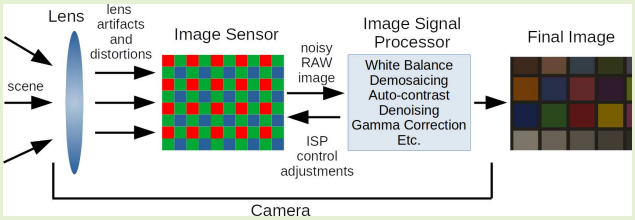
Modeling Cameras for Autonomous Vehicle and Robot Simulation: An Overview

Asher Elmquist and Dan Negrut

***Abstract*—Simulation is increasingly important in the development and testing of robots and autonomous vehicles as it opens the door for candidate navigation, perception, and sensor fusion algorithms to be expeditiously probed in complex and safety-critical scenarios. As most robots and autonomous vehicles make heavy use of cameras to perceive their surroundings, camera modeling becomes a prerequisite for the successful simulation of these autonomous agents. This contribution outlines the context in which camera models are used; provides a component-by-component analysis of the image acquisition pipeline along with algorithms used for its modeling; and closes with a discussion of data-driven approaches that embrace a different perspective on camera modeling.**



***Index Terms*—Simulation, imaging sensors, automotive, robotics.**

I. Introduction

AUTONOMOUS vehicles and robots are poised to have an impact in multiple fields, e.g., transportation, health care, disaster relief, farming, planetary exploration, etc. Since the use of autonomous agents (AAs) permeates safety critical applications where risk to human life is manifest [1]–[3], the idea of improving AA designs through simulation is very attractive [4]. Thus, it is preferred for a design to be proven weak in simulation rather than in reality, since failures identified in simulation come with no human injury/death and/or property loss. Setting aside the difficult issue of simulation-to-reality transferability [5], which falls outside the scope of this contribution, validated and accurate simulation provides the means for efficient and versatile evaluation of novel AA designs, see, for instance, [6]. The expectation is that the control decisions reached in the real world, along with the ensuing dynamics, are close or identical to the ones observed in simulation when the AA model is exercised in a digital twin of the real world scenario. If the simulation-to-reality gap is addressed, simulation enables an expeditious, cost effective, safe, and thorough approach to improving AA design [7].

Manuscript received August 19, 2021; accepted September 24, 2021. Date of publication October 8, 2021; date of current version November 12, 2021. This work was supported in part by the Safety Research using Simulation (SAFER-SIM) Program by a Grant from the U.S. Department of Transportation's University Transportation Centers Program under Grant 69A3551747131 and in part by the National Science Foundation under Grant CPS1739869. The associate editor coordinating the review of this article and approving it for publication was Prof. Huang Chen Lee. *(Corresponding author: Asher Elmquist.)*

The authors are with the Department of Mechanical Engineering, University of Wisconsin-Madison, Madison, WI 53706 USA (e-mail: amelmquist@wisc.edu; negrut@wisc.edu).

Digital Object Identifier 10.1109/JSEN.2021.3118952

As perception informs, downstream, the tasks of planning and control, camera modeling is a prerequisite for the successful simulation of AAs [8]. Against this backdrop, the goal is to provide an overview of the foundations of modeling cameras for their use in AA simulation. Throughout this manuscript, the term "autonomous agents" will be used to refer to robots and autonomous vehicles, including automated vehicles and advanced driver-assistance systems (ADAS). To focus this manuscript, discussion will be limited to RGB mono-cameras due their ubiquity in robot and autonomous vehicle applications, but it should be noted that infrared (IR) cameras, neuromorphic (event) cameras, polarized cameras, and stereo cameras are all important to the application of AAs. Furthermore, the discussion is constrained to CMOS image sensors, as CMOS is the technology of choice in this field.

The task of camera modeling is complicated by two facts: (i) the model is tightly intertwined with that of the virtual world that needs to be sensed; and, (ii) model development requires an understanding of the hardware pipeline and image processing algorithms encountered in the physical camera. In relation to (ii), it is likely that the task of camera modeling will increasingly embrace data-driven approaches, which eschew many of the difficulties faced by physics-based modeling approaches. We recognize this by including a discussion of data-driven approaches that are used in several capacities – from augmenting the physics-based approach to replacing stages of the sensing pipeline, or even the entire pipeline.

To further contextualize the application of camera modeling and its connection to the virtual environment, a primary focus of this paper will be on closed-loop operation, where a camera model is placed in the AA simulation with the

expectation that it produces images of the virtual environment near, or faster than, real time. A secondary camera model use-case of interest is that of synthetic dataset generation. Indeed, extensive amounts of automatically labeled training data can be generated in simulation and subsequently used in offline training, tuning, and evaluation of perception networks in machine learning. Since the synthetic data is generated for offline use, the benefit of real-time or better performance is reduced, which opens the door for more sophisticated sensor models. This manuscript touches on offline approaches as well, which despite being slower can point to future directions of improvement for closed-loop solutions. For clarity, this contribution will use the term "camera model" to refer to the mathematical approximations of the physical phenomena and induced artifacts of the camera. The term "camera simulation" will be used to refer to the time-evolution of the aforementioned model and the process by which synthetic images are produced.

In order to motivate the need for camera models/simulation, this contribution starts off with a short overview of how the models are used. Subsequently, Section §III outlines the physics of camera sensing, with the discussion touching on both internal and external components of the camera. Modeling techniques that target internal camera components are reviewed as follows: Section §IV concentrates on pre-sensor distortion; Section §V focuses on sensor noise; Section §VI is dedicated to post-sensor distortion. External factors in camera simulation, including virtual environment considerations, are discussed in Section §VII. Section §VIII highlights state-of-the-art approaches for replacing physics-based modeling with machine learning-enabled data-driven models. Section §IX summarizes the relevant software solutions for AA camera simulation, and touches on validation efforts and use cases.

II. Preamble; Camera Simulation

The level of detail at which a camera must be modeled is tightly coupled with the application for which synthetic images are to be produced. For example, if the simulation is used for offline training in hazardous weather conditions, then it would be important to sacrifice computation performance for weather realism. Or if the task is to test pedestrian recognition systems with hardware in the loop in all lighting conditions, then modeling the camera's auto-exposure must be done with high-performant algorithms even if model fidelity is lowered. With this in mind, AA camera simulation is primarily used for two general purposes: closed-loop evaluation, and offline training. The former refers to software- or hardware-in-the-loop simulation that embeds the control stack into the simulation loop such that the navigation algorithms are tested while being oblivious to the fact that they are flexed in simulation rather than a real-world scenario. In offline training, camera simulation is used to generate synthetic datasets for training perception algorithms. The ability to automatically label segmented images from simulation mitigates manual labeling costs and increases throughput. These two application areas are briefly discussed next to contextualize choices made in camera model design.

A. Closed-Loop Simulation for AAs

For closed-loop simulation, the focus is on testing, designing, and validating an AA control stack. To that end, the control stack is exercised in conditions that resemble real-world operating scenarios. What is being monitored in simulation may be outcomes such as lane keeping ability, grasping ability of a robotic arm, or obstacle avoidance. In these circumstances, temporal consistency in camera simulation is critical. For example, the camera simulator should not have pedestrians or objects flickering in and out of existence, or moving unrealistically around the scene. This temporal importance is discussed further in [9].

By-and-large, camera simulators for closed-loop testing fall into two categories: either the model uses a game engine graphics pipeline, e.g. [10]–[12]; or the model implements its own camera pipeline [13]–[15]. Using a gaming engine is convenient; however, accuracy might be inadequate, as is the ability to add new features or improve the accuracy of the model. This is because the video gaming graphics pipeline is developed under tight real-time constraints for *human* consumption; using gaming graphics for camera simulation is serendipitous and assessing its adequacy for AA simulation remains an open question whose answer is likely problem dependent.

B. Simulation for Training Perception Algorithms

A second use of camera simulation is tied to developing synthetic datasets for training. The idea is to use camera simulation to generate synthetic images of never-before-seen scenarios or objects. This alleviates the burden of data collection and image labeling, since simulation frameworks can automatically label the synthetic data. In this use case, real-world datasets are replaced or augmented with synthetic images; the end goal is to improve the perception software stack of an AA. Since the datasets are created offline, lower simulation speed is acceptable as long as image fidelity is high. High image realism along with dataset bias and variance are the main issues of concern [16].

Generating synthetic training data has been documented, e.g., [17]–[21], and has met varying levels of success. One prevailing tendency is the occurrence of downstream issues tied to spatial and/or temporal coherence artifacts [9]. Only rarely are these synthetic data sets benchmarked to evaluate the realism of the data synthesized, e.g., realism of the scene, including where and how objects are placed. Even more rare is an evaluation of the quality of sensing done on these synthetic images. One of the reasons for this is that how to quantify the concept of sensor realism is still an open question.

Synthetic image generation has not been exclusively tied to models built around the physics-based simulation of the processes associated with image sensing. Recently, machine learning (ML) techniques have been used to generate synthetic datasets. These methods use data-driven approaches to remove the burden of modeling some or all of the processes taking place in the camera. An image stylizing approach was used in [22] to augment a training dataset, producing augmented synthetic samples. A generative adversarial network (GAN)

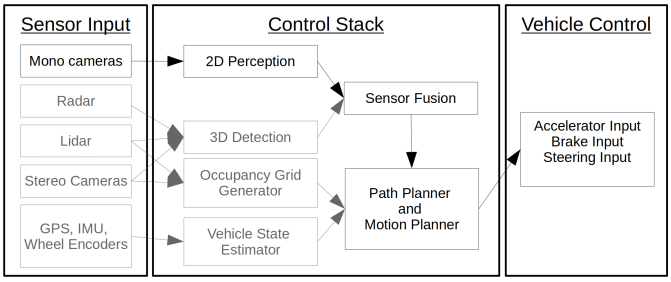


Fig. 1. Illustration of the task-flow of an AA control stack (adapted from [26]). Control stack implementation may vary depending on the application.

(a) Semantic Segmentation from BDD100K dataset [28].

(b) Object Detection from BDD100K dataset [28].

Fig. 2. Semantic segmentation and object detection which are the primary downstream processes of camera simulation. Examples are from the BDD100K driving dataset [28].

was used in [23] to generate synthetic data for learning traffic sign detection. A translation of sign appearance was also proposed in [24] in order to generate additional data for countries whose signs appear infrequently in training sets. In line with expanding datasets, [25] proposed a method for ML-based generation of images from a single image of an object and its 3D geometrical representation. This was shown to have positive effects when training object detection networks since it was able to qualitatively retain object edges. Many of these methods however are not directly effective for closed-loop simulation owing to long computation times.

C. Autonomous Agent Perception

Whether the simulation is used for training perception or evaluating closed-loop behavior, the direct downstream process of camera simulation is perception, as shown in Fig. 1. Camera-based AA perception includes semantic segmentation, object detection, and visual odometry.

Semantic segmentation is the process of pixel-wise labeling of an entire image as shown in Fig. 2a. Object detection, while closely related to segmentation, is the detection, classification, and estimation of specific categories of objects. For instance, the output of object detection could be the size, location, orientation, and confidence level when identifying pedestrians in an image. An example of this is shown in Fig. 2b. While less common, visual odometry is the use of images to determine an ego-vehicle's position, location, and velocity in space. This is detailed further in [27].

By and large, the technology of choice for perception is machine learning, specifically deep learning and convolutional neural networks (CNN) due to their ability to associate regions of pixels and detect high-dimensional features such as edges

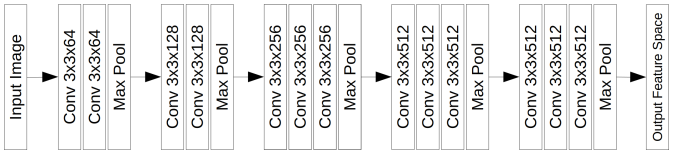


Fig. 3. Illustration of VGG16 architecture [29] where all convolution layers have ReLU activation. When used in segmentation and detection, the network is truncated to exclude the fully connected layers that would exist in the full VGG network.

and blobs. Many of these CNNs have a similar initial structure whether it is applied to segmentation or recognition. This structure is composed of stacked convolutional layers as shown in Fig. 3. The most common structure are VGG [29] variants such as VGG-16. This VGG network maps the RGB image into a high-dimensional feature space that is then processed by segmentation or detection-specific networks that compute the final outputs, i.e., pixel label or bounding boxes. An example of this initial CNN architecture is illustrated in Fig. 3. The mapping of RGB image to higher-dimensional space means specific distortion and artifacts will play a key role depending on the network's sensitivity to the artifact. For example, object edge blur could change the magnitude of a machine-learned edge-detector and make it more susceptible to missing the object or, more likely, cause error in a bounding box prediction as the exact edge may be unknown. While an in-depth study of CNN architectures is out of the scope of this paper, further information on the subject can be found in the following representative work: YOLO [30], Faster-RCNN [31], SSD [32], Mask-RCNN [33], FCN [34], and DeepLab [35].

III. Camera Pipeline Components

In a physics-based camera model, the process of generating synthetic images when the camera is immersed in a virtual environment calls for an understanding of the process by which camera sensors produce data in the physical world. The camera pipeline can be broken down into three distinct modules: i) the optical systems, ii) the image sensor, iii) the image signal processor (ISP). These are illustrated in Fig. 4. In the optical system (module (i)), the light is focused by a complex set of lenses. The intensity of the focused light is then measured by the image sensor (module (ii)) and are translated into electrical signals. These signals are then passed through the ISP (module (iii)), which converts the RAW array into an RGB image. Additionally, the ISP can adjust the light acquisition via auto exposure. Depending on the camera, manufacturer, and application, there are variations in the pipeline owing to factors such as multiple lenses, anti-reflective coatings, or adjustable parameters associated with the ISP.

Each stage of the camera sensing pipeline, i.e., lens, sensor, or ISP, distorts the final image from the ground truth. In the lens, optical distortions change how the camera perceives reality. The sensor array, which measures the incident light, introduces pixel noise. The ISP, which seeks to adjust and improve the image characteristics, can change the magnitude and distribution of the noise, adjust intensity levels across the image, compress and encode the final output, and deblur and denoise the image. The distortions and artifacts that correspond

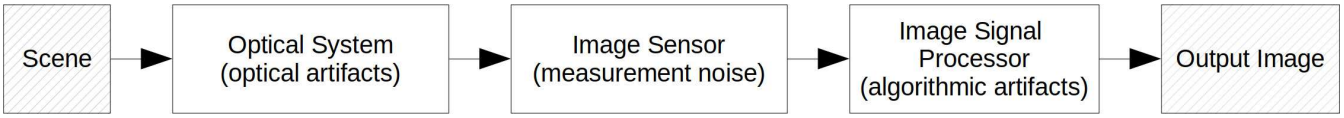


Fig. 4. Each step in the imaging pipeline is responsible for distortion, noise, or artifacts. Optical distortion in the lens, noise introduced by the sensor, and artifacts from post-processing the sensed image can significantly change the output image. Hashed boxes represent modules outside of the physical camera.

to each step in the pipeline are summarized in Fig. 4. These issues are further discussed in [36].

Beyond capturing the inner workings of the camera sensor, two other aspects come into play in camera simulation: 1) the virtual environment; and 2) the downstream application that will process and interpret the image. In regard to (1), satisfactory virtual sensing requires a suitably defined virtual environment. Too simple, and the virtual world resembles so little of the real world that the simulation ceases to serve a useful purpose no matter how sophisticated the camera model. Too complex, and the handling of details becomes unnecessarily burdensome. Owing to strong impetus provided by the video gaming industry, many computer graphics advances have been made specifically to represent sophisticated, feature-rich virtual worlds. In its agency, the camera model is expected to handle both virtual world assets and scene effects. Regarding (2), the synthetic data should be of a quality suitable for use in tasks such as offline training or closed-loop evaluation of obstacle detection, image segmentation, and visual odometry. As such, downstream use dictates the required level of fidelity of all aspects of the camera model.

IV. Pre-Sensor Distortion Models

The first component of the camera sensor is the lens system, which focuses light on the imaging plane. One or multiple lenses redirect light onto the sensor array such that the light is both intense enough to be measurable, and focused enough to provide a meaningful representation of the scene. In doing so, the resulting light can significantly differ from the ground truth. Artifacts include radial and tangential distortions, chromatic aberration, vignetting, lens flare, and depth-of-field blur. Since automotive and robotic camera are often focused at long distances with the object rarely appearing close to the camera, depth-of-field is rarely a source of interest in these application.

Radial and tangential distortions occur when the lens refracts the incoming light differently across the sensor array. This is most obvious in wide-angle cameras, with an example from [37] shown in Fig. 5. Although many perception systems first calibrate the image to remove the distortion [37], this artifact fundamentally changes the extent of the scene that is visible to the sensor and therefore AA. Exclusion of this distortion in simulation could lead to artificial results since the edges of the image may not faithfully represent the limits of visibility.

The research in this field has focused on both accurately recreating/understanding distortions as well as removing them from existing images (as a post-processing step). A summary of classical models is given in [38], which discusses approaches such as pinhole, fish-eye, and polynomial models

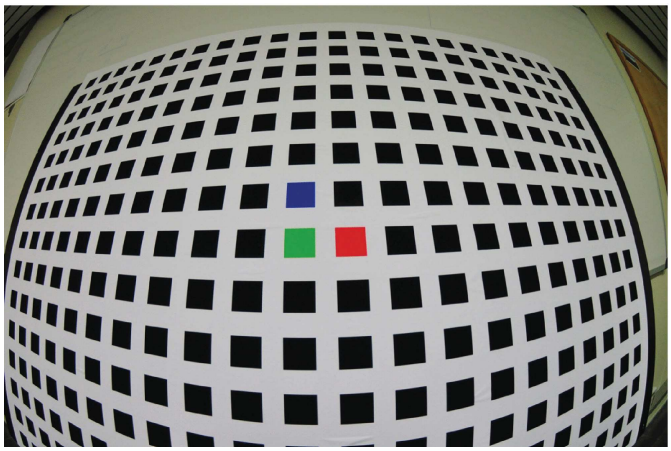


Fig. 5. Example lens distortion [37] ©2017 IEEE.

for the lens. More recent approaches for improving distortion modeling are presented in [39]. The radial, division, polynomial, and rational models discussed therein rely on regression techniques; the field of view (FOV) model characterizes the same distortion using physical parameters, i.e. the camera FOV. Being physics-based, the FOV model potentially removes the need for calibration, but is more constraining, limiting the lens systems that can be faithfully modeled. The FOV model is given by

$$r_2 = \frac{\tan(r_1 \tan(\omega))}{\tan \omega}, \tag{1}$$

where ω is the field of view, r_1 is the radius from the optical axis of the undistorted pixel location in the image, and r_2 is the radius of the distorted pixel. The radius from the optical axis can be defined as $r_i = \sqrt{x_i^2 + y_i^2}$ for $i = 1, 2$ where x_i and y_i are the horizontal and vertical pixel distances from the image center, taking care to adjust for the true angular distances these pixels represent in the optical system. This model is based exclusively on geometric considerations made in conjunction with a single spherical lens. It captures the mapping shown in Fig. 6, where the dashed lines represent the undistorted image extent and the solid lines represent the extent of the image captured by the camera (distorted image).

In robotic applications, a physics-based model can be helpful as multiple camera setups can be tried to identify a suitable FOV for the application without ever having access to physical hardware. However, for complex lenses the FOV model may not hold. In this case, regression approaches are common. In the simplest regression approach, the radial model gives the distorted radius of the pixel location, r_2, as a polynomial function of the undistorted radius, r_1:

$$r_2 = r_1(k_0 + k_1 r_1 + k_2 r_1^2 + k_3 r_1^3 + \ldots + k_n r_1^n), \tag{2}$$

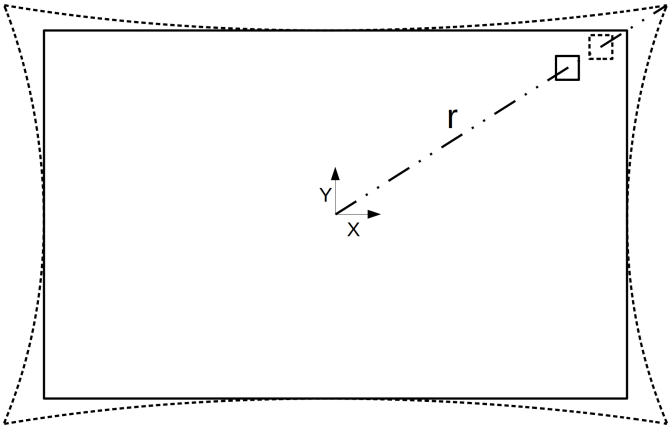


Fig. 6. Mapping of lens distortion, showing the original image extent and pixel location (dashed lines), and the distorted image extent and pixel location (solid lines). The distorted image is what is captured by the camera.

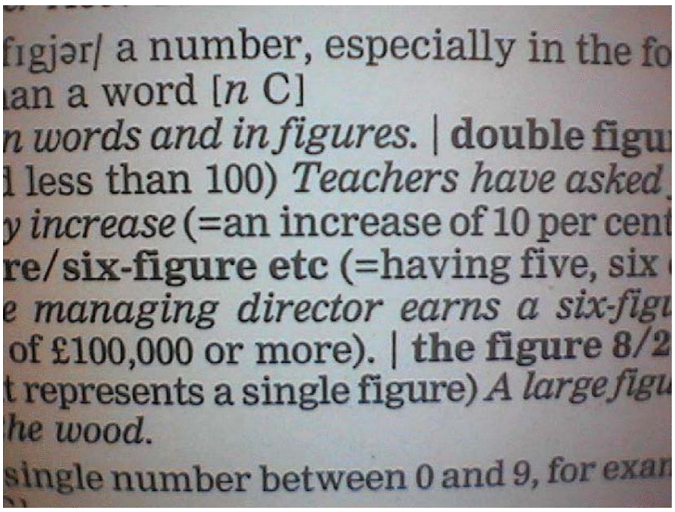


Fig. 7. Example vignetting [43] ©2004 IEEE.

where $k_0, \ldots, k_n$ are fitting parameters. The division model is the reciprocal of the radial model and uses similar tuning parameters. Lastly, the polynomial and rational models are analogous to the radial and division model respectively, with the caveat that the horizontal and vertical distortions are represented separately. While this makes for a more versatile model, it increases the burden on fitting the model to data, which is nontrivial. To calibrate these lens models, a regression method for estimating model parameters for both the radial and division models is proposed in [37]. Model parameters are obtained as the solution of an optimization problem that minimizes the error between projected lines and detected lines from a repeated square calibration pattern. Depending on the distortion model, calibration parameters will include the distortion fitting parameters (FOV model requires ω, radial model requires $k_0, \ldots, k_n$). Additionally, parameters that define the sensor's field of view, or depth-of-field characteristics (focal length, pixel size, axis skew, and focal point) can also be included in the camera calibration. A ray-tracing approach to distortion is outlined in [40], where the entire lens system is replaced with a mapping from input ray direction to output ray direction. This is easily implemented in a simulation framework within a ray tracing operation, but becomes more complex for raster graphics. In [41], the authors discuss a lower computational cost approach that modifies a scene in a pre- or post-process step to account for radial distortion when leveraging raster graphics. For closed-loop AA simulation, this computational burden may be too great to warrant its use. A data-driven method can be used to emulate lens distortion. In [42], the polynomial models are replaced with a neural network (NN). The advantage of this comes from a highly flexible NN that can represent complex distortions. While an NN could be used to represent the entire lens system, in [42] the authors train it to be a surrogate model for a single lens based on ray-traced scenes.

Another artifact that is considered in camera modeling is vignetting, a subtle distortion manifested as a darkening of the image around its outer edges. An example is shown in Fig. 7 [43]. The darkening is a consequence of the aperture

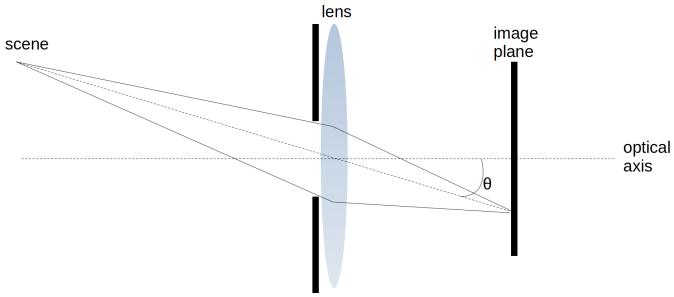


Fig. 8. Illustration of the optical axis and the off-axis angle, a quantity used for vignetting and lens distortion.

limiting the field of view of the scene for pixels near the extent of the sensor array [44]. Similarly to radial distortion, vignetting can be corrected in the final image. However, since this must be done digitally, and because vignetting decreases the number of collected photons, the noise in these regions may be increased when gain is applied.

In [44], vignetting is shown to have an effect on downstream processes related to perception, pointing to the importance of its modeling. Therein, vignetting in synthetic images is produced via a lookup table generated using calibration data collected from the sensor of interest. If focal length and image sensor diameter of the camera are known, a model described in [36] with derivation shown in [45] defines the relative fall off of illumination as $R = \cos^4\theta$, where R is the relative illumination and θ is the angle from the optical axis shown in Fig. 8. Subsequently, θ can be calculated using the pixel coordinates, the sensor size, and the distance from the lens to the image sensor.

Highly dependent on the lens material, lens flare is an artifact that leads to ghost objects appearing in the image. Often, these are blurry, regular-shaped light patches caused by undesired multi-path reflections of intense light that can change what information is present in a swath of pixels. As this can produce light blobs not present in the scene, it can cause problems during AA object detection. An example of lens flare discussed in [46] is shown in Fig. 9.

Fig. 9. Example lens flare [46] ©2018 Springer.

(a) Noise with minimal ISP from SIDD [51]. (b) Noise after ISP from Cityscapes [52].

Fig. 10. 20×20 pixel patches of noise from different datasets, illustrating noise with and without ISP augmentation.

Lens flare has complex underlying physics that is difficult to characterize, and its elimination is actively pursued by camera manufacturers who develop proprietary, anti-reflective lens coating to mitigate flare. Flare modeling is of interest in the computer graphics community where reconstruction of lens flare artifacts is done purely for visual appeal, and physical plausibility takes back seat to simulation speed [47]. For the modeling of physically plausible lens flare, the solution discussed in [47] models the probable multi-path reflections through the lens system. While effective, this is computationally expensive for closed-loop simulation. Additionally, it requires specialized information about the lens system, which is often hard to determine and/or difficult to access, most often being proprietary in nature. In an effort to reduce the reliance on physical properties, a method proposed in [46] uses a data-driven approach for flare reconstruction. It first learns the propensity of lenses to flare, and then generates a basis of ghost artifacts which, through optimization, can be used to generate realistic lens flares. While the approach is capable of modeling and simulating flare in unknown lens systems, it requires a significant amount of setup and measurement control to perform the data collection required. Although there are few data-driven solutions reported in the literature, it is likely that these pragmatic, data-driven approaches will make an impact in AA perception applications.

Chromatic aberration is caused by wavelength-dependent refraction that creates a slight positional offset of colors in an image. Specifically, the divergence of colors when light is refracted within the lens system is tied to the inability of the lens to focus all components of the light's spectrum at the same location in space [48]. This artifact can be problematic if precise edge-detection is needed in AA perception. A physics-based approach to generating chromatic aberration requires the ray tracing of wavelength-dependent rays through the lens system and into the scene [49]. While useful for computational camera simulation, this approach is too compute intensive to be useful in AA simulation. A similar method is proposed in [42]. However, rather than requiring full knowledge of the lens system, image calibration data is used to generate a regression model of the lens, including wavelength-dependent optimization parameters to model chromatic aberration. An approach more common for real-time rendering found in the gaming community is to produce distortions and artifacts via heuristics that generate visually appealing results. This paradigm is adopted in [48], where separation of the color spectrum at the edge of objects is performed, neglecting the lens or physics-based models entirely. This is a tradeoff between fidelity and performance; no study has been carried out to date to indicate whether this tradeoff is acceptable or not in AA simulation.

V. Sensor Noise Models

Individual photoreceptors in the image sensor array measure the intensity of incoming light, and output a voltage proportional to its intensity. There are several sources of noise associated with this process: photon shot, fixed pattern, dark current, readout, and quantization noise [36]. Photon-shot noise is associated with the uncertainty in the number of photos incident on a single photoreceptor. The noise distribution of photon count on the receptor follows a Poisson distribution which, for high light intensity (large photon count), can also be approximated with a Gaussian distribution. Fixed-pattern noise is the variation in both the sensitivity and the voltage associated with zero intensity across the array of photoreceptors. This fixed-pattern noise is reproducible between image frames and can often be reduced. Dark-current noise is the result of heat or non-visible light causing phantom fluctuations in the pixel measurements. Finally, noise is introduced when signals are read from the sensor array and falls into two categories: reading of the signal (readout noise); and conversion to digital signal (quantization noise). The readout noise occurs in the analog amplification and reset phases. Since the quantization noise is associated with the quantization of the signal, this takes on a uniform distribution with variance equal to one twelfth of the quantization interval in the digital precision [50]. An example of the noise produced by the camera is shown in Fig. 10a.

The raw sensor-measurement noise can be characterized as a pixel dependent Gaussian distribution, see the European Machine Vision Association (EMVA) Standard [53]. Referring to Fig. 11, the noise can be expressed as

$$\begin{aligned} I_n(p) &= I(p) + \eta, \qquad \eta \sim \mathcal{N}(0, \sigma_p^2) \\ \sigma_p^2 &= K^2\sigma_d^2 + \sigma_q^2 + K(\mu_p - \mu_{p,dark}), \end{aligned} \tag{3}$$

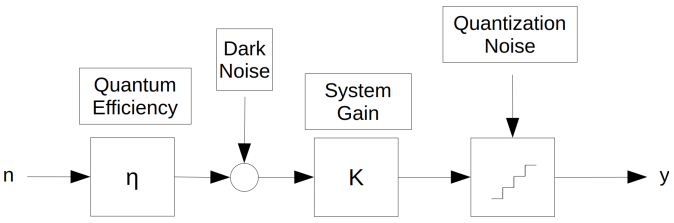


Fig. 11. Noise model based on EMVA standard.

where $I_n(p)$ is the intensity of the noisy pixel p, $I(p)$ is the intensity of the ground truth, η is the noise sampled from a normal distribution parameterized by σ_p. The latter is a function of the sensor gain K, the sensor readout noise σ_d, the quantization noise σ_q, and the dark noise and shot noise, which come from Poisson distributions parameterized by $\mu_{p,dark}$ and μ_p respectively.

The intensity-dependent, uncorrelated-noise model is commonly used to estimate noise levels or generate noisy synthetic data [54], [55]. This estimate is accurate for photoreceptor measurements or for evaluating new camera designs if image comparison is made in a raw sensed image [49]. However, only rarely are raw images used directly in a practical computer vision application. Indeed, computer vision applications mostly use modified RGB images that are based on the raw sensor readout. These RGB images are generated in the camera after processing and compressing/encoding the data. The pixel dependent model in Eq. (3) is not representative of noise after the on-chip processing of the ISP, which is discussed next.

VI. Post-Sensor Artifact Models

Several artifacts in RGB images are traced back to the ISP, which generates a more appealing image by adjusting for color, brightness, noise, and pixel errors. The ISP runs a collection of software algorithms that alter the image and include white balancing, demosaicing, auto exposure, auto contrast, color correction, deblurring, denoising, gamma correction, down sampling, and compression. The nature of the ISP transformations for a specific camera are typically unknown, proprietary, and complex, with only general descriptions publicly shared. Not providing solution details is the norm, since both the hardware and software at work in processing raw pixel data is what often provides a competitive advantage to a camera manufacturer.

The ISP alters both the color and noise in an image, with an example of the latter shown in Fig. 10b. Compared to the standard model for noise in the raw image, the noise in the RGB image tends to be longer-grain, and spatially, chromatically, and temporally correlated [57], [58]. Figure 12 elaborates on how a raw noisy image feeds into the ISP and how it is altered via a select set of image processing algorithms. The right half of Fig. 12 shows a sample set of algorithms proposed in [56], but it represents neither the full extent of image processing, nor the variation within specific algorithms, such as, for instance, white balance.

White balance processing seeks to mitigate color biases in the image [59] and can additionally adjust the brightness levels by applying gain to specific channels. A recent research direction has embraced this concept to balance or apply auto contrast to specific regions of the image to make darker regions more perceivable as well as to prevent saturation in bright regions of the image. Algorithms to produce context-aware automatic white balance have been proposed in [60].

Another process present in any raw-to-RGB conversion is demosaicing. This is the process by which the raw image from the sensor array is interpolated to generate an image of the same size with RGB values present at every pixel. This occurs at the resolution of the image array and, due to the nature of spatial interpolation, introduces spatial correlation of pixel noise. The pattern used in the raw image array is hardware-dependent, but the most common is the Bayer pattern. The algorithms for demosaicing can vary from bilinear interpolation to more complex edge-aware methods such as adaptive homogeneity directed (AHD) demosaicing [61], [62]. While the precise algorithm may not be available for a given camera, a method for estimating the algorithm from calibration data is described in [62]. The two primary effects of demosaicing are spatial correlation of noise, and the blurring or generation of artifacts near color boundaries.

In addition to white balancing and demosaicing, the ISP often includes denoising and deblurring [63]. These procedures are not perfect and may additionally correlate noise and alter color boundaries, either blurring or artificially sharpening regions. Auto exposure modeling is also important, since adjusting exposure plays a significant role in AA applications as the environmental lighting conditions can change rapidly (e.g. under a bridge or in a tunnel). The effects of auto exposure, which adjusts the exposure time, is delayed by a few frames since the ISP employs pipelining in order to achieve high frame throughput [64]. The challenge is in modeling this lag, along with the image artifacts that crop up during exposure adjustment. For instance, it is highly desirable to faithfully generate the image stream as a simulated AA moves, for instance, from intense sunlight to shade. Indeed, these transitions, through the artifacts that crop up in the sensing process, challenge the image segmentation and object recognition tasks that anchor many AA control policies. Finally, further processing such as compression, gamma correction, and color correction, alter the chromatic, spatial, or temporal nature of the images [65], [66].

If the ISP algorithms were known, they could be implemented directly in the simulation, removing the need for modeling. In reality, many algorithms are proprietary; even when publicly available, the parameters controlling them are not shared by the camera manufacturer. Consequently, by and large, the raw information processing carried out by the ISP is viewed as a black box.

In the image processing field of machine-learned denoising, it has been found that accurately modeling the noise and ISP as a single module can vastly improve training when using synthetic images. This is also relevant to the perception field as correlation between pixels can pose a challenge when attempting to precisely segment, pixel-wise, the edges of pedestrians or vehicles. To assist with training denoising algorithms, [57], [65]–[67] have each proposed and demonstrated lumped noise models which factor in the sensor and ISP together,

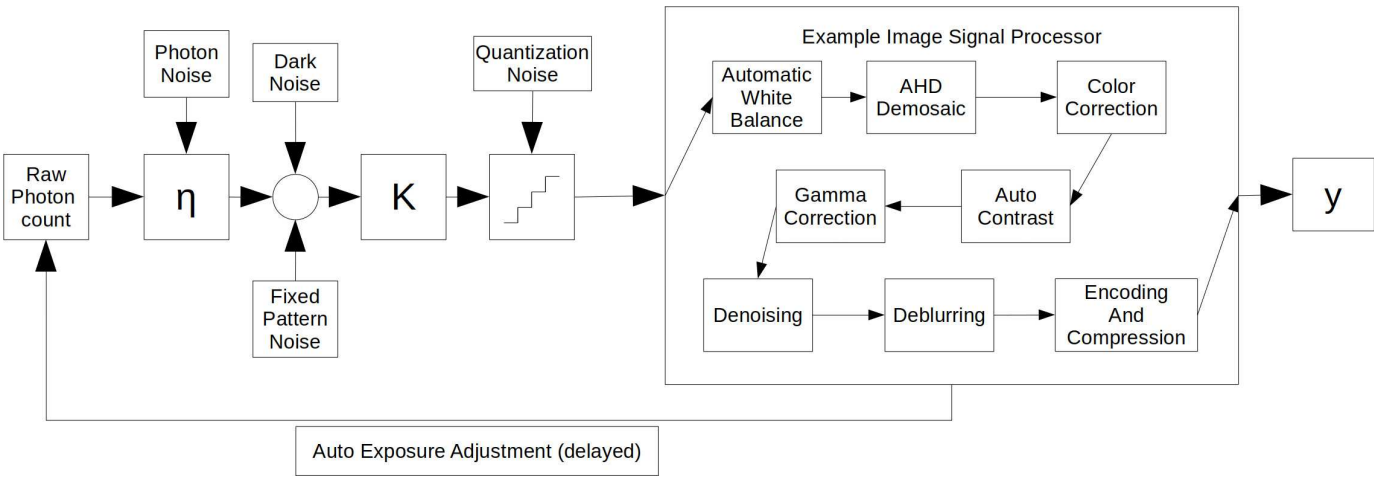


Fig. 12. Full noise factors including ISP which introduces significant noise correlation. ISP example modified from [56].

such that they incorporate spatial, chromatic, and temporal correlations.

In an effort to produce synthetic, noisy images for the purpose of training denoising algorithms, raw, pixel-level input was modified in [67] to account for plausible local spatial and chromatic correlations by approximating the effect of ISP demosaicing. The noise model is a modified version of the standard following

$$\begin{aligned} I &= f(L + \eta_s + \eta_c) + \eta_q \\ \eta_s &\sim \mathcal{N}(0, L\sigma_s^2) \\ \eta_c &\sim \mathcal{N}(0, \sigma_c^2) \\ \eta_q &\sim \mathcal{U}\Big(-\frac{q_0}{2}, \frac{q_0}{2}\Big), \end{aligned} \tag{4}$$

where I is the noisy image, f is the camera response function (CRF), L is the photon intensity, q_0 is the quantization, and σ_c, σ_s, η_s, and η_c are tunable parameters, of which the latter two are used to represent the intensity dependent and intensity independent noise distributions, respectively. The CRF, and conversely the inverse camera response function (ICRF), is a camera-dependent function that maps the photon-intensity space into the image color space. A CRF represents the pixel sensitivity, gain, and gamma correction of a specific camera. In order to convert the color image back to photon intensity, the ICRF is applied. In the work described in [67], the ICRF is important as it brings the image into a space where the modified standard model is appropriate.

This intensity-dependent and demosaicing method was augmented in [65] to include gamma correction and compression. The resulting model, in combination with domain randomization techniques, was shown to be more effective than previous approaches in training a convolutional denoising algorithm [65]. In [66], a similar technique implemented a pipeline for generating synthetic images by augmenting synthetic data with realistic noise in order to account for artifacts produced by the demosaicing, compression, and denoising algorithms typically present in cell phone cameras. The results shown in [66] demonstrate that the method advanced therein allowed for the effective training and transfer to reality of a denoising algorithm. A model for cross-channel dependence of noise was proposed in [57] to increase the fidelity of color-specific relationships. Although these implementations are specific to training denoising networks, they provide insights for synthetically adding noise to images to increase their veracity when used in AA simulation.

In [68], the entire ISP was approximated such that images taken with one camera could be mapped back to raw inputs, and then mapped forward as if they had been taken with a different camera. This solution relied on prohibitively large amounts of real and calibrated data. Similar work was introduced in [69] for the simulation and estimation of proposed ISP algorithms to evaluate their effectiveness. The approach combined ML and simulation of the entire imaging system to estimate the final image that would be formed with a proposed camera. However, the reliance on full camera simulation, including modeling all lenses and pixel sensitivities, would be computationally expensive for AA applications, preventing these methods' direct transfer to AA simulation.

It should be noted that there is an entire computational camera simulation community keen on predicting sensor capabilities for new designs; i.e., characterizing sensor performance before building it. This requires accurately predicting the incoming light, precisely estimating the photon intensity levels at the sensor, and then, through accurate models of the image sensor, predicting the distortion, noise levels, and abilities of the proposed design [70]. While this process is highly precise, it poses extreme computational challenges that make it intractable for AA simulation [71].

VII. Virtual Environment Considerations

In this section, the focus shifts from camera modeling to camera simulation, the latter being highly influenced by the virtual world in which the camera is immersed. Note that the virtual world's influence extends beyond its interplay with the camera model, as it shapes, downstream, the task of *perception*. As far as camera simulation is concerned, the virtual world assets and attributes of interest include meshes, material properties, textures, lighting, sprites, decks,

etc., which are concepts that originated in the computer graphics community. A discussion of physically-based rendering and photorealism is provided in [72]. Against this backdrop, a subpar representation of the virtual world will adversely impact perception, regardless of how faithfully the camera model can replicate lens artifacts, sensor noise, ISP distortion, etc. For instance, the effective modeling of light in the virtual world directly impacts the post-sensor effects discussed in Section § VI, as the lighting conditions and contrast influence the autoexposure, which controls the gain, and therefore noise levels or saturation. Hence, without proper modeling of the scene, a highly accurate autoexposure model would still be unable to produce realistic results.

Beyond lighting, several other virtual world effects play an important role in perception [73]. It has been noted repeatedly that perception algorithms, e.g. [30], [31], [34], [35], [74], are highly sensitive to weather-based artifacts associated with cameras [75]. In comparison to lighting, the task of accommodating weather effects in virtual worlds is less established. Much of the work to render environment conditions such as fog, rain, snow, fire, dust, smoke, etc., has been driven by the gaming community [76], [77] in the pursuit of achieving human-judged realism and meeting performance requirements. Simulating these weather-induced effects at a level that is AA perception satisfactory, poses stiff challenges. Alas, these are precisely the conditions encumbering many safety critical and hazardous scenarios of interest, for which simulation may be the only feasible method for probing AA behavior without placing people or property in harms way.

An example of a gaming community-provided solution for rain distortion of images used in AA perception is presented in [78]. A rain-related discussion in [79] touches on a method to reconstruct a scene from an image, predict the density of rain on a per-pixel basis, and augment the scene with rain to improve the robustness of machine learned algorithms. Research and techniques used in rain removal [80], if played in reverse, can provide insights into how to augment camera sensor data. Rain based on computer graphics research for video gaming is supported in various simulation environments [10], [81] with further research from the computer graphics community detailed in [82]–[84]. It should be emphasized that the challenge is not to *inject* rain in a scene but rather to do so and have the camera model pick it up and produce synthetic images of a quality suitable for use downstream, in AA perception.

To accommodate snow in virtual worlds, falling snow has been discussed in [85], where the precipitation is treated much like rain, inducing streaks and artifacts in the image. However, snow poses additional challenges to image plausibility due to the complex movement of accumulated snow. This is the subject of research aimed at capturing the realism of snow accumulation and movement [86], [87]. From a vehicle's perspective, this movement may be important as snowy roads with traffic significantly change in appearance over time. For other aspects of snow in a scene, it may be important to model how the accumulated snow impacts the appearance of the background scene [88].

For fog representation, the graphics community has developed expeditious models that approximate the light scattering based on fog density and color, by using attenuation laws [89], [90]. However, little work has been conducted to quantify how well the approaches work in perception: the synthetic images look good to the human eye, but might be of little use when used for perception. What has been shown though, is that a large change in perception ability occurs in images with fog [85], [89].

Effects such as dust and smoke are known to impede the perception in real-world applications. Similar to snow, this may take the form of airborne particulates (e.g. smoke), or may present as an accumulated layer on camera lenses or the ground. Various techniques are used in graphics and the gaming community and some attempts have been made to adapt them for AA simulation purposes. Smoke simulation work, reported in [91]–[93], looks at the mechanics of movement. Depending on nature of the AA simulation, a sophisticated model may not be necessary, and as such a cruder model may suffice. While smoke represents an extreme edge case for AAs, dust, even in typical use, can accumulate and begin to affect normal operation. Layered dust has been studied, as shown in [94], [95] in order to realistically model how it accumulates and affects the visual characteristics of surfaces.

While various environment conditions can be accounted for in several simulation platforms [10], [96]–[98], very little research has been done into gauging, for instance, the realism of simulated rain, in order to understand the extent to which data produced is realistic insofar as AA perception is concerned. Moreover, there are many nuances for the problem at hand. For instance, how snow collects on a camera and possibly occludes vision is highly dependent on the camera, lens, shielding, and vehicle. Ultimately, the models and methods used in gaming and computer graphics can provide a solid foundation, yet are not sufficient to be immediately and effectively used in AA simulation.

VIII. Data Driven Camera Simulation

Hitherto, the discussion concentrated on capturing processes that take place in the individual stages of the camera sensing pipeline, i.e., the optical system, image sensor, or image signal processing. When combined, submodels of these components yield camera models that are robust and expeditious. A limitation of this approach is tied to the lack of meaningful parameters associated with the submodels. Producing these parameters is nontrivial. For instance, for the ISP component model, it is exceedingly difficult to identify model parameters for the black box algorithms that come into play. As an alternative, or potentially an augmentation to the traditional approach, machine learning (ML) has shown promise in generating visually plausible camera sensor data [99]. ML can be used to replace submodels (briefly touched upon in Sections § IV and VI), the entire sensing pipeline, or augment the virtual environment.

Recent ML research has led to significant progress in the so-called image-to-image translation, e.g., producing a map from an aerial image; producing a picture of a landscape in winter from an image in summer time; translating a black-and-white image into a color image, etc. Seminal work in this field centers on generative adversarial networks (GANs) [100].

TABLE I

AV SIMULATION PLATFORMS SUPPORTING MODELS OF RGB MONO-CAMERAS. THESE ARE SPLIT INTO TWO CATEGORIES BASED ON THE AMOUNT OF PUBLISHED INFORMATION AVAILABLE TO REFLECT WHICH PLATFORMS ARE JUDGED BASED ON LIMITED DOCUMENTATION ONLY. PLATFORMS ARE SORTED WITHIN THE TWO GROUPS BY ALPHABETICAL ORDER. KEY: AWGN: ADDITIVE WHITE GAUSSIAN NOISE; IDGN: INTENSITY-DEPENDENT GAUSSIAN NOISE; FOV: FIELD-OF-VIEW MODEL; DASH(−): NOT REPORTED OR NOT AVAILABLE; PLUS (+): CAPABILITY REPORTED WITHOUT SPECIFIC MODEL. NOTE: PLATFORMS MAY SUPPORT MODELS AND CAPABILITIES BEYOND THE SCOPE OF THIS REVIEW INCLUDING OTHER SENSORS, OTHER WEATHER OR ARTIFACTS (E.G. DUST), VEHICLE DYNAMICS, PEDESTRIANS, ETC. FURTHERMORE, SPECIFIC MODELS MAY BE SUPPORTED YET INFORMATION WAS UNATTAINABLE BY THE AUTHORS. THE INFORMATION BELOW IS NOT INTENDED AS AN ENDORSING OF A PRODUCT OR A RANKING OR JUDGMENT OF PLATFORMS, BUT RATHER AS A SUMMARY OF MORE COMMON PLATFORMS CURRENTLY USED IN THIS FIELD

	Simulation Software	Lens Distortions	Noise	ISP Artifacts	Weather
Published Info Available	AirSim	From Unreal	From Unreal	From Unreal	From Unreal
	AutonoVi	fish-eye	+	-	Fog
	Carla	Fish-eye, lens flare	+	auto-exposure, gamma	Rain
	Chrono	FOV	IDGN, AWGN	Gamma	None
	CoppelliaSim (Formerly V-Rep)	-	-	-	-
	Gazebo	Radial	AWGN	-	-
	LG SVL	Fish-eye	-	JPEG compression	Rain
	MAVS	+	+	Gamma	Rain, Snow, Fog
	USARSim	-	-	-	-
	VANE/ANVEL	+	-	Compression, Gamma	Rain
	Webots	-	-	-	-
Limited Info Available	4D Virtualiz	Fish-eye, omnidirectional	-	-	+
	Ansys SPEOS and OPTIS	+	+	+	-
	ASM Traffic (dspace)	+	+	+	-
	CarMaker (IPG Automotive)	+	+	+	-
	DYNA4 (Tesis)	+	+	+	-
	ESI Pro-SiVIC	Fish-eye, omnidirectional,+	+	+	Rain, Fog, Snow
	Mathworks Simulation 3D Camera	radial and tangential	-	-	-
	NVIDIA ISAAC Sim	+	+	+	-
	NVIDIA DRIVE Sim	+	+	+	-
	OpenDaVINCI and OpenDLV	+	+	+	-
	rfPro	+	+	+	Rain, Snow, Fog
	SCANeR from AV Simulation	+	+	+	-
	Siemens (TASS) PreScan	+	+	+	-
	SynCity	+	Data driven	+	-
	Vigrade	+	+	+	Rain
	VTD - Vires	+	+	+	Snow

These ML-based models are obtained by learning to map input to output through optimization. An approach proposed in [101] seeks to learn a model for image noise and then engage in synthetic generation of noisy images. The work is analogous to transferring noise from one image to another via a GAN [100], [102] which can introduce abstract image augmentations. By using machine learning, and specifically convolutional neural networks (CNN), high-dimensional features can be learned such as edges and contextual relationships between pixel patches.

Examples of GAN-based image-to-image translation can be found in [103]–[106]. In [105], it is demonstrated that on-road images can be converted between various weather and lighting conditions. GAN-based weather-induced augmentations were employed in [107] to evaluate the ability of autonomous vehicle perception networks to remain robust in decision making between clear, foggy, and rainy versions of operating scenarios (or images thereof). Another method for converting between weather domains such as sunny, rainy, and foggy, was proposed in [108], where the authors solved the image-to-image translation problem for weather with remarkable accuracy. Concretely, synthetic images were rendered using traditional graphics techniques and then altered to introduce complex weather effects. In [109], a GAN was successfully trained to remove rain from images. If the training data is reversed, this would result in an approach to generate synthetic rain for rendered images. Similar work is reported in [110], but rather than using a GAN, a NN is trained directly using supervised learning to remove rain droplets from images obtained while driving. While supervised learning allows for more direct training approaches, this was only made possible by collecting paired samples with a stereo camera setup with one of the cameras obscured by droplets in front of the lens. The image-to-image translation can be applied to semantic maps directly, as shown in [111]. The approach accomplishes two things: it encodes all weather translations into a single network; and it removes the need to generate an initial image altogether. While this allows a fully data-driven and general approach to a complex project, it requires the neural network to generate fine-grain detail, burdening the network's capacity to encode more weather-detailed information.

Note that handling weather-related artifacts does not represent the only use of ML in synthetic perception. GANs have also been used in driving augmentations in [112], where the virtual world was altered by injecting synthetic pedestrians into the scene. This method was applied to specific regions of the image, with background-aware training, in an attempt to prevent unrealistic position and size of the augmentation. In the future, synthetic pedestrians could become participants in closed-loop simulation. However, in line with most of

the ML-based research, the solution in [112] would require additional temporal constraints to increase realism.

IX. Current State of Camera Simulation for Autonomous Agents

A. Camera Model Implementations

For application in robotics and autonomous vehicles, camera models have been implemented in several simulation frameworks. These models tend to be expeditious in nature, which often comes at the expense of fidelity. This stems from a broad reliance on game engines for rendering. To understand the current state of camera simulation, Tab. I lists known simulation packages with significant camera support for AAs. Of particular importance are the simulators that have support for lens distortion, noise, and ISP artifacts, as these represent higher-fidelity modeling support. With the topic of AA camera simulation being nascent, very much in flux, and highly competitive, it comes as no surprise that many simulation platforms do not have published details on the implemented models and algorithms. As such, the strengths and limitations of these camera models have to be surmised from sparse information available on the developers' websites and from marketing information.

Of particular note are the capabilities of Carla [10] and AirSim [11] due to their extensive use in the research community. These two platforms are built on gaming engines (Unreal and Unity), and include camera and weather models developed for gaming applications. Another important software platform in the robotics community is Gazebo, which is well known for its in-depth support for ROS. Gazebo supports a broad range of sensors, but only includes a simple lens distortion model and an oversimplified AWGN model for noise, which may constrain the fidelity of camera-reliant Gazebo simulations.

In the commercial sector, software like SPEOS and OPTIS from Ansys, Pro-SiVIC from ESI group, VTD from MSC Software, DRIVE Sim from NVIDIA, and SynCity from CVEDIA exemplify a prolific field of highly developed software, many with validated solutions. Unfortunately, little information is published about their specific camera models and implementations.

B. Validation and Sensitivity Studies

Current efforts to validate camera models center around traditional methods of data validation which seek to control an environment and compare the simulated data to the real-world data. These are usually performed in a tightly controlled setting, with specific texture patterns and lighting. An example of such validation is shown by the IFSTTAR team, [113], [114], where Pro-SiVIC software from ESI group is demonstrated to produce very similar results for lens distortion and camera response. While this validation in controlled environments is highly important, it is not necessarily indicative of performance in on-road scenarios, which pose unique challenges insofar as the downstream perception is concerned. Specifically, even when a simulation platform and underlying camera model are validated in controlled tests, they do not typically generate synthetic data with sufficient realism to replace real images when training machine learning algorithms for perception [20], [115]. Furthermore, research has suggested for improved downstream perception, matching the proper distortion levels, blur, and noise play an important role in the simulator [116], [117].

X. Conclusion

This overview of the camera modeling and simulation topic touched on two classes of approaches. The class of physics-based methods attempt to mimic the processes that take place in the camera sensing pipeline, which has three stages: optical system, sensor array, and ISP. Each stage has associated with it distortion, noise, and artifacts that affect the end product, in this case a synthetic image that is a replica of the virtual world in which the camera is immersed. A second class of methods relies on data-driven approaches that seek to augment the virtual world, to replace submodels of various camera stages, or even eliminate altogether the traditional, physics-based camera models. The contribution highlights two important aspects. First, camera modeling and simulation are closely tied to the virtual world the camera is immersed in and expected to generate images of. This camera-virtual world interplay permeates the design of the camera, its use in simulation, and the way in which the virtual world needs to be set up. Second, there are few methods in place to measure the quality of the synthetic images generated by a camera sensor. The fact that one finds the image visually plausible, or "realistic," may not reflect how successful the downstream use of these synthetic images will be in perception for AA simulation. Looking ahead, reducing the simulation-to-reality gap will require action in multiple directions, see, for instance [4]. However, it is our believe that prominent among these directions should be camera and virtual world modeling. There are numerous outstanding questions about the fidelity required in this nascent simulation domain, as well as about the design of quantitative metrics that gauge the level of realism associated with camera simulation insofar as the task of perception is concerned. Having quantitative metrics to judge the realism in camera simulation will help the community understand the level of fidelity required in camera modeling and virtual world definition, which bodes well for the problem at hand as well as the larger AA simulation aim.

Acknowledgment

The U.S. Government assumes no liability for the ideas expressed in this document or the use thereof.

References

[1] The New York Times. *Self-Driving Uber Car Kills Pedestrian in Arizona, Where Robots Roam*. Accessed: Apr. 7, 2018. [Online]. Available: https://www.nytimes.com/2018/03/19/technology/uber-driverless-fatality.html

[2] Digital Trends. *6 Self-Driving Car Crashes That Tapped the Brakes on the Autonomous Revolution*. Accessed: Sep. 29, 2018. [Online]. Available: https://www.digitaltrends.com/cool-tech/most-significant-self-driving-car-crashes/

[3] Wired. *Waymo's Self-Driving Car Crash in Arizona Revives Tough Questions*. Accessed: Sep. 29, 2018. [Online]. Available: https://www.wired.com/story/waymo-crash-self-driving-google-arizona/

[4] H. Choi *et al.*, "On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward," *Proc. Nat. Acad. Sci. USA*, vol. 118, no. 1, Jan. 2021, Art. no. e1907856118. [Online]. Available: https://www.pnas.org/content/118/1/e1907856118

[5] J. Collins, D. Howard, and J. Leitner, "Quantifying the reality gap in robotic manipulation tasks," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 6706–6712.

[6] Y. Kang, H. Yin, and C. Berger, "Test your self-driving algorithm: An overview of publicly available driving datasets and virtual testing environments," *IEEE Trans. Intell. Vehicles*, vol. 4, no. 2, pp. 171–185, Jun. 2019.

[7] Lindsay Brooke. *SAE Kicks off Process to Establish Performance Standards for Autonomous Vehicle Testing*. Accessed: Sep. 29, 2018. [Online]. Available: https://www.sae.org/news/2018/08/sae-kicks-off-av-testing-performance-standards-process

[8] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," *Sensors*, vol. 19, no. 3, p. 648, 2019.

[9] N. Sünderhauf *et al.*, "The limits and potentials of deep learning for robotics," *Int. J. Robot. Res.*, vol. 37, nos. 4–5, pp. 405–420, 2018.

[10] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 1–16.

[11] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*. Cham, Switzerland: Springer, 2018, pp. 621–635.

[12] NVIDIA. (2018). *NVIDIA DRIVE Constellation*. Accessed: Feb. 6, 2018. [Online]. Available: https://www.nvidia.com/en-us/self-driving-cars/drive-constellation/

[13] D. W. Carruth, "Simulation for training and testing intelligent systems," in *Proc. World Symp. Digit. Intell. Syst. Mach. (DISA)*, Aug. 2018, pp. 101–106.

[14] C. Goodin, R. Kala, A. Carrrillo, and L. Y. Liu, "Sensor modeling for the virtual autonomous navigation environment," in *Proc. IEEE Sensors*, Oct. 2009, pp. 1588–1592.

[15] A. Elmquist, R. Serban, and D. Negrut, "A sensor simulation framework for training and testing robots and autonomous vehicles," *J. Auto. Vehicles Syst.*, vol. 1, no. 2, Apr. 2021, Art. no. 021001.

[16] C. Bowles *et al.*, "GAN augmentation: Augmenting training data using generative adversarial networks," 2018, *arXiv:1810.10863*. [Online]. Available: http://arxiv.org/abs/1810.10863

[17] P. Durst, "Using physics-based M&S for training and testing machine learning algorithms," in *Proc. 5th Int. Conf. Modeling Simulation Auto. Syst. (MESAS)*, vol. 11472, Prague, Czech Republic: Springer, Oct. 2018, p. 445.

[18] Z. Liu *et al.*, "A system for generating complex physically accurate sensor images for automotive applications," *Electron. Imag.*, vol. 2019, no. 15, pp. 1–53, 2019.

[19] M. Wrenninge and J. Unger, "Synscapes: A photorealistic synthetic dataset for street scene parsing," 2018, *arXiv:1810.08705*. [Online]. Available: http://arxiv.org/abs/1810.08705

[20] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3234–3243.

[21] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 102–118.

[22] A. Dundar, M.-Y. Liu, T.-C. Wang, J. Zedlewski, and J. Kautz, "Domain stylization: A strong, simple baseline for synthetic to real image domain adaptation," 2018, *arXiv:1807.09384*. [Online]. Available: http://arxiv.org/abs/1807.09384

[23] N. Soufi and M. Valdenegro-Toro, "Data augmentation with symbolic-to-real image translation GANs for traffic sign recognition," 2019, *arXiv:1907.12902*. [Online]. Available: http://arxiv.org/abs/1907.12902

[24] A. Lukashou, "Improving benchmarks for autonomous vehicles testing using synthetically generated images," 2019, *arXiv:1904.10261*. [Online]. Available: http://arxiv.org/abs/1904.10261

[25] A. Rozantsev, V. Lepetit, and P. Fua, "On rendering synthetic images for training an object detector," *Comput. Vis. Image Understand.*, vol. 137, pp. 24–37, Aug. 2015.

[26] G. Girardin, "Road to robots. Sensors and computing for autonomous vehicle," in *Proc. Auto. Vehicle Sensors Conf.*, 2018, pp. 1–38. [Online]. Available: https://www.autosensorsconf.com/sites/autosensorsconf/files/assets/1

[27] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., (CVPR)*, vol. 1, Jun. 2004, p. 1.

[28] F. Yu *et al.*, "BDD100K: A diverse driving dataset for heterogeneous multitask learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2636–2645.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[30] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*. [Online]. Available: http://arxiv.org/abs/1804.02767

[31] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[32] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 21–37.

[33] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proc. ICCV*, Oct. 2017, pp. 2961–2969.

[34] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[35] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.

[36] J. E. Farrell and B. A. Wandell, "Image systems simulation," in *Handbook of Digital Imaging*. Hoboken, NJ, USA: Wiley, 2015, pp. 1–28.

[37] D. Santana-Cedrés, L. Gomez, M. Alemán Flores, A. Salgado, L. Mazorra, and L. Alvarez, "Estimation of the lens distortion model by minimizing a line reprojection error," *IEEE Sensors J.*, vol. 17, no. 9, pp. 2848–2855, May 2017.

[38] P. Sturm, S. Ramalingam, J.-P. Tardif, S. Gasparini, and J. Barreto, "Camera models and fundamental concepts used in geometric computer vision," *Found. Trends Comput. Graph. Vis.*, vol. 6, pp. 1–183, Jan. 2011.

[39] Z. Tang, R. G. von Gioi, P. Monasse, and J.-M. Morel, "A precision analysis of camera distortion models," *IEEE Trans. Image Process.*, vol. 26, no. 6, pp. 2694–2704, Jun. 2017.

[40] Q. Zheng and C. Zheng, "Adaptive sparse polynomial regression for camera lens simulation," *Vis. Comput.*, vol. 33, nos. 6–8, pp. 715–724, Jun. 2017.

[41] M. Lambers, H. Sommerhoff, and A. Kolb, "Realistic lens distortion rendering," in *Proc. 26th Int. Conf. Central Eur. Comput. Graph., Vis. Comput. Vis.*, May 2018, pp. 27–32.

[42] Q. Zheng and C. Zheng, "NeuroLens: Data-driven camera lens simulation using neural networks," *Comput. Graph. Forum*, vol. 36, no. 8, pp. 390–401, Dec. 2017.

[43] W. Yu, Y. Chung, and J. Soh, "Vignetting distortion correction method for high quality digital imaging," in *Proc. 17th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2004, pp. 666–669.

[44] K. Saad and S.-A. Schneider, "Camera vignetting model and its effects on deep neural networks for object detection," in *Proc. IEEE Int. Conf. Connected Vehicles Expo (ICCVE)*, Nov. 2019, pp. 1–5.

[45] D. A. Kerr, "Derivation of the cosine fourth law for falloff of illuminance across a camera image," Tech. Rep. 4, 2007.

[46] A. Walch *et al.*, "Lens flare prediction based on measurements with real-time visualization," *Vis. Comput.*, vol. 34, no. 9, pp. 1155–1164, Sep. 2018.

[47] S. Lee and E. Eisemann, "Practical real-time lens-flare rendering," in *Computer Graphics Forum*, vol. 32. Hoboken, NJ, USA: Wiley, 2013, pp. 1–6.

[48] N. Bisagno and N. Conci, "Virtual camera modeling for multi-view simulation of surveillance scenes," in *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, Sep. 2018, pp. 2170–2174.

[49] T. Nürnberg, M. Schambach, D. Uhlig, M. Heizmann, and F. P. León, "A simulation framework for the design and evaluation of computational cameras," *Proc. SPIE*, vol. 11061, Jun. 2019, Art. no. 1106102.

[50] M. Granados, B. Ajdin, M. Wand, C. Theobalt, H.-P. Seidel, and H. P. A. Lensch, "Optimal HDR reconstruction with linear digital cameras," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 215–222.

[51] A. Abdelhamed, S. Lin, and M. S. Brown, "A high-quality denoising dataset for smartphone cameras," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1692–1700.

[52] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223.

[53] *Standard for Characterization of Image Sensors and Cameras*, EMVA Standard, European Machine Vision Association, Barcelona, Spain, 2010, vol. 3.

[54] J. Rossmann, N. Hempe, M. Emde, and T. Steil, "A real-time optical sensor simulation framework for development and testing of industrial and mobile robot applications," in *Proc. 7th German Conf. Robot. (ROBOTIK)*, 2012, pp. 1–6.

[55] S. W. Hasinoff, F. Durand, and W. T. Freeman, "Noise-optimal capture for high dynamic range photography," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 553–560.

[56] S.-H. Choi, J. Cho, Y.-M. Tai, and S.-W. Lee, "Implementation of an image signal processor for reconfigurable processors," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2014, pp. 141–142.

[57] S. Nam, Y. Hwang, Y. Matsushita, and S. J. Kim, "A holistic approach to cross-channel image noise modeling and its application to image denoising," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1683–1691.

[58] S. W. Hasinoff *et al.*, "Burst photography for high dynamic range and low-light imaging on mobile cameras," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–12, Nov. 2016.

[59] Z. Deng, A. Gijsenij, and J. Zhang, "Source camera identification using auto-white balance approximation," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 57–64.

[60] M. Afifi and M. S. Brown, "Deep white-balance editing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1397–1406.

[61] H. S. Malvar, L.-w. He, and R. Cutler, "High-quality linear interpolation for demosaicing of Bayer-patterned color images," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2004, p. 485.

[62] J. Takamatsu, Y. Matsushita, T. Ogasawara, and K. Ikeuchi, "Estimating demosaicing algorithms using image noise variance," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 279–286.

[63] A. Ignatov, L. Van Gool, and R. Timofte, "Replacing mobile camera ISP with a single deep learning model," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 536–537.

[64] Y. Su and C.-C. J. Kuo, "Fast and robust camera's auto exposure control using convex or concave model," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2015, pp. 13–14.

[65] S. Guo, Z. Yan, K. Zhang, W. Zuo, and L. Zhang, "Toward convolutional blind denoising of real photographs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1712–1722.

[66] R. Jaroensri, C. Biscarrat, M. Aittala, and F. Durand, "Generating training data for denoising real RGB images via camera pipeline simulation," 2019, *arXiv:1904.08825*. [Online]. Available: http://arxiv.org/abs/1904.08825

[67] C. Liu, R. Szeliski, S. B. Kang, C. L. Zitnick, and W. T. Freeman, "Automatic estimation and removal of noise from a single image," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 299–314, Feb. 2008.

[68] S. J. Kim, H. T. Lin, Z. Lu, S. Süsstrunk, S. Lin, and M. S. Brown, "A new in-camera imaging model for color computer vision and its application," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2289–2302, Dec. 2012.

[69] H. Jiang, Q. Tian, J. Farrell, and B. A. Wandell, "Learning the image processing pipeline," *IEEE Trans. Image Process.*, vol. 26, no. 10, pp. 5032–5042, Oct. 2017.

[70] M.-G. Retzlaff, J. Hanika, J. Beyerer, and C. Dachsbacher, "Physically based computer graphics for realistic image formation to simulate optical measurement systems," *J. Sensors Sensor Syst.*, vol. 6, no. 1, p. 171, 2017.

[71] H. Blasinski, J. Farrell, T. Lian, Z. Liu, and B. Wandell, "Optimizing image acquisition systems for autonomous driving," *Electron. Imag.*, vol. 2018, no. 5, pp. 1–161, 2018.

[72] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*. San Mateo, CA, USA: Morgan Kaufmann, 2016.

[73] R. Song, J. Wetherall, S. Maskell, and J. Ralph, "Weather effects on obstacle detection for autonomous car," in *Proc. 6th Int. Conf. Vehicle Technol. Intell. Transp. Syst.*, Jan. 2020, pp. 331–341.

[74] X. Liu, Z. Deng, and Y. Yang, "Recent progress in semantic image segmentation," *Artif. Intell. Rev.*, vol. 52, no. 2, pp. 1089–1106, Aug. 2019.

[75] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in *Proc. 40th Int. Conf. Softw. Eng.*, May 2018, pp. 303–314.

[76] Unity3D. (2016). *Main Website*. Accessed: Jun. 9, 2016. [Online]. Available: https://unity3d.com/

[77] Epic Games. (2020). *Unreal Engine*. [Online]. Available: https://www.unrealengine.com

[78] M. Tremblay, S. S. Halder, R. de Charette, and J.-F. Lalonde, "Rain rendering for evaluating and improving robustness to bad weather," *Int. J. Comput. Vis.*, vol. 129, no. 2, pp. 341–360, 2020.

[79] D. Hospach, S. Mueller, W. Rosenstiel, and O. Bringmann, "Simulation of falling rain for robustness testing of video-based surround sensing systems," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2016, pp. 233–236.

[80] X. Fu, J. Huang, X. Ding, Y. Liao, and J. Paisley, "Clearing the skies: A deep network architecture for single-image rain removal," *IEEE Trans. Image Process.*, vol. 26, no. 6, pp. 2944–2956, Jun. 2017.

[81] A. Best, S. Narang, L. Pasqualin, D. Barber, and D. Manocha, "AutonoVi-sim: Autonomous vehicle simulation platform with weather, sensing, and traffic control," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 1048–1056.

[82] P. Rousseau, V. Jolivet, and D. Ghazanfarpour, "Realistic real-time rain rendering," *Comput. Graph.*, vol. 30, no. 4, pp. 507–518, Aug. 2006.

[83] Y. Weber, V. Jolivet, G. Gilet, and D. Ghazanfarpour, "A multiscale model for rain rendering in real-time," *Comput. Graph.*, vol. 50, pp. 61–70, Aug. 2015.

[84] K. Garg and S. K. Nayar, "Vision and rain," *Int. J. Comput. Vis.*, vol. 75, no. 1, pp. 3–27, 2007.

[85] A. V. Bernuth, G. Volk, and O. Bringmann, "Simulating photo-realistic snow and fog on existing images for enhanced CNN training and evaluation," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 41–46.

[86] P. Goswami, C. Markowicz, and A. Hassan, "Real-time particle-based snow simulation on the GPU," in *Proc. EGPGV Eurograph.-Eur. Assoc. Comput. Graph.*, 2019, pp. 1–9.

[87] A. Wasmut, "Deformable snow rendering," Bachelor thesis, Univ. Koblenz-Landau, Landau, Germany, 2019.

[88] B. Neukom, S. M. Arisona, and S. Schubiger, "Real-time GIS-based snow cover approximation and rendering for large terrains," *Comput. Graph.*, vol. 71, pp. 14–22, Apr. 2018.

[89] K. Li, Y. Li, S. You, and N. Barnes, "Photo-realistic simulation of road scene for data-driven methods in bad weather," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 491–500.

[90] S. Kahraman and R. De Charette, "Influence of fog on computer vision algorithms," Inria, Paris, France, Tech. Rep., 2017.

[91] K. Zhou, Z. Ren, S. Lin, H. Bao, B. Guo, and H.-Y. Shum, "Real-time smoke rendering using compensated ray marching," in *Proc. ACM SIGGRAPH Papers (SIGGRAPH)*, 2008, pp. 1–12.

[92] R. Fedkiw, J. Stam, and H. W. Jensen, "Visual simulation of smoke," in *Proc. 28th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 2001, pp. 15–22.

[93] T. Brochu, T. Keeler, and R. Bridson, "Linear-time smoke animation with vortex sheet meshes," in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Animation*. Princeton, NJ, USA: Citeseer, 2012, pp. 87–95.

[94] S.-C. Hsu and T.-T. Wong, "Simulating dust accumulation," *IEEE Comput. Graph. Appl.*, vol. 15, no. 1, pp. 18–22, Jan. 1995.

[95] J. Guo and J.-G. Pan, "Real-time simulating and rendering of layered dust," *Vis. Comput.*, vol. 30, nos. 6–8, pp. 797–807, Jun. 2014.

[96] O. Michel, "Cyberbotics Ltd. Webots: Professional mobile robot simulation," *Int. J. Adv. Robotic Syst.*, vol. 1, no. 1, p. 5, Mar. 2004.

[97] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "USARSim: A robot simulator for research and education," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 1400–1405.

[98] G. Rong *et al.*, "LGSVL simulator: A high fidelity simulator for autonomous driving," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2020, pp. 1–6.

[99] M. Uřičář, P. Křížek, D. Hurych, I. Sobh, S. Yogamani, and P. Denny, "Yes, we gan: Applying adversarial techniques for autonomous driving," *Electron. Imag.*, vol. 2019, no. 15, pp. 1–48, 2019.

[100] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[101] H. Yan, X. Chen, V. Y. F. Tan, W. Yang, J. Wu, and J. Feng, "Unsupervised image noise modeling with self-consistent GAN," 2019, *arXiv:1906.05762*. [Online]. Available: http://arxiv.org/abs/1906.05762

[102] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4401–4410.
[103] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1125–1134.
[104] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2017, *arXiv:1703.10593*. [Online]. Available: https://arxiv.org/abs/1703.10593
[105] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 700–708.
[106] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional GANs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8798–8807.
[107] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "DeepRoad: GAN-based metamorphic autonomous driving system testing," 2018, *arXiv:1802.02295*. [Online]. Available: http://arxiv.org/abs/1802.02295
[108] Y. Lin, Y. Li, H. Cui, and Z. Feng, "WeaGAN:Generative adversarial network for weather translation of image among multi-domain," in *Proc. 6th Int. Conf. Behav., Econ. Socio-Cultural Comput. (BESC)*, Oct. 2019, pp. 1–5.
[109] H. Zhang, V. Sindagi, and V. M. Patel, "Image de-raining using a conditional generative adversarial network," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 11, pp. 3943–3956, Nov. 2020.
[110] H. Porav, T. Bruls, and P. Newman, "I can see clearly now: Image restoration via de-raining," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 7087–7093.
[111] L. Karacan, Z. Akata, A. Erdem, and E. Erdem, "Learning to generate images of outdoor scenes from attributes and semantic layouts," 2016, *arXiv:1612.00215*. [Online]. Available: http://arxiv.org/abs/1612.00215
[112] X. Ouyang, Y. Cheng, Y. Jiang, C.-L. Li, and P. Zhou, "Pedestrian-synthesis-GAN: Generating pedestrian data in real scene and beyond," 2018, *arXiv:1804.02047*. [Online]. Available: http://arxiv.org/abs/1804.02047
[113] M. Grapinet, P. De Souza, J.-C. Smal, and J.-M. Blosseville, "Characterization and simulation of optical sensors," *Accident Anal. Prevention*, vol. 60, pp. 344–352, Nov. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0001457513001693
[114] D. Gruyer, M. Grapinet, and P. De Souza, "Modeling and validation of a new generic virtual optical sensor for ADAS prototyping," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2012, pp. 969–974.
[115] E. Pollard, D. Gruyer, J.-P. Tarel, S.-S. Ieng, and A. Cord, "Lane marking extraction with combination strategy and comparative evaluation on synthetic and camera images," in *Proc. 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2011, pp. 1741–1746.
[116] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *Proc. 8th Int. Conf. Qual. Multimedia Exper. (QoMEX)*, Jun. 2016, pp. 1–6.
[117] Y. Zhou, S. Song, and N.-M. Cheung, "On classification of distorted images with deep convolutional neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 1213–1217.

Asher Elmquist received the B.S. (Hons.) and M.S. degrees in mechanical engineering from the University of Wisconsin-Madison in 2017 and 2019, respectively, where he is currently pursuing the Ph.D. degree. His research interests include simulating autonomous vehicles and robots, specifically relating to realistic sensor simulation for developing, testing, and evaluating autonomous behavior. While an undergraduate, he received Faustin Prinz Undergraduate Research Fellowship and graduated with honors in research.

Dan Negrut received the Ph.D. degree in mechanical engineering from the University of Iowa in 1998, under the supervision of Prof. Edward J. Haug. He has courtesy appointments with the Department of Computer Sciences and the Department of Electrical and Computer Engineering. He spent six years working at Mechanical Dynamics, Inc. (a software company), Ann Arbor, MI, USA. In 2004, he served as an Adjunct Assistant Professor for the Department of Mathematics, University of Michigan, Ann Arbor. In 2005, he joined as a Visiting Scientist at Argonne National Laboratory, Mathematics and Computer Science Division. He joined the University of Wisconsin-Madison in 2005, where he is Mead Witter Foundation Professor with the Department of Mechanical Engineering. He leads the Simulation-Based Engineering Laboratory. The lab's projects focus on high performance computing, computational dynamics, artificial intelligence, terramechanics, autonomous vehicles, robotics, and fluid-solid interaction problems. His interest is in computational science. Since 2010, he has been an NVIDIA CUDA Fellow. He received the National Science Foundation Career Award in 2009.