover the image.
No chrom. aber-ration correction	NOCHROMAB		Chromatic aberration effects can be reduced e.g., with image processing [22], [67].
No demosaicing	NODEMOS		Several efficient ways for demosaicing exist as [61], [62]; however it is difficult to recover the image in case of demosaicing failure.
No lens distortion correction	NOLENDIS		Lens distortion can be measured and detected [65], and it can be corrected with image processing [64], [66].
No noise reduction	NONOISE		Several solutions to reduce or remove noise and sharpness are available, also in commercial tools. Solutions that operate at sensor-level also exist e.g., [63], [29].
No sharpness correction	NOSHARP		

Amongst the various autonomous driving agents that exist for Carla, we use the trained agent from [3]. Technical details on the trained agent are outside the scope of this paper and are reported in [3]; we introduce only the notions required to illustrate our test plan. Using this trained agent, at each simulation step it is acquired: i) one RGB image from the frontal camera of the vehicle at a resolution of 384×160 pixels, and ii) the current speed from the speed sensor. These values are processed by the trained agent to predict waypoints in the camera coordinates, and then these waypoints are projected into the vehicle's coordinate image [3]. In simpler words, the trained agent "designs" a trajectory composed of five waypoints on the image. From this, a low-level controller is executed that decides the steering angle, the throttle level, and the braking force. Finally, throttle, speed and braking are applied on the vehicle.

We selected this trained agent amongst the various available because: i) the code is compliant with the latest Carla 0.9 release; ii) it uses only the camera as sensing system, while the usage of additional Carla sensors as depth camera or lidar would invalidate the objectives of our simulations; iii) it presents very good performances such that it is, to the best of our knowledge, currently the most performing model for autonomous driving in Carla using only the camera, showing a minimal number of vehicle collisions; and iv) it has easy-to-use suites for benchmarking, data retrieval and data analysis, including recording of videos of the simulations, and a fix to Carla 0.9.6 to support pedestrian crossing. Further, the model was trained with image augmentations following [8], including pixel dropout, blurring, Gaussian noise, and color perturbations which partially overlap with our failures model.

We preferred a self-driving agent over an object recognition agent, that is instead typically used in works discussing images quality [33], [35], [36], [37]. In fact, a self-driving agent allows showing the effect of persistent failures on consequential images and actions, rather than on individual images without a continuous context.

B. *Injection strategy and failure implementation*

Our injection strategy consists of the following actions, performed at each simulation step: i) acquire the output image from the camera; ii) modify the image by injecting the selected failure before the trained agent processes the image; and iii) feed the modified image to the trained agent. This can be realized by modifying the code of the simulator: in a method called *carla_img_to_np*, the simulator extracts the image from the vehicle's camera before sending it to the decision and control parts. In this method, the simulated failures (implemented in Python) were injected.

In Table II we report implementation details of the simulated failures, including the 31 configurations we will simulate. First, it can be observed that some failures have different intensities levels or configurations. For example, the White failure may have different brightness intensities, from the addition of a mild brightness up to the totally white image: consequently, we considered multiple configurations of the White failure (BRIGH1, BRIGH2, WHI in Table II). Further, it should be noted that some failures alter the image in a similar way: this allowed grouping failures that have similar effects on the output image. This applies to the Blurred and the Broken VR failures (we will consider only the first one, from now on), and to the Dirty Internal-Dirty External and Spots failures (again, we will consider only the first one). We ignore failures whose effect is either i) not providing an output image (Electrical Overload, No Action, Water), or ii) not univocally determined (Brackish/Salt-Water, Heat, Sand). We exclude Bright Lines and Wind as we believe they are very rare for vehicles camera. We exclude also Flare as it depends on the sun position and it needs to take into account vehicle movements, requiring environmental data; further, it would overlap with the flare effects already represented in the Carla simulator. Finally, we don't apply the No Lens Distortion because we are not using a wide-angle lens in Carla.

C. Test plan and execution

The test plan is based on the *corl2017* benchmark from [1]. This benchmark is composed of multiple runs in which a target vehicle must reach a destination position B from a starting position A before a timeout expires. The timeout value is the time required to cover the distance from A to B at an average speed of 10 Km/h as in [1], [3]. Re-using the nomenclature from [1], the starting position A and the destination position B are selected such that three *test objectives* are set:

- *Straight road*: Destination position B is located straight ahead of the starting position A.
- *Turn road*: Destination position B is one turn away from the starting position A.

TABLE II. IMPLEMENTATION AND CONFIGURATION OF THE SIMULATED FAILURES.

Failures	Implementation details	Configurations (named as: failure acronym+configuration identifier)
Banding	We use the PIL Image Module [12] and the images in [9]	BAND: we use resize() and blend() methods to superimpose an image (available at [9]) representing banding
Black	We set every pixel black with the Module Image of PIL [12]	BLA: via the PIL *load* method, making each pixel black
White (brightness)	Brightness introduced using the PIL Image and ImageEnhance Modules [12]	BRIGH1: using *ImageEnhance.Brightness*(*img*) with factor 1.5 BRIGH2: using *ImageEnhance.Brightness*(*img*) with factor 2.5 WHI: via the PIL *load* method, we make each pixel white
Blurred (and Broken VR)	Blur introduced using cv2 [11]	BLUR: we set cv2.blur(img,(12,12))
Broken Lens	Failures are simulated using the PIL Image Module [12] and different images of broken lens, dirt, rain drops, condensation, ice on lens available at [9]. These images are superimposed to the frame acquired by the vehicle camera.	BRLE1, BRLE2: we use PIL *resize* and *blend* methods to superimpose two different images of broken lens
Condensation		COND: we use PIL *resize* and *blend* methods to superimpose an image representing condensation on lens
Dirty internal / Dirty external (and Spots)		DIRTY1, DIRTY2: we use PIL *resize* and *blend* methods to superimpose two different images representing dirt on lens.
Ice		ICE1, ICE2: we use PIL *resize* and *blend* methods to superimpose two different transparent images an image representing ice on lens
Rain		RAIN: we use PIL *resize* and *blend* methods to superimpose an image representing water drops on the lens
Dead pixel	We change the color channel and image processing with cv2 [11]	DEAPIX1: we consider a single black pixel introduced at the bottom right of the frame DEAPIX50: we consider 50 black pixels introduced as a grid (5 vertical x 10 horizontal) DEAPIX200: we consider 200 black pixels introduced as a grid (10 vertical x 20 horizontal) DEAPIX1000: we consider 1000 black pixels introduced as a grid (25 vertical x 40 horizontal) DEAPIX-vcl: we consider a single vertical line, composed of adjacent black pixels, in the center of the view of the camera mounted on the vehicle DEAPIX-3l: we consider 3 lines formed by adjacent black pixels of which 2 are horizontal and 1 is vertical DEAPIX-5l: we consider 5 lines formed by adjacent black pixels of which 3 are horizontal and 2 are vertical DEAPIX-10l: we consider 10 lines formed by adjacent black pixels of which 5 are horizontal and 5 are vertical DEAPIX-r: we consider 2 oblique lines that may overlap with the straight track (Figure 3i) DEAPIX-ro: we consider 2 oblique lines as in DEAPIX-r and a small block of black pixels in the center (which may be interpreted as an obstacle)
No Bayer filter	We open and change the color channel of the image with cv2 [11] and convert to a grayscale with PIL Image Module [12]	NBAYF: open with *cv2.imread*(*img*), change color channel with *cv2.cvtColor*(*img*, *cv2.COLOR_BGR2RGB*) and converted with *.convert*('LA') method to obtain the grey-scale image
No chromatic aberration correction	Image processing with PIL Image Module [12], numPy [13] and [68]	NOCHROMAB-b, NOCHROMAB-nb: we introduce the effect of Chromatic Aberration through the code in [68] respectively with and without blur
No demosaicing	We process the image with cv2 [11], and resize with PIL Image Module [12]	NODEMOS: cv2.imread(img) to read image, elaboration, imgOut.resize((h,w), Image.ANTIALIAS) to resize the final image
No noise reduction	Speckle noise is introduced using cv2 [11] and numPy [13]	NONOISE1: in the np.random.normal method we set the three parameters as follows: 0, 0.5, img.size; then np.random.normal(0,0.5,img.size) NONOISE2: in the np.random.normal method we set the three parameters as follows: 0, 1, img.size; then np.random.normal(0,1,img.size)
No sharpness correction	Sharpness is introduced using the PIL Image and ImageEnhance Modules [12]	NOSHARP: the method ImageEnhance.Sharpness(img) is invoked with factor -3.5

TABLE III. DETAILS ON THE RUNS.

Test objective	Number of runs and weather
Straight road	17 runs with weather SUNNY
	17 runs with weather CLOUDYWET
	16 runs with weather HARDRAIN
Turn road	17 runs with weather SUNNY
	17 runs with weather CLOUDYWET
	16 runs with weather HARDRAIN
Navigation	17 runs with weather SUNNY
	17 runs with weather CLOUDYWET
	16 runs with weather HARDRAIN
Total	**150 runs, repeated for the golden runs and the 31 configurations in Table II, 3rd column.**

- *Navigation*: There is no restriction on the location of the destination position *B* relative to the starting position *A*; this results in runs of longer distance and multiple turns.

For each individual run, the success criteria is that the destination *B* is reached before expiration of the timeout. The failure criteria is whenever the vehicle collides or the timeout expires: we modified the *corl2017* benchmark to halt the run whenever a collision occurs, as in our work we prioritize safety over travelled distance.

The target town we select is the Carla Town02, that is a basic town layout with all "T junctions" and it is also the town used for testing trained agent in [3]. Further, we select three different weather conditions, that represent a clear noon, a wet cloudy noon, and an hard rain sunset. We will call these respectively SUNNY, CLOUDYWET and HARDRAIN in what follows. In addition, in each of the runs performed, the town includes exactly 50 vehicles and 30 pedestrians. We always use the same randomization seed so that spawning positions of vehicles and pedestrians are the same in the repeated runs.

The experiments were organized in two phases. In the first phase, we performed golden runs to produce clean data i.e., we execute the simulation runs without introducing any modification in the images captured by the camera. The second phase repeats the same runs of the previous phase, but with the injection of camera failures to each acquired image. Each run begins by setting the failure code within the method mentioned above. After that, the Carla server is started, and consequently also the Carla client that represents the vehicle and produces the run data.

The three test objectives, with the three weather conditions are investigated in 150 runs for the golden runs and for each of the injection of all the failures configurations (see Table III). This number of runs is a compromise between statistical evidence and completion of the experiments in a reasonable time. With a total of 4800 individual runs, the simulated time in Carla is the equivalent of approximately 160 hours of driving. The simulations were executed on an Intel i9-9920X@3.50GHz CPU with Nvidia Quadro RTX 5000 GPU.

V. RESULTS AND DISCUSSION ON FAILURES RISKS

We present and discuss results of our test campaign. All logs and some videos that describe the various runs are available at [10].

A. Collisions and success rate

First, we discuss the impact of each individual failure on the decisions of the trained agent of [3]. Figure 6 and Figure 7 show respectively the success rate and the number of collisions for the three test objectives Straight road, Turn road and Navigation, ordered by the Navigation test objective. Not surprisingly, the golden runs perform the best, with the highest success rate (right side of Figure 6) and the lowest number of collisions (left side of Figure 7: 4 collisions, all under the Navigation test objective). Similar results are achieved by DEAPIX1 (5 collisions) and BAND (6 collisions): differences from the golden runs are small. In fact, it is credible that DEAPIX1 does not affect significantly the trained agent: this failure consists in a single black pixel introduced at the bottom right of the image and, on top of this, the agent was trained with pixel dropouts augmentation. Instead, it is a positive surprise that the trained agent is robust to banding i.e., horizontal and vertical lines overlaid to the image. On the opposite, the failures white (WHI), black (BLA), broken lens (BRLE1), ice (ICE2), and blur (BLUR) are the worst performing. In fact, these failures significantly modify the image. Still, they show some successful runs, and may look especially surprising for the Straight road with White and Black failures. This is simply because the car is moving forward blindly in a straight direction and, if there are no obstacles, the run ends successfully.

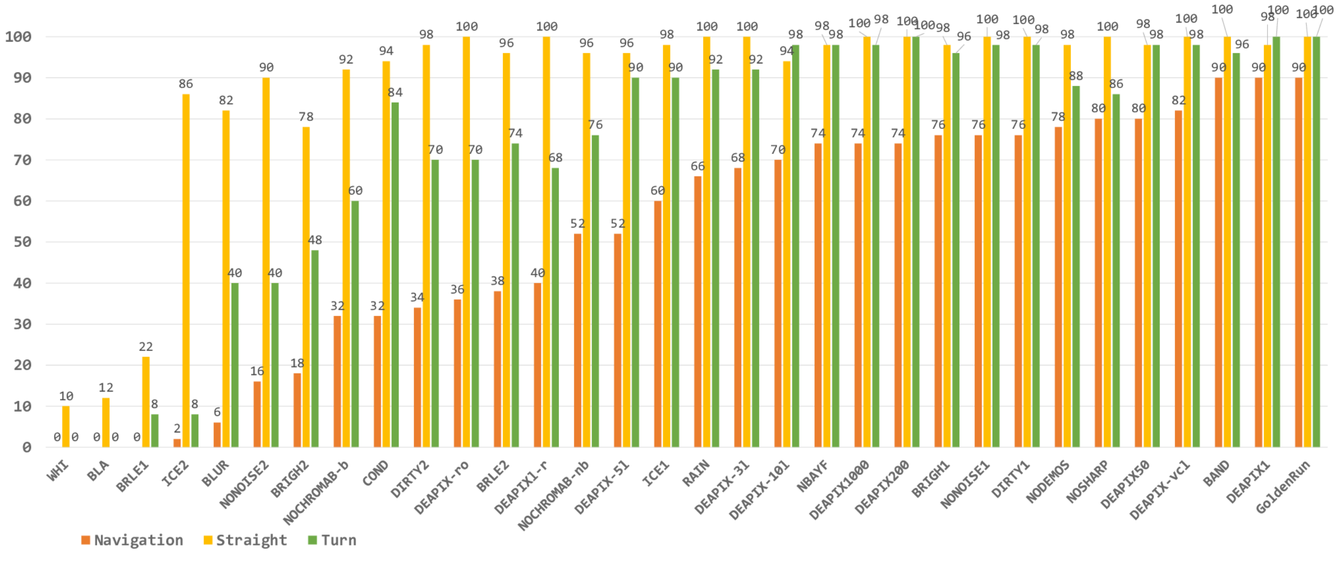


Figure 6 - Success rate of the three test objectives, for each failure configuration and the golden runs.

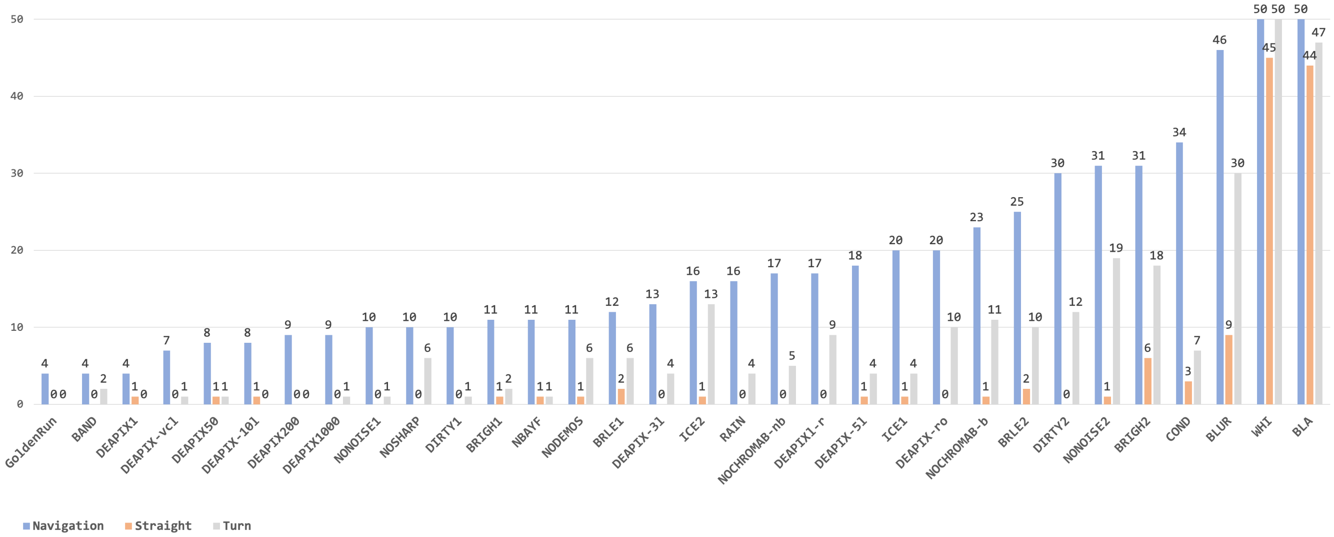


Figure 7 - Number of collisions in the three test objectives, for each failure configuration and the golden runs.

The remaining failures have variable performances, however for each of them the number of collisions is always the highest in Navigation and the lowest in Straight road, with irrelevant exceptions as DEAPIX-101 where there are 0 collisions in Turn road and 1 in Straight road. This trend matches the results of the golden runs and can be easily explained considering the description of the three test objectives.

It should be observed that the ordering of failures in Figure 6 and Figure 7 is not exactly specular. This is because some runs terminate due to the timeout: in general, a timeout occurs when the vehicle is unable to decide how to advance after a particular event e.g., after hard braking to avoid a vehicle in a colliding trajectory. Since this is not leading to collisions, we consider that timeouts occur in case of safe failures and we do not discuss them further.

B. *Weathers and its effects*

Another relevant argument is the impact of the simulated failures under the three different weather conditions, that we discuss with the help of Table IV. First, we note that the golden runs behave the same under the different weather conditions, with 1 collision (out of 17 runs) with SUNNY, 2 collisions (out of 17 runs) with CLOUDYWET, and 1 collision (out of 16 runs) with HARDRAIN.

Table IV shows the five best performing and worst performing failures for each weather. These are similar to the five best and worst failures in Figure 6, with some interesting differences. The Dead Pixel failures are well-mitigated under CLOUDYWET weather, with DEAPIX1000, DEAPIX-10l and DEAPIX-vcl in the best positions, and DEAPIX1 and DEAPIX50 slightly outside the top 5. However, Dead Pixel failures are not this good under SUNNY weather. We hypothesize that the dead pixels with the cloudy sky of CLOUDYWET do not confuse the trained agent.

Another interesting observation is that BRIGH2 with SUNNY performs much worse than in CLOUDYWET and HARDRAIN: the images being acquired in CLOUDYWET and HARDRAIN are rather dark, and adding brightness have reduced negative effects with respect to adding brightness on a SUNNY weather. It is on the opposite behavior ICE1, which shows 3 failures under SUNNY, and the remaining 22 failures with CLOUDYWET and HARDRAIN.

C. *Safety risks and summary of results*

It is evident that our results depend on the target application, and consequently an univocal definition of failure criticalities and risks cannot be devised. However, important concluding observations can be defined. Section V.A brings evidence that the represented failures shouldn't be ignored when building image-based autonomous driving systems and applications. In fact, we show that failures with small visual

TABLE IV - THE FIVE FAILURES LEADING TO THE LEAST AND MOST COLLIDING RUNS UNDER THREE DIFFERENT WEATHERS.

	SUNNY	CLOUDYWET	HARDRAIN
Golden runs	1.96 %	3.92 %	2,08 %
Least colliding runs: percentage of collisions			
SUNNY	1.96 %	DEAPIX1	
		DEAPIX50	
		DIRTY1	
	3.92 %	BAND	
	5.88 %	ICE1	
CLOUDYWET	1.96 %	DEAPIX1000	
		DEAPIX-10l	
		DEAPIX-vcl	
	3.92 %	BAND	
		NBAYF	
HARDRAIN	2.08 %	DEAPIX1	
		DEAPIX200	
	4.17 %	BAND	
		NBAYF	
		NONOISE1	
Most colliding runs: percentage of collisions			
SUNNY	98.04 %	WHI	
	94.12 %	BLA	
	68.63 %	BRIGH2	
	47.06 %	BLUR	
	33.33 %	NONOISE2	
CLOUDYWET	96.08 %	WHI	
	94.12 %	BLA	
	62.75 %	BLUR	
	35.29 %	COND	
	33.33 %	NONOISE2	
HARDRAIN	95.83 %	WHI	
	93.75 %	BLA	
	60.42 %	BLUR	
	35.42 %	NONOISE2	
	27.08 %	COND	

effects affect the decisions of the trained agent. Even small problems, for example the DEAPIX50 failure that scatters 50 dead pixels on a camera with a resolution of 384×160 (above 60.000 pixels), can generate images that deceive the trained agent. Further, from Section V.B we conclude that the safety risk associated to a failure also depends on the environmental conditions: the impact of some failures varies significantly depending on the natural light and the color of the sky.

VI. Related Works

To the best of our knowledge, no research works discuss a complete failures model of a vehicle camera, including an analysis of effects and safety risks, and the provision of a software library. However, several works dealt with similar or inherent problems.

The *performance, robustness, and security of an RGB camera* have been widely explored, however usually focusing on specific elements or target metrics and without addressing the full set of failures. For example, Bijelic et al. [25] present a test and evaluation methodology to compare sensor technologies: the paper shows the difference between an image captured by a standard CMOS camera and one captured by a gated camera. Schops et al. [26], motivated by the limitations of existing multi-view stereo camera benchmarks (a stereo camera has two or more lenses, each with a separate image sensor: this allows the device to simulate binocular human vision and capture three-dimensional images [34]), introduce a new dataset and a technique to minimizes the photometric errors. In [32] a simulation environment is presented which includes the virtual structures of a car designed for autonomous driving tests; typical driving situations have been used to analyze how sensors respond when used in real circumstances as well as to confirm the impacts of environmental conditions. Considering instead sensor security issues, Petit et al. [19] blind a commercial camera system used in commercial vehicles with several light sources. The work shows that leveraging a laser or LED matrix could blind the camera. Similarly, Yan et al. [34] successfully blind the camera by aiming the LED and the laser light at the camera directly: radiating a laser beam against a camera of a vehicle may cause irreversible damage and disrupt the corresponding autonomous applications. In general, it is observed that, because of the vulnerability of the camera caused by its optical characteristics, it is difficult to build a completely secure camera system [18].

In the domain of image-based AI/ML algorithms and applications, many works acknowledge that *the risk of accidental alterations of the output image of the camera is realistic* e.g., [47]. However, this consideration is usually ancillary to the main contribution of the work. Nonetheless, works in the AI/ML domain strongly helped us refining and cross-checking the completeness of our failures model. In fact, chromatic aberration, noise, color temperature, blur and brightness alteration are often considered in image-based AI/ML trained agents, although for the scope of data augmentation during training [30]. For example, Toromanoff et al. [31] present a new convolutional neural networks (CNN) model, in which label augmentation based on translation and rotation allows generating data using only a short-range fisheye (wide angle) camera. Menze et al. [23] elaborate a new model and data set for 3D scene flow estimation, and explicitly take advantage of the background movement caused by the camera mounted on a vehicle. Behzadan et al. [24] show a new deep reinforcement learning framework; in order to develop robust sensors and algorithms, testing under certain meteorological conditions is deemed crucial for determining the impact of bad weather on sensors.

Several other works instead focused on *security and robustness of the trained agents that contribute to the autonomous driving system*, trying to understand the possible modification of camera images that could be produced by an attacker, or to define corner cases. Attackers may maliciously alter the images with transformations that are similar, in concept, to those that could happen with non-malicious failures: also these works were useful when devising our failures model. Most relevant, K. Pei et al. [35] apply input space reduction techniques to transform the image, and can simulate a wide range of real-world distortions, noises, and deformations. W. Wu et al. [36] present a faults model for deep neural networks classifiers, which includes several corner cases based on the alteration of the input image, including amongst the possible causes brightness, camera alignment, and object movements. Finally, evasion attacks consist in modifying the input to a classifier such that it is misclassified, while keeping the modification as small as possible [37]. For example, to create such adversarial images, the correct images can be modified by overlaying carefully crafted noise [38], or altering few selected pixel [39], or with rotation and translation [40]. A representative list of such evasion attacks and their implementations is available at [37].

VII. Conclusions And Future Works

Several works acknowledge the possibility that camera images are accidentally or intentionally modified before they are used by an autonomous driving application. However, to our knowledge, a clear definition of a camera failures model and a software fault library for failures reproduction are still missing. These would benefit software and system engineers that can rely on a reference model to assess robustness of their autonomous driving applications and systems. This paper discussed the failures model identified through the application of an FMEA and literature review on a vehicle camera; the discussion is complemented with the identification of potential mitigations, and with a public library that can be used to reproduce the failures on images sets. Further, we injected such failures in an autonomous driving simulator, to understand their impact on image-based AI/ML applications. Even if results are application-dependent, it is clear that even failures that slightly perturb the image may impact the decisions of a trained agent. Further, it was interesting to observe that the safety risk associated to camera failures depends also on the environmental conditions: for some failures, the number of collisions were closely related to the weather conditions.

We are planning different further works. First, we are defining a failure detector that can identify incorrect images. We are currently training an agent to recognize failed images, such that it can alert the driving system. Second, we are developing an interactive tool to facilitate the usage of the software library. Third, we will exercise such tool on multiple trained agents and automotive datasets: the goal is to compare the effects of camera failures in different image-based autonomous driving applications, including agents trained with data augmentation approaches.

Acknowledgment

Removed for double blind.

REFERENCES

[1] Dosovitskiy, Alexey and Ros, German and Codevilla, Felipe and Lopez, Antonio and Koltun, Vladlen, "CARLA: An Open Urban Driving Simulator", *Conference on Robot Learning*, pp1–16, 2017.

[2] Unreal Engine, www.unrealengine.com [last accessed 20 May 2020]

[3] Chen Dian et al., "Learning by Cheating", *Conference on Robot Learning (CoRL)*, 2019.

[4] Huang, Jia, et al. "Failure mode and effect analysis improvement: A systematic literature review and future research agenda." *Reliability Engineering & System Safety* (2020): 106885.

[5] Bouti, Abdelkader, and Daoud Ait Kadi. "A state-of-the-art review of FMEA/FMECA." *International Journal of reliability, quality and safety engineering* 1.04 (1994): 515-543.

[6] Allen, Elizabeth, and Sophie Triantaphillidou. *The manual of photography*. CRC Press, 2012.

[7] Guy, N. K. *The Lens: A Practical Guide for the Creative Photographer*. Rocky Nook, Inc., 2012

[8] Codevilla, Felipe, et al. "End-to-end driving via conditional imitation learning." *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

[9] *Link to github anonymized for double blind.*

[10] *Link to google drive anonymized for double blind.*

[11] CV2 filtering, https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html (last accessed 21 May 2020)

[12] PIL 3.0, https://pillow.readthedocs.io/en/3.0.x/ (last accessed 21 May 2020)

[13] Numpy, https://numpy.org/ (last accessed 21 May 2020)

[14] Teng, Sheng-Hsien Gary, and Shin-Yann Michael Ho. "Failure mode and effects analysis." *International journal of quality & reliability management* (1996).

[15] Premebida, Cristiano, Gledson Melotti, and Alireza Asvadi. "RGB-D Object Classification for Autonomous Driving Perception." *RGB-D Image Analysis and Processing*. Springer, Cham, 2019. 377-395.

[16] Levinson, Jesse, et al. "Towards fully autonomous driving: Systems and algorithms." *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011.

[17] Lin, Shih-Chieh, et al. "The architectural implications of autonomous driving: Constraints and acceleration." *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems*. 2018.

[18] Ren, Kui, et al. "The Security of Autonomous Driving: Threats, Defenses, and Future Directions." *Proceedings of the IEEE* (2019).

[19] Petit, Jonathan, et al. "Remote attacks on automated vehicles sensors: Experiments on camera and lidar." *Black Hat Europe 11* (2015): 2015.

[20] Hagiwara, Hiroyuki. "Zoom lens and electronic apparatus." U.S. Patent No. 9,541,768. 10 Jan. 2017.

[21] Hughes, C., et al. "Wide-angle camera technology for automotive applications: a review." *IET Intelligent Transport Systems* 3.1 (2009): 19-31.

[22] Chung, Soon-Wook, Byoung-Kwang Kim, and Woo-Jin Song. "Removing chromatic aberration by digital image processing." *Optical Engineering* 49.6 (2010): 067002.

[23] Menze, Moritz, and Andreas Geiger. "Object scene flow for autonomous vehicles." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[24] Behzadan, Vahid, and Arslan Munir. "Adversarial reinforcement learning framework for benchmarking collision avoidance mechanisms in autonomous vehicles." *IEEE Intelligent Transportation Systems Magazine* (2019).

[25] Bijelic, Mario, Tobias Gruber, and Werner Ritter. "Benchmarking image sensors under adverse weather conditions for autonomous driving." *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018.

[26] Schops, Thomas, et al. "A multi-view stereo benchmark with high-resolution images and multi-camera videos." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[27] Phillips, Jonathan B., and Henrik Eliasson. *Camera image quality benchmarking*. John Wiley & Sons, 2018.

[28] AXIS Q60 PTZ, https://www.axis.com/it-it/products/axis-q60-series [last accessed 25 May 2020]

[29] Guo, Longxiang, et al. "Automatic sensor correction of autonomous vehicles by human-vehicle teaching-and-learning." *IEEE Transactions on Vehicular Technology* 67.9 (2018): 8085-8099.

[30] Carlson, Alexandra, et al. "Modeling Camera Effects to Improve Visual Learning from Synthetic Data." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[31] Toromanoff, Marin, et al. "End to end vehicle lateral control using a single fisheye camera." *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018.

[32] Hossain, Sabir, et al. "CAIAS simulator: self-driving vehicle simulator for AI research." *International Conference on Intelligent Computing & Optimization*. Springer, Cham, 2018.

[33] Zheng, Zejia, Xie He, and Juyang Weng. "Approaching camera-based real-world navigation using object recognition." *Procedia Computer Science* 53 (2015): 428-436.

[34] Yan, Chen, Wenyuan Xu, and Jianhao Liu. "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle." *DEF CON* 24.8 (2016): 109.

[35] Pei, Kexin, et al. "Towards practical verification of machine learning: The case of computer vision systems." *arXiv preprint* arXiv:1712.01785 (2017).

[36] Wu, Weibin, et al. "Deep validation: Toward detecting real-world corner cases for deep neural networks." *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2019.

[37] Nicolae, Maria-Irina, et al. "Adversarial Robustness Toolbox v0. 4.0." *arXiv preprint* arXiv:1807.01069 (2018).

[38] Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[39] Vargas, D. V., and Kotyan, S. "Robustness Assessment for Adversarial Machine Learning: Problems, Solutions and a Survey of Current Neural Networks and Defenses. " *arXiv preprint* arXiv:1906.06026 (2019).

[40] Engstrom, Logan, et al. "A rotation and a translation suffice: Fooling cnns with simple transformations." *arXiv preprint* arXiv: 1712.02779 1.2 (2017): 3.

[41] Northrup, T. *Tony Northrup's photography buying guide: How to choose a camera, lens, tripod, flash, & more*, Waterford, Mason Press, 2016.

[42] Poli, P. *Digital photography: Complete Guide*, Apogeo, Milano, 2014.

[43] Townsend, Herbert E. "Outdoor atmospheric corrosion." *ASTM*, 2002.

[44] Elkins, David E. *The camera assistant's manual*. Taylor & Francis, 2013.

[45] Cai, Jian-Feng, et al. "Blind motion deblurring from a single image using sparse approximation." *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009.

[46] Fang, Lu, et al. "Separable kernel for image deblurring." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014.

[47] Gu, Jinwei, et al. "Removing image artifacts due to dirty camera lenses and thin occluders." *ACM Transactions on Graphics (TOG)* 28.5 (2009): 1-10.

[48] Eigen, David, Dilip Krishnan, and Rob Fergus. "Restoring an image taken through a window covered with dirt or rain." *Proceedings of the IEEE international conference on computer vision*. 2013.

[49] Chabert, Floris. *Automated lens flare removal*. Technical report, Stanford University, Department of Electrical Engineering, 2015.

[50] Kondou, Masayoshi. "Condensation prevention camera device." U.S. Patent No. 9,525,809. 20 Dec. 2016.

[51] Loiacono, Dominick F. "Noncondensing security camera housing window assembly." U.S. Patent Application No. 12/433,457.

[52] Englander, Benjamin. "Video camera unit, protective enclosure and power circuit for same, particularly for use in vehicles." U.S. Patent No. 5,455,625. 3 Oct. 1995.

[53] Talbert, Abrams, Milford B. Moore, and Nelson Roy. "Lens heater." U.S. Patent No. 2,442,913. 8 Jun. 1948.

[54] Riffe, William J., and Jack D. Carter. "Method for the prevention of fouling and/or corrosion of structures in seawater, brackish water and/or fresh water." U.S. Patent No. 5,346,598. 13 Sep. 1994.

[55] Zamfir, Adrian, et al. "An optical model of the appearance of blemishes in digital photographs." *Digital Photography III*. Vol. 6502. International Society for Optics and Photonics, 2007.

[56] Steinberg, Eran, Petronel Bigioi, and Adrian Zamfir. "Detection and removal of blemishes in digital images utilizing original images of defocused scenes." U.S. Patent No. 7,295,233. 13 Nov. 2007.

[57] Cho, Chao-Yi, et al. "Real-time photo sensor dead pixel detection for embedded devices." *2011 International Conference on Digital Image Computing: Techniques and Applications*. IEEE, 2011.

[58] Dong, Kimble. "On-chip dead pixel correction in a CMOS imaging sensor." U.S. Patent No. 7,522,200. 21 Apr. 2009.

[59] Wikipedia, "Dithering." https://it.wikipedia.org/ wiki/Dithering [last accessed May 2020].

[60] Barrett, S. *Environmental Engineering Considerations and Laboratory Tests*. MIL-STD-810G, 2008.

[61] B. Grossmann and Y. C. Eldar. "An efficient method for demosaicing" in *2004 23rd IEEE Convention of Electrical and Electronics Engineers in Israel*, pp. 436-439, Sep 2004.

[62] Lilong, S. H. I., and Ilia Ovsiannikov. "Demosaicing rgbz sensor." U.S. Patent Application No. 13/944,859.

[63] Baek, Yeul-Min, et al. "Noise reduction for image signal processor in digital cameras." *2008 International Conference on Convergence and Hybrid Information Technology*. IEEE, 2008.

[64] Yoneyama, Satoru, et al. "Lens distortion correction for digital image correlation by measuring rigid body displacement." *Optical engineering* 45.2 (2006): 023602.

[65] Prescott, B., and G. F. McLean. "Line-based correction of radial lens distortion." *Graphical Models and Image Processing* 59.1 (1997): 39-47.

[66] Sawhney, Harpreet S., and Rakesh Kumar. "True multi-image alignment and its application to mosaicing and lens distortion correction." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.3 (1999): 235-243.

[67] Kang, Sing Bing. "Automatic removal of chromatic aberration from a single image." *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007.

[68] Realistic Lens Blur/Chromatic Aberration Filter, https://github.com/yoonsikp/kromo [last accessed 23 May 2020]

[69] Kim, Tai-hoon, et al., eds. *Computer Applications for Web, Human Computer Interaction, Signal and Image Processing, and Pattern Recognition: International Conferences, SIP, WSE, and ICHCI 2012, Held in Conjunction with GST 2012*, Jeju Island, Korea, November 28-December 2, 2012. Proceedings. Vol. 342. Springer, 2012.

[70] Floyd F. Sabins, Jr., James M. Ellis, *Remote Sensing: Principles, Interpretation, and Applications*, Fourth Edition, Waveland Press, 2020.

[71] Baumgartner, Daniel, et al. "Benchmarks of low-level vision algorithms for DSP, FPGA, and mobile PC processors." *Embedded Computer Vision*. Springer, London, 2009. 101-120.

[72] Sharma, G. *Photography Redefined*, Lulu.com, 2013.

[73] Bhargava, S. C. *Electrical Measuring Instruments and Measurements*, CRC Press, 2012.

[74] Kelby, S. *Scott Kelby's Digital Photography Boxed Set, Parts 1, 2, 3, 4, and 5*, Peachpit Press, 2014.