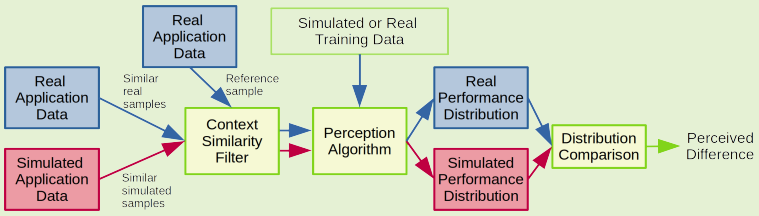
Synthetic image generation for robot simulation: quantifying the impact of model modifications on perception

Asher Elmquist, Radu Serban and Dan Negrut

***Abstract*— Modeling cameras for the simulation of autonomous robotics is critical for generating synthetic images with appropriate realism to effectively evaluate a perception algorithm in simulation. In many cases though, simulated images are produced by traditional rendering techniques that exclude or superficially handle processing steps and aspects encountered in the actual camera pipeline. The purpose of this contribution is to quantify the effect that modifying the camera model has on the perception algorithm evaluated in simulation. We investigate what happens if one ignores aspects tied to processes from within the physical camera, e.g., lens distortion, noise, and signal processing; scene effects, e.g., lighting and reflection; and rendering quality. The results quantifiably indicate that, for the evaluated task, modeling modifications that result in large-scale changes to color, scene, and location had far greater impact on perception than model aspects concerned with local, feature-level artifacts.**



***Index Terms*— camera modeling and simulation, model analysis, image-based perception**

I. Introduction

SYNTHETIC image generation is a prerequisite for simulating robots as image-based perception is foundational for robot autonomy. The quality of image generation is dictated by the fidelity of the camera models used to capture the appearance of the environment in which the robot operates. Research in modeling and simulation has been steadily improving the visual quality of simulation engines owing to the ongoing development of photorealistic techniques for games and rendering engines [1], [2]. However, autonomy perception tasks are not the primary consideration of these gaming graphics improvements. For simulation of autonomy, the downstream consumers of images are computer-based perception algorithms, not humans. Therefore, photorealism in these applications should be determined by the perception algorithm, not the human eye.

In recent work [3], we introduced a methodology that leverages a perception algorithm, e.g., object detection or semantic segmentation, to quantify the difference between simulated (synthetic) and real images. Herein, we seek to use this technique to understand which aspects of camera simulation have significant impact on the performance of the downstream perception algorithm in an autonomy task. This exercise reveals which aspects of the camera simulation provide the greatest return on investment and warrant further investigation for improving the use of simulation in robotics. To refer to the components of simulation which seeks to capture some quality of a real camera or real images, we use the term "camera model." These can include physics-based models of the camera internals or empirical models of image phenomena which are calibrated from real data. Specifically, in this paper we assess the degree to which (a) refinements in a camera model's components, and (b) image enhancements techniques, impact the performance of two object detection and image segmentation tasks. At the onset of this study, one should keep in mind that fast simulation places stringent constraints on the level of sophistication that a camera model can possess when used in robot simulation. Indeed, real-time or faster simulation speed inevitably limits the level of sophistication that a camera model can enjoy.

The two tasks considered in this work are an indoor lab environment and an on-road urban environment. The lab environment allows a high degree of configuration of the environment and simulation, giving insights into specific aspects of the camera model and potential image modifications. The lab environment is associated with the research of a 1/6th scale autonomous vehicle to understand how the behavior in simulation reflects the behavior in reality [4]. This task and associated environment allow for testing in real scenarios which can be subsequently recreated in simulation. From this application, we can identify modifications to simulation that demonstrate significant impact on the object detector used by the scaled autonomous vehicle. The urban environment allows

A. Elmquist, R. Serban and D. Negrut are with the University of Wisconsin-Madison. emails: {amelmquist,serban,negrut}@wisc.edu

the conclusions drawn from the indoor lab to be tested in an independent environment and using a different perception algorithm (semantic segmentation), thus giving insights into the generalization of the conclusions.

The contributions of this manuscript are summarized as follows.

1) We quantitatively measure the impact of a subset of camera model components and image alterations on the behavior of a down-stream perception algorithm.
2) We quantify the relative importance of scene characteristics and image-level alterations, such as lighting and color representation, as compared to pixel-level artifacts such as noise.

The two environments, along with simulation modifications considered, are described in Sec. II and Sec. III. The dataset and model overview is followed by a description of the comparison approach in Sec. IV and results in Sec. V.

A. Related Work

There are three main areas of modeling and simulating images covered by this contribution: camera modeling, camera validation, and the study of camera model effects. The most comprehensive and high-fidelity camera modeling and simulation known to the authors comes from the field of computational camera modeling where the focus is the ability to simulate prototype camera designs. While this field focuses more extensively on the camera internals, typically at the expense of simulation speed and versatility, it also provides high-fidelity models of the entire camera system.

These approaches can capture spectral scene modeling, full lens characteristics, precise noise estimation, and complete image processing. Examples of the full pipeline can be found in [5], [6]. An overview of camera modeling and closed-loop simulation for robotics, and the inclusion of camera component models in such frameworks, can be found in [7].

Often, to validate such component models, research focuses on pixel- and structural-level comparison between simulated and real images, with comparison performed at a data level. Examples of this methodology are found in [8]–[10] and typically require highly controlled and simple scenes. The approach has been shown to be effective in validation of component level models, and was proposed as a two-stage validation strategy in [11].

While such approaches can be beneficial for understanding pixel-level models, they place an unnecessary burden on camera simulation, potentially resulting in models that are higher fidelity than needed for perception. Additionally, the controlled nature of the scenes that can be validated makes it difficult to validate the camera models in the complex, uncontrollable scenes where the cameras and associated autonomy algorithms will operate. Therefore, in this work we perform the analysis using an application-focused strategy, where comparison is conducted at the output of a downstream perception algorithm. Our work builds on a recent validation methodology introduced in [3] and [12].

For the model component evaluation, our work follows a set of inquiries similar to those of Liu et al. [13] who analyzed camera model parameters to understand their impact on object detection generalization to other datasets. Our work differs in three primary respects:

- We are not interested in the generalization of the perception algorithm, but instead are interested in the predictive nature of the simulation (i.e. the performance in simulation accurately representing the performance in reality). In simulation, we want to be able to accurately predict real-behavior, even for networks with high specificity. For a given perception algorithm, we seek to produce a simulator that accurately predicts the algorithm's performance in reality, such that lessons learned in closed-loop testing of the algorithm can be reliable and general.
- We embrace a closed-loop simulation perspective rather than a camera prototyping and system simulation perspective as with ISET/ISET-Auto [6], [14]. Consequently, the rendering pipeline can produce sequences of images in a rapid and efficient manner. To that end, we are interested in understanding which types of models induce significant response in the perception algorithms tested in simulation in order to understand where higher-fidelity camera models might warrant further study and/or modeling effort.
- We are less interested in the effect of the model parameters and more interested in the impact of the model or image transformation itself to understand if it has an impact on the perception algorithm.

Additional work from Liu et al. [15] considered the prototyping of new sensor designs to optimize perception algorithm capability and therefore further considered the impact of sensor parameters on the resulting images and subsequent detection algorithm. Where the current work considers parameter effect, we look at model effect to understand the fidelity of simulation needed for closed-loop and online testing of autonomous systems.

Other studies have been conducted on the impact of artifacts on neural networks, by and large the technology of choice for image-based perception, but have focused more on the impact on perception of general artifacts (increasing magnitude of noise, image blur, etc.) rather than the camera models and the impact of their associated phenomena on image-based perception [16], [17].

II. Camera Models and Indoor Lab Datasets

A. Simulation framework

For the lab environment, simulation is provided by Chrono [18], [19], with the camera simulated using Chrono::Sensor [20]. Chrono::Sensor is a module of Chrono that leverages physically-based rendering and real-time ray tracing [21] to allow modeling of sensors for robotic applications. Beyond camera simulation, Chrono::Sensor supports several other sensors, e.g., lidar, radar, and IMU. The camera model supports geometric lens distortion using either the radial model [22] or FOV model [22] of geometric distortion. It allows noise that can be linearly dependent on pixel intensity. Lighting and material modeling in Chrono::Sensor is based on physically-based rendering techniques [23] to represent diffuse and specular reflections. Optionally, Chrono::Sensor can capture indirect

(a) Real image (b) Simulated image

Fig. 1: Example real and simulated indoor lab environment with cones.

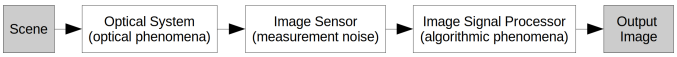


Fig. 2: Illustration of the imaging pipeline, showing the progression of light and data through a camera.

lighting using a stochastic sampling technique combined with the OptiX denoiser [21]. Because Chrono is focused on time-evolving simulations, Chrono::Sensor supports sensor update lag and motion blur; however the effect of time-based distortion is beyond the scope of this study.

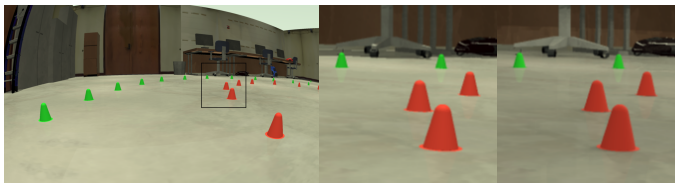
The virtual lab environment was constructed in Blender from reference image of the real lab. The cone locations where measured using a motion capture system and placed in simulation to reflect the real configuration. Models of the cones were generated in CAD with appropriate geometry and color. The cone and environment geometry and textures were based on the physical version, and not on the visual reconstruction of the objects in Chrono. The simulation scene was lit with 20 point lights to approximate the ceiling lights of the real environment.

The camera model is based on the real camera which was used to collect images in the real lab. The update rate was measured from the real camera, and images were taken from the same position as with the real camera, as measured by the motion capture system. The lens model in Chrono was calibrated from the real camera using MATLAB's image calibration toolbox [24]. Example real and simulated cone images are provided in Fig. 1.

B. Overview and modeling pipeline

Camera modeling can be broken down into component models of the imaging pipeline illustrated in Fig. 2, which are typically implemented within a render engine or graphics pipeline. A detailed description of this pipeline is available in [25]. More information on camera modeling, specifically for autonomous vehicle and robotics, is provided in [7]. This contribution considers each section of the imaging pipeline to understand how it impacts the accuracy of an image-based robot perception algorithm.

For each component of the camera pipeline, we consider a limited set of camera models and image transformation which could alter how the perception algorithm processes the simulated images. When the considered model change directly relates to a potential model of the real camera, we quantify the magnitude of improved realism. When the model is not based on the real camera, we instead seek to understand how sensitive the perception algorithm is to the considered



(a) Full image from single-light simulation (modified) (b) Single light (modified) (c) Multiple lights (baseline)

Fig. 3: Example of modifying simulation by using a single point light source (modified) vs 20 point lights (baseline). Note the difference in the shading of the cones.

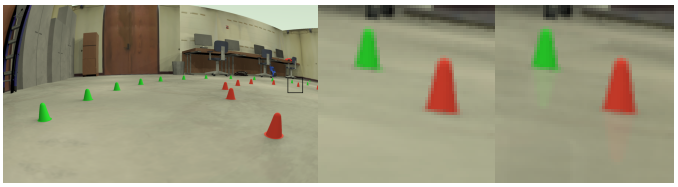
change. Each time a component is considered, the initial simulation is referred to as the "baseline" and the simulation with the considered alteration is referred to as "modified". An illustrative example is provided each time such a modification is introduced. An example of the baseline simulation is shown in Fig. 1b.

C. Scene modeling

In the use of simulated data for training perception algorithms, the virtual environment is understood to play a major role in the realism of simulated data. The domain gap (sim-to-real gap) is a combination of both the appearance and the content differences [26]. The content differences can induce bias in training and are largely attributable to more variety and entropy present in real environments. Appearance, or how the objects are represented in the data, including texture differences, texture variation, object shapes and colors, scene lighting, etc., can also be significant in testing and training perception algorithms. A question we consider in this paper is the level of importance of scene appearance relative to camera modeling. The effect of content modifications on simulation are beyond the scope of this paper.

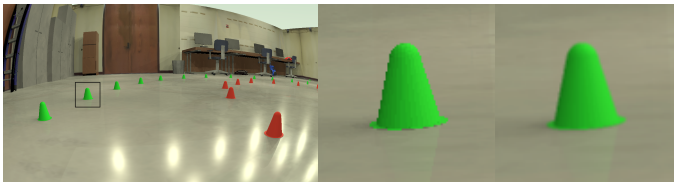
To investigate the impact of changes to the environment, we consider the lighting of the scene as well as the reflection of cones on the floor as these could play a significant role in object detection. The two light configurations tested are: (baseline simulation) 20 point lights that approximate the ceiling lights in the lab and create specular highlights on the floor; and (modified simulation) a single point light above the environment which provides even lighting and no reflective highlights. Example of the single-light configuration, with a highlighted region of interest is shown in Fig. 3. For the single light case, the illumination of the cones causes different shading and results in no specular highlights on the floor, even though the floor in both cases has the same reflectance parameters (note the same cone reflections).

Since reflections were prominent in the real images (see Fig. 1), the floor material parameters were calibrated to produce visually similar reflections for the baseline simulation. A simulation variant with no floor reflections is also considered to evaluate the impact of the reflections on the perception algorithm. See Fig. 4 for an example of the difference with no reflection on the floor. To remove the reflections of the cones, the floor's roughness parameter was set to 1.0 and the metallic parameter set to 0.0 such that no reflections were possible.



(a) Full image from simulation without reflection (modified) (b) No reflections (modified) (c) Reflections (baseline)

Fig. 4: Example of modifying simulation using floor material parameters to limit reflections. Reflection in controlled via the roughness and metallic material parameters [23].



(a) Full image from simulation with single sample per pixel (modified) (b) Single sample (modified) (c) Four samples (baseline)

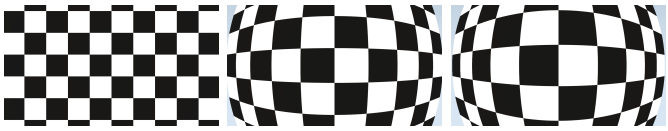
Fig. 5: Example of altering simulation to use a single ray-per pixel (modified) rather than four rays-per-pixel (baseline). Note the hard edges of the single ray-per-pixel image.

D. Data acquisition: image quality

Rendering can be performed via two general algorithms: rasterization and ray tracing. As the simulation in this contribution uses ray tracing, we will focus on the number of rays that are used, per pixel, to sample the scene to determine the final colors in the image. When a single ray is used, each pixel can only include the color from a single object. For example, near the edge of a cone, a ray either hits the cone or goes past, so the edges of object will be hard and zoomed-in views of those obstacles may appear pixelated (see Fig. 5b). When additional rays (in this case 4) are used, these can be used to super sample the pixel in multiple locations, producing an anti-aliased image and a better approximation of soft edges for real images. Due to how the camera collects data, a model with multiple samples represents a higher-fidelity model of light acquisition. To compare the impact of this image quality, we generated a modified dataset using one sample (ray) per pixel (SPP), with the baseline using four SPP. An example image rendered with one SPP, with a highlighted region of interest, alongside the same region of interest from the baseline simulation is shown in Fig. 5.

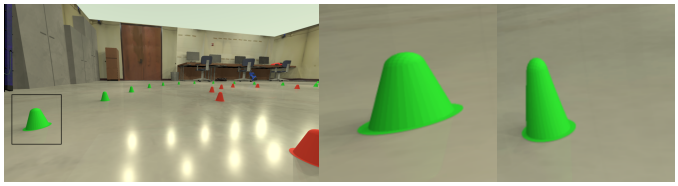
E. Optical model: lens distortion

Lens systems play a key role in the accumulation of light on the image sensor. Particularly when considering wide angle lenses, geometric distortion can be substantial near the edges of the image. The real camera used for data collection in the lab environment demonstrates this wide-angle geometric distortion. As such, we consider three models of the lens with increasing fidelity. First, a pinhole camera is used, which is a common non-distorted model employed in game engines. Second, we analyze a single-parameter model based on geometric



(a) Pinhole model (b) Radial model (c) FOV model

Fig. 6: Checkerboard rendered with each of the three lens models (left-to-right: pinhole, radial, fov-model). Each model is set to have the same effective horizontal field-of-view.



(a) Full image from simulation with pinhole model (modified) (b) Pinhole model (modified) (c) Radial model (baseline)

Fig. 7: Example of altering simulation by removing lens distortion in the camera. Note the wider appearance of cones near the edges of the image relative to the baseline simulation.

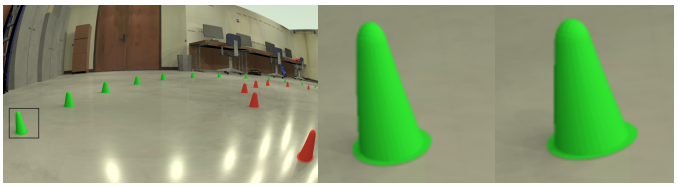
considerations of a single, spherical lens. This is called the FOV model [22] and can recreate limited geometric distortion. Lastly, we consider the radial model [22] which is commonly used in computer vision. This distorts pixels radially based on polynomial coefficients which can be calibrated from real images, for example using the MATLAB image calibration toolbox [24].

For the cone environment simulation, the radial model was calibrated to match the distortion measured on the real camera. After calibration, the radial model was found to give geometric distortions accurate to within 1% of the distortion of the real camera. The implemented FOV model was parameterized by the calibrated field of view from the radial model. Finally, the pinhole model was parameterized to have the same effective horizontal field of view as the radial model to limit the difference in the content obtained within the images.

The baseline simulation used the radial distortion model. However, here we seek to understand the magnitude of the perceived difference between the other two models supported in Chrono::Sensor (pinhole and FOV models). The cone dataset was therefore regenerated with the pinhole model and the radial model. An example of using the pinhole model as the modified simulation is shown in Fig. 7, which highlight significant geometric distortion of the cones near the edge of the images. An example of the FOV model used to alter the simulation is shown in Fig. 8. This example shows that the FOV model can recreate similar distortions to those of the radial model.

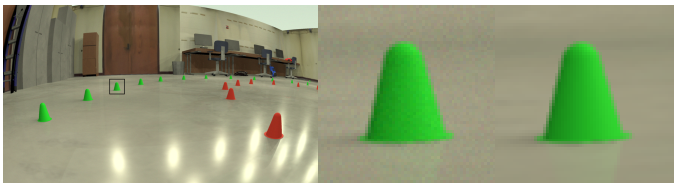
F. Image sensor: noise

The image sensor's noise can be modeled using several approaches which have been found to effectively approximate noise and the image sensor level [28]–[30]. Accurate noise modeling in the image, however, is not straightforward, since onboard processing of the raw image can significantly distort the pixel- and chromatically-independent noise through



(a) Full image from simulation with FOV model (modified) (b) FOV model (modified) (c) Radial model (baseline)

Fig. 8: Example of altering simulation with a lower-fidelity, single-parameter distortion model (FOV model [27]) in the simulation. Note the slight difference in cones near the edges of the image.



(a) Full image from simulation with AWGN (modified) (b) AWGN (modified) (c) No noise (baseline)

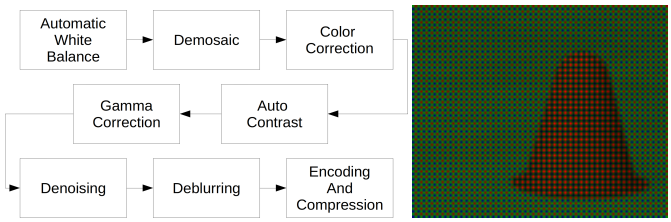
Fig. 9: Example of altering simulation with a low-fidelity noise model by adding white Gaussian noise ($\sigma = 0.01$) to the simulated cone images.

algorithms such as demosaicing, denoising, deblurring, gamma correction, and compression. Therefore, these existing noise models may not precisely capture the noise in the final image.

Due to images being rendered in sRGB space, estimated noise on the real images being low, and the scene radiance being unknown, higher fidelity models were excluded from comparison in favor of understanding the more general impact of noise on training and testing perception. The noise model considered here is an additive white Gaussian noise (AWGN). AWGN is a simplistic noise model applied to make the data less idealistic and often only used when the noise is of unknown origin. The estimated noise of the real images had standard deviation less than half the image precision. This indicates that, between image processing and downsizing, very little noise remained and as such noise was excluded from the baseline simulation. Although the real data demonstrated negligible noise, for the study, we seek to understand the impact of noise on a downstream perception algorithm. Since we will not compare realism, we can chose the noise levels of interest based on other considerations. To understand the impact of this change, AWGN was applied with $\sigma = 0.01$ to images with color 0-1. For an 8-bit-per-channel image, this is approximately 2.5$\times$ the image precision. We will consider and discuss the choice of magnitude further when analyzing the impact in Section V. Example of a noisy image can be seen in Fig. 9.

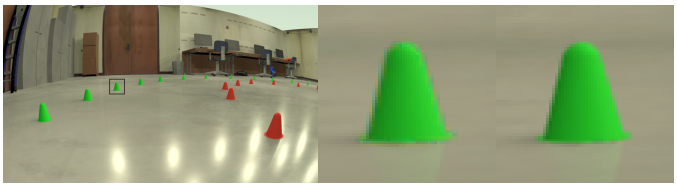
G. Image signal processing (ISP): demosaicing, exposure, and color balance

Cameras capture intensity over an array of light-sensitive pixels, called the image sensor. To capture distinct red, green, and blue values that make up the three channels of an RGB



(a) Example ISP [33] showing a collection of algorithms that can alter the quality of an image. (b) Image viewed with Bayer RGGB pattern

Fig. 10: Image signal processing



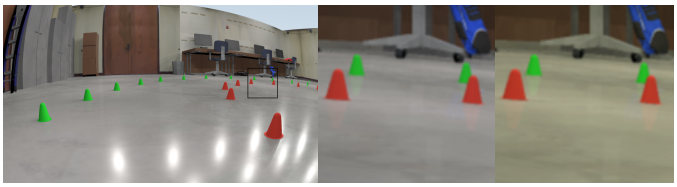
(a) Full image from simulation with demosaicing (modified) (b) With demosaicing (modified) (c) Without demosaicing (baseline)

Fig. 11: Example of altering simulation by recreating a raw-to-RGB demosaicing conversion in the camera.

image, the sensor leverages a mosaic of color filters which provide the intensity of light at a given location for a single channel. A common filter pattern is the red-green-green-blue (RGGB) Bayer pattern, which we will consider here. Other patterns, such as RCCC (red-clear-clear-clear), also exist for dedicated purposes in automotive sensing. The image collected by this pattern is the raw image which is then converted to an RGB image through a demosaicing process that interpolates the missing RGB values from nearby pixels [31]. The image can be further processed by the ISP to denoise, deblur, color correct, white balance, or otherwise modify the image. A set of example post-sensor processing operations is provided in Fig. 10a. These post-sensor operations are often overlooked when simulating robotics using off-the-shelf game or rendering engines, thus failing to capture potentially significant alterations to the final image. Here, we consider the effect of three post-sensor algorithms: demosaicing, exposure, and white balancing.

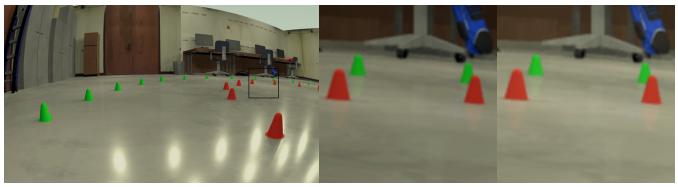
Since real images are often captured with a Bayer pattern, demosaicing can soften object edges and lead to slight color bleeding. For the indoor environment, we do not know which demosaicing algorithm is used onboard the camera, nor do we have a straight-forward way of measuring it. Therefore, for this environment we simulate images as RGB, then convert to raw by sampling the color in the Bayer pattern, see Fig. 10b. We then use OpenCV [32] and an edge aware demosaicing algorithm to convert back to RGB. An example of the resulting demosaiced image is shown in Fig. 11. Because this process is not based on the real data or sensor, we will consider this only to understand the induced change in the perception algorithm and not the realism of the model.

To understand the impact of exposure and color balance, we seek to leverage post-processing color transformations that approximate the real data, rather than modeling internals in the



(a) Full image from simulation without color gain matching (modified) (b) No color gain matching (modified) (c) With color gain matching (baseline)

Fig. 12: Example of altering simulation by ignoring the color-gain matching transform used in the baseline simulation.



(a) Full image from simulation with brightness matching (modified) (b) Brightness matching (modified) (c) No brightness matching (baseline)

Fig. 13: Example of altering simulation by matching the brightness levels measured in the real images.

camera, to which we are not privy. For both exposure and color balance, we begin with simple models of these transforms in this paper and acknowledge that these will not faithfully model the real camera.

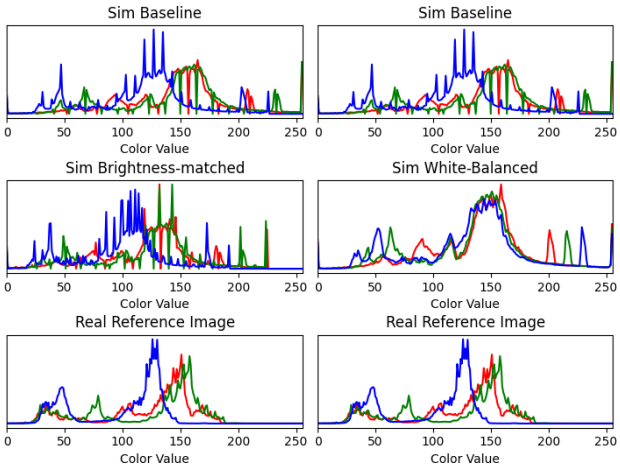
Using a traditional computer graphics pipeline, which typically does not consider pixel sensitivities, results in rendered data with correct color balancing and white balancing. During real image dataset capture, it was noted that the images were not white balanced, either because no white balancing was enabled on the camera or because the white balancing algorithm was insufficient. As this significantly changed the overall color of the images, we modeled it in simulation by adjusting the color gains to match the mean white balance gains from a real calibration set. The algorithm followed a gray-world white balancing method, but distorted the image away from a gray world, to the mean of the real images. While this is a simple empirical model, it demonstrated improved results in prior object detector training. To understand the impact of this decision, we compared the baseline simulation (with the color distortion) to the simulation without this model (with correct color reproduction). The correctly white balanced version is the simulation without the color-distortion model. An example white-balanced image is provided in Fig. 12.

To understand the impact of exposure, we note the difference in the mean and standard deviations of brightness in simulated vs. real images. We then altered the simulated images so that their mean and standard deviations match the mean of a calibration set consisting of real images. This crude model is empirical rather than physical, and is implemented as

$$I_s(x,y) = (I_s(x,y) - \overline{I_s}) \cdot \frac{\sigma(I_r)}{\sigma(I_s)} + \overline{I_r}\,,$$

where $I_s(x,y)$ is the simulated image at pixel x, y, $\overline{I_s}$ and $\overline{I_r}$ are the simulated and real image means, and $\sigma(I_r)$ and $\sigma(I_s)$ are the real and simulated image standard deviation, respectively. While cameras do not use this to correct exposure, it allows us to modify the simulation to produce a similar visual effect to the real data with a slight darkening of the simulated images, as seen in Fig. 13. A comparison of the histograms between baseline simulation, modified simulation, and real image are provided in Fig. 14a. Other auto-contrast/brightening algorithms exist [34], [35], but would be challenging to use for matching the mean characteristics of the real data.

Additionally, we consider a variation of the exposure model for the case where the mean and standard deviation of the color channels had been perfectly reconstructed for each image.



(a) Brightness matching (b) White-balanced

Fig. 14: Plots of the color histogram for an image from the baseline simulation which includes color distortion, modified simulation which excludes color distortion, modified simulation which additionally matches the brightness level of the real dataset, and a real image. Note the shift in the histogram of the simulated image, which indicates the effect of the brightness and color unbalancing approach.

This is possible since the metric dataset has sim-real images pairs. For the detector training dataset, we consider a stochastic pairing between the sim and real detector training sets. This results in matching not only the exposure, but also the variation of exposure in the datasets.

III. City Environment Datasets

While highly controllable, the cone environment is simplistic, meaning the analysis in this application may not translate to other, more complex tasks and environments. Therefore, we consider a second environment which is more applicable to the broader robotics community in the form of a on-road urban environment. In this environment, we consider semantic segmentation rather than object detection to broaden the analysis to a second perception algorithm.

To serve as real data for an urban environment, we use Cityscapes [36] (2,389 images), which is a widely used benchmark urban dataset for perception. This dataset includes a semantic map for each image, with labels for cars, pedestrians, bicycles, roadways, signs, and other urban objects. As a simulated counterpart, we use the GTAV dataset [37] (24,966

(a) Real image (Cityscapes) (b) Simulated image (GTAV)

Fig. 15: Example images from the real (Cityscapes) and simulated (GTAV) city environment datasets.

(a) Full image from GTAV with radial distortion (modified) (b) Radial model (modified) (c) Pinhole model (baseline)

Fig. 16: Example of altering GTAV with geometric lens distortion based on a radial model whose parameters come from the camera used to capture cone images.

images) which is qualitatively similar in content, and semantically consistent to Cityscapes, but was generated by its authors from the popular video game GTA-V. The GTAV dataset does not specifically attempt to model the camera or environment of Cityscapes. Furthermore, it is not transparent which models precisely are implemented to model the camera and scene in GTAV. However, for those that can be measured, we can still seek to understand the data and models empirically. Example Cityscapes (real) and GTAV (simulated) images are provided in Fig. 15. The GTAV data without modifications is treated as the baseline.

Scene modeling. Since the GTAV dataset is provided directly, we have no control and cannot modify aspects of the environment such as lighting, materials, etc. These effects are therefore ignored in this study. For post-processing GTAV images and enhancing the image's scene appearance, see work related to generative adversarial networks such as [38] and a related analysis of this algorithm on realism for semantic segmentation [12].

Data acquisition: image quality. Similar to scene modeling, we ignore the impact of rendering quality as we cannot appropriately alter the data in the post-process stage.

Optical model: lens distortion. While it is unknown if a lens-distortion model was applied during the rendering of GTAV, we make the assumption here that a pinhole camera was used. This is very common in game engines and this assumption is supported by the fact that objects near the edge of the images appeared undistorted. Therefore, we seek to understand if applying lens distortion would have changed the behavior of perception on the GTAV images. To do so, we use a radial model with calibrated parameters from the lab environment. We use these parameters as a control to understand if the difference between distorted and undistorted images is equivalent between the two applications. An example of a distorted GTAV image can be seen in Fig. 16.

(a) Full image from GTAV with AWGN (modified) (b) AWGN (modified) (c) No noise (baseline)

Fig. 17: Example of altering GTAV by adding white Gaussian noise ($\sigma = 0.01$).

(a) Full image from GTAV with color gain matching (modified) (b) With color gain matching (modified) (c) No color gain matching (baseline)

Fig. 18: Example of modifying GTAV by matching the mean color-gains measured from the Cityscapes dataset.

Image sensor: noise. As with real cones, Cityscapes and GTAV had low estimated noise level, lower than half the image precision. It is likely that GTAV had little to no noise initially, and between any onboard processing for Cityscapes and the downsizing done herein, noise was minimal. However, we still seek to understand the degree to which any impact of noise observed in the lab environment is reproducible in the urban environment. Therefore, similar to the cone dataset, AWGN was applied with $\sigma = 0.01$. Example of the noised images for GTAV can be seen in Fig. 17.

ISP: demosaicing, exposure, and color balance. Because of the aforementioned downsizing, we ignore demosaicing as a modification to GTAV as it would only impact fine-grain detail. The two post-sensor alterations we consider for GTAV are white balancing and brightness adjustment similar to the consideration in the indoor lab environment. We model these changes in the same way as for cones images, where white-balancing brings the GTAV data into the same off-white mode as a Cityscapes calibration set. For brightness matching, we again adjust the GTAV images to match the mean and standard deviation of a set of Cityscapes calibration images. See examples of white balance matching in Fig. 18 and additional brightness matching in Fig. 19. While this model is low-fidelity, is it empirical can still approximates the real data. Therefore we will consider both the magnitude of induced change in perception as well as the change in realism due to this model. To demonstrate the color distortion, Fig. 20a shows the histograms of GTAV, Cityscapes, and the white-balance matched image. Figure 20b shows the histograms for a GTAV, Cityscapes, and brightness-matched image.

IV. Comparison Method and Perception Algorithms

A. Machine learned image enhancement

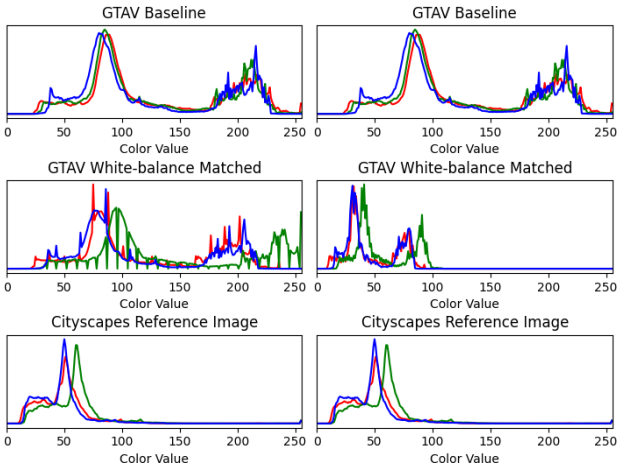
For both the indoor environment and the urban dataset, we

(a) Full image from GTAV with color gain and brightness matching (modified)

(b) Color gain and brightness matching (modified)

(c) Without color gain and brightness matching (baseline)

Fig. 19: Example of altering GTAV by matching the mean color gain and brightness level measured from the Cityscapes dataset.



(a) White-balance matching (b) Brightness-matching

Fig. 20: Plot of the color histogram for an image from the baseline GTAV dataset, a modified GTAV frame which matches the color-gain of the Cityscapes dataset, a modified GTAV frame which matches the color gain and brightness level of the Cityscapes dataset, and an image from the Cityscapes dataset.

consider a machine-learned (ML) image enhancement. The ML enhancement is EPE-GAN ([38]), which is a generative adversarial network (GAN) which is designed to enhance the realism of simulated images. For the urban dataset, [38] provides an off-the-shelf enhanced version of GTAV that has been trained using GTAV and Cityscapes to make alterations to GTAV such that it appears visually similar in quality to Cityscapes. For the cone dataset, we trained a version of EPE-GAN to modify the simulated images to appear visually similar to the real images. The cone images evaluated in this paper to measure the quantities presented were not used to train our variant of EPE-GAN. While this is an important and interesting enhancement to study on its own, it will be used herein only as a reference to contextualize this paper. Specific analysis of EPE-GAN is beyond the scope of this paper and is detailed in [12]. Since EPE-GAN was trained on a real and simulated data, we expect it to produce more realistic data than any of the enhancements herein. However, any benefit is important to quantify as the specific enhancements may induce different responses in the perception algorithm than the image modifications considered in this paper.

B. Comparison Methodology

The methodology we use here to compare the datasets follows the contextualized performance difference (CPD) proposed in [3] and further generalized in [12]. This technique measures the difference in two datasets by analyzing the behavior of a perception algorithm on those two datasets.

CPD is a measure of the difference between datasets using a discriminator/judge that is provided by the application in which simulation is to be used. In this paper, the discriminators are the perception algorithms associated with the cone detection in the indoor environment and semantic segmentation of the urban environment images. The process to measure CPD takes two datasets and finds smaller regions within images in the sets that have high correlation between their labels (i.e., similar content). This allows datasets to be unpaired, with images of different sizes. Using these samples, CPD evaluates the difference in performance of the perception algorithm when encountering similar regions. Further, CPD evaluates the difference using the distribution of performance when encountering the similar regions in order to evaluate the similarity of performance mean and variance between the datasets. For more details, the reader is referred to [3].

When the two analyzed datasets are simulated and real, the difference in behavior on the datasets is a quantity of realism as inferred by the perception algorithm, that is a measure of how different the simulated data is from reality from the perception algorithm's point of view. When the datasets are both simulated (baseline simulation and modified simulation), the measure provides a quantification of the impact of the modification. When the perception algorithm is sensitive to the alteration, the perceived difference (CPD) will be large. Conversely, a low perceived difference (CPD) indicates that the modification has little effect in changing the performance of the perception algorithm.

C. Perception Algorithms

The purpose of the paper is to understand the effect of modifying the simulation on the downstream perception algorithm in a robotic application. While the process for comparison is described above, here we describe the perception algorithms used in the two tasks, and used as the discriminator to measure CPD.

For the cone environment, the cones were specific object of interest to a broader task. Therefore, we used a two-class object detector trained to detect red and green cones. YOLOv5 Nano [39] was chosen for the task based on high accuracy, low inference time, and low memory requirement. For each cone dataset (real, baseline, and each modified variant), a version of the detector was trained using data exclusively from its respective domain. The trained networks were used then as the judge in the similarity comparison to understand the impact of the modification on the perception algorithm. Each network was trained with the same hyper-parameters to convergence. From here, these trained object detectors are referred to as: Net_{real} for the real dataset, $Net_{sim}^{baseline}$ for the

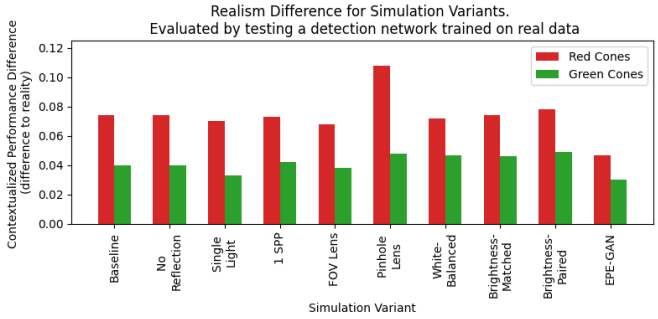


Fig. 21: Perceived difference between each simulation variant and reality when testing Net_{real}. Lower difference indicates more similar behavior on the data.

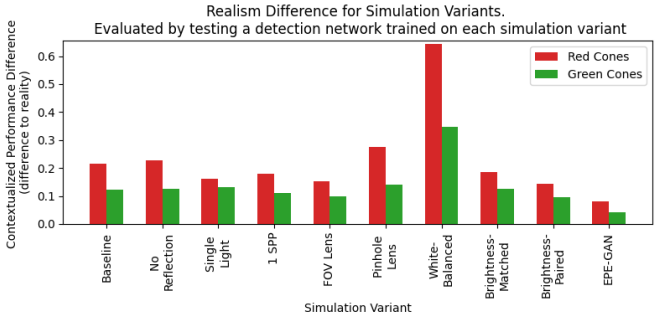


Fig. 22: Perceived difference between each simulation variant and baseline simulation when testing $Net_{sim}^{baseline}$. Lower difference indicates more similar behavior on the data.

baseline dataset, and $Net_{sim}^{modification}$ for the variants. Each time an object detector was trained, a training dataset with random cone locations were used, and was independent of the test set used for evaluation in this paper.

The perception task associated with the Cityscapes benchmark task is semantic segmentation. Since GTAV has semantic labels which are consistent with Cityscapes, the chosen perception algorithm was an off-the-shelf semantic segmentation network from NVIDIA [40], [41]. The network was trained on all existing classes in the datasets including cars, trucks, roads, signs, pedestrians, vegetation, etc. For each city dataset (Cityscapes, GTAV, and GTAV variants) we trained the semantic segmentation network using the same hyperparameters, until convergence, exclusively in its own domain. From here, these semantic segmentation networks are referred to as: Net_{city} for the Cityscapes dataset, Net_{GTA} for the GTAV baseline, and $Net_{GTA}^{modification}$ for the GTAV variants. For Cityscapes, GTAV, and GTAV-EPE, 586 images from each were set aside for testing to ensure the analysis in this paper did not evaluate performance on trained portions of the datasets.

V. Results

A. Indoor lab

First, we evaluated the realism of the indoor lab datasets but testing Net_{real} in each simulation. To do this, we compared $real - sim_{variant}$ from the perspective of Net_{real} for each variant and quantified the difference with CPD. Figure 21 shows the CPD for each of the simulation variants which modeled the real data. The results show that few of the modifications had significant impact on realism. The largest change was when using a pinhole model, where the simulation became far less realistic, likely due to the significant difference in object shape. While the FOV model and single light saw slight improvements, the results were very similar to the baseline. None of the improvements approached the improvement demonstrated by EPE-GAN.

The second quantification of realism leveraged simulation to produce a new object detector and quantify its ability to transfer to reality. For each simulation variant that modeled the real data, we measured the CPD for each class using the network trained in that simulation variant ($Net_{sim}^{variant}$). Figure 22 shows the magnitude of these changes. Again, the results of EPE-GAN are the most predictive of the real performance, with no model variant having produced a network with such similar results. On the other hand, the white-balanced simulation results in poor predictive power, with the performance of $Net_{sim}^{white-balanced}$ on real being far different from its performance on itself and much worse than the baseline simulation. Each other variant was unremarkable relative to these two extremes. In absolute terms, none of these variants are satisfactorily close to reality; indeed, a difference between 0.1 and 0.3 (all other variants), when the full range of accuracy is defined 0-1, is insufficient to instill confidence in the sim-trained network. The last notable conclusion to draw from these results is that for each variant its network performed more similarly for green cones, except for the combined modifications where the performance difference is almost identical. This observation motivates the need for future analysis as it indicates a bias in the sensitivity of the perception algorithm.

As some of the considered modifications do not model the real data, we next considered (irrespective of real data), the difference between the baseline simulation and each modified simulation from the perspective of the object detector. First we tested $Net_{sim}^{baseline}$ on each simulation variant to measure the CPD between baseline and each modification from the perspective of $Net_{sim}^{baseline}$. These results are shown in Fig. 23. Here, we see that most simulation variants induced a response in $Net_{sim}^{baseline}$ that was similar to the baseline dataset, except for two significant outliers. These outliers were EPE-GAN and pinhole. EPE-GAN results were as expected as this was the closest to reality, and introduced features and artifacts that may be difficult for a sim-trained network to detect; see [12] for analysis and the related supplemental material [42] for examples. The second outlier was the pinhole simulation, which suggests that shape was likely a factor in the learned policy in simulation. This was also expected due to the visually large differences in the pinhole example (Fig. 7). Although expected, these quantified results are significant.

Similar to the two realism studies, we next compared CPD with respect to the network trained in each altered dataset. Results of this analysis are shown in Fig. 24. Many of the results seen thus far are reproduced here, with white-balanced, pinhole, and EPE-GAN being among the largest differences.

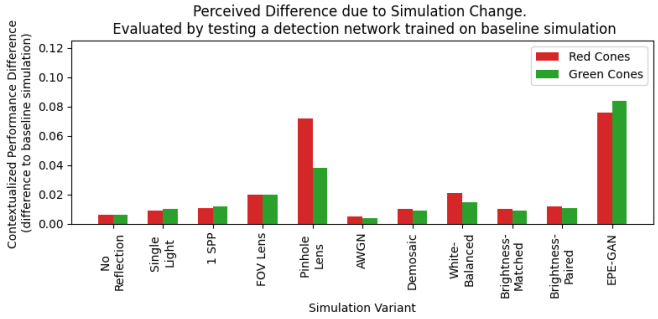


Fig. 23: Perceived difference between each simulation variant and reality when testing each $Net_{sim}^{variant}$. Lower difference indicates more similar behavior on the data.

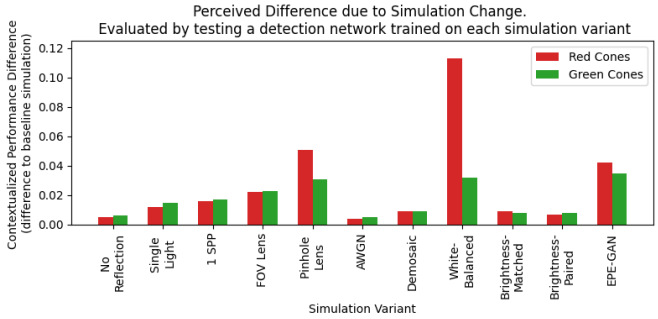


Fig. 24: Perceived difference between each simulation variant and baseline simulation when testing each $Net_{sim}^{variant}$. Lower difference indicates more similar behavior on the data.

It should be noted that the magnitude for all sim variants vs. baseline are lower than that of the variant vs. real data, indicating that the effects of our modifications on the whole are relatively minimal.

B. City environment

To understand the impact of the modifications on GTAV data that modeled Cityscapes, we followed the same procedure as above by beginning with evaluating the performance of Net_{city} on each dataset. This study described the realism of each simulation variant. The results from this study are shown in Fig. 25. Evident in the results is that no modification performed as similar to reality as EPE-GAN. However, the white-balance match dataset is the closest; it is interesting to note however, that full brightness matching does not further improve the dataset.

Next, we ran the comparison a second time, considering how similarly a GTAV-trained network would transfer to Cityscapes. The results in Fig. 26 include two reference marks: GTAV baseline and EPE-GAN, with EPE-GAN producing a network which performed most similar on its own and real data. White-balance matching and brightness matching both significantly altered the trained network and in both cases the network's performance was less similar to reality. This is counter to the results for testing Net_{city} where these modifications slightly improved the GTAV data realism. This behavior, where altering a dataset can improve testing yet reduce the effectiveness of the dataset for training, is similar

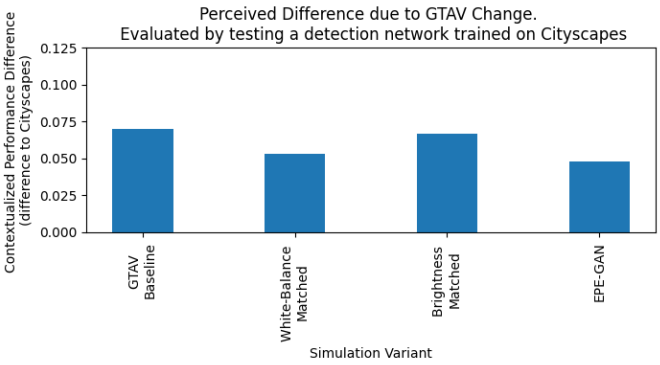


Fig. 25: Perceived difference between each GTAV variant and Cityscapes when testing Net_{city}. Lower difference indicates more similar behavior on the data.

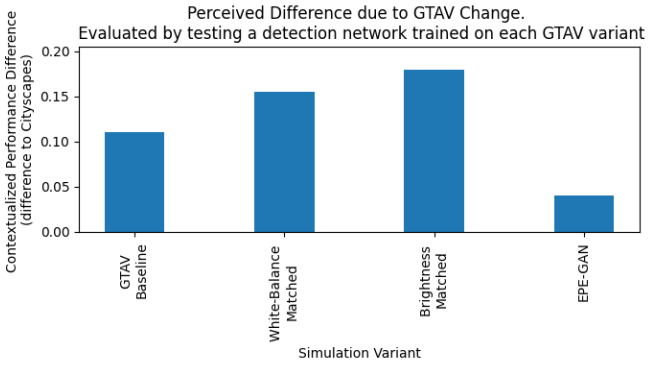


Fig. 26: Perceived difference between each GTAV variant and Cityscapes when testing each $Net_{GTA}^{variant}$. Lower difference indicates more similar behavior on the data.

to previously reported results [12] and is related to dataset variety and network robustness.

The final comparison was designed to quantify, irrespective of Cityscapes, the impact each modification had on the GTAV data from the perspective of Net_{GTA}. The results are shown in Fig. 27. Taking into account the scale of the plot, it is clear here that each modification (apart from GTAV-EPE) induced similar results to that in the baseline, with all differences below 0.02. Only GTAV-EPE induced significantly different performance. This demonstrates the greater impact of scene changes such as the texture changes apparent in the GTAV-EPE images.

Applying AWGN with σ=0.01 induced changes that were less significant than the color or lens distortion changes. With a standard deviation of 0.01, we considered noise approximately 5 times that estimated in the real data. This indicates that, while it may have impact in other applications or for other algorithms, the algorithms for the tasks chosen herein were not sensitive to such image transformations.

VI. Conclusion and Future Work

This contribution assesses the impact of specific camera models and image alterations on the behavior of a downstream perception algorithm used for robot autonomy. The impact of

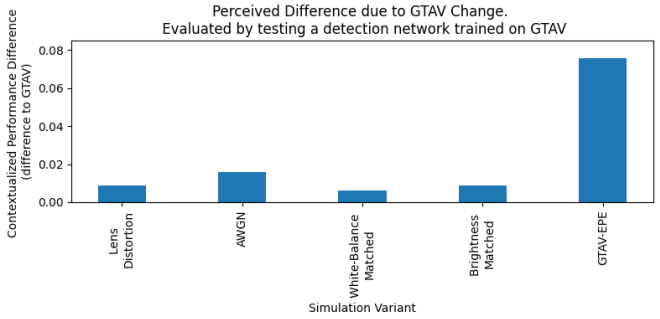


Fig. 27: Perceived difference between each GTAV variant and GTAV when testing each Net_{GTA}. Lower difference indicates more similar behavior on the data.

the changes to simulation is gauged with respect to the perception algorithm using the contextualized performance difference (CPD). Where we modified the camera model to reflect the real camera or real data, we measured the improvement in realism as perceived by an object detector and semantic segmenter. When the change to simulation was independent of reality, we quantified the magnitude of the change of response by the object detector and semantic segmenter in simulation.

The modifications that produced significant change for our two perception algorithms in their respective environments were large-scale modifications to color and shape. We note that lens distortion can be significant as it induces structural changes which impact testing and training. Beyond detection, lens distortion could play a major role in any 3D projection of the image, where the location of pixels would determine 3D size or shape. Other large-scale changes such as color modification were significant factors in this study. Therefore, accurately modeling post-sensor image processing such as auto-contrast, white-balancing, and color correction could be important factors to include in closed-loop testing, particularly if few alterations were performed during training of the perception algorithm. This indicates further investigation into higher-fidelity models of color reproduction and post-sensor processing are warranted.

However, none of the evaluated modifications were as impactful as EPE-GAN, indicating that accurate modeling of the appearance of the virtual environment is critical. This is intuitive as white-balance and contrast are intertwined in the appearance of the virtual environment. As such, a hybrid approach, including precise modeling of distortion and post-sensor color adjustments, along with a GAN for scene-based enhancement could improve the quality of the camera simulators in robotics. These results quantitatively support other works where simulation of the environment played a critical role in the generalization of perception algorithms.

Small and fine-grain detail including noise, demosaicing, image quality, or slight changes to how exposure is modeled (not to be confused with *if* exposure is modeled), resulted in very similar results to the baseline simulation. This indicates little sensitivity to these image features in the perception networks. However, if the application or algorithms were changed, this would need to be verified for the particular simulation use case.

These results indicate that focus and effort for the simulation of these use cases should center on the environment and large-scale phenomena in the camera model. To understand this further, the results found herein indicate potential benefit of studying the impact of higher-fidelity models for these aspects of simulation. This same procedure and methodology can be used to understand the impact of the camera component models on different perception tasks (such as tracking or visual odometry) and different perception algorithms/architectures, in different environments, or using a different measure of autonomous behavior altogether. When simulation is used for each of these tasks, what modeling aspects are important will be determined by the purpose the synthetic images are used for. In each case however, our approach would quantify the impact of the model, giving insights into which aspects of simulation warrant improvement.

ACKNOWLEDGMENT

This work was carried out in part with support from National Science Foundation project CPS1739869. Special thanks to the ARC Lab at the University of Wisconsin-Madison for their support through their motion capture facilities.

REFERENCES

[1] Unity3D, "Real-Time 3D Tools," https://unity3d.com/, 2016, accessed: 2022-12-28.
[2] Epic Games, "Unreal Engine," https://www.unrealengine.com, 2020, accessed: 2021-11-23.
[3] A. Elmquist, R. Serban, and D. Negrut, "A performance contextualization approach to validating camera models for robot simulation," *arXiv preprint arXiv:2208.01022*, 2022.
[4] A. Elmquist, A. Young, I. Mahajan, K. Fahey, A. Dashora, S. Ashokkumar, S. Caldararu, V. Freire, X. Xu, R. Serban, and D. Negrut, "A software toolkit and hardware platform for investigating and comparing robot autonomy algorithms in simulation and reality," *arXiv preprint arXiv:2206.06537*, 2022.
[5] M.-G. Retzlaff, J. Hanika, J. Beyerer, and C. Dachsbacher, "Physically based computer graphics for realistic image formation to simulate optical measurement systems," *Journal of Sensors and Sensor Systems*, vol. 6, no. 1, p. 171, 2017.
[6] H. Blasinski, J. Farrell, T. Lian, Z. Liu, and B. Wandell, "Optimizing image acquisition systems for autonomous driving," *Electronic Imaging*, vol. 2018, no. 5, pp. 161–1, 2018.
[7] A. Elmquist and D. Negrut, "Modeling cameras for autonomous vehicle and robot simulation: An overview," *IEEE Sensors Journal*, vol. 21, pp. 25 547–25 560, 2021.
[8] Z. Lyu, T. Goossens, B. Wandell, and J. Farrell, "Validation of physics-based image systems simulation with 3d scenes," *IEEE Sensors Journal*, 2022.
[9] M. Grapinet, P. De Souza, J.-C. Smal, and J.-M. Blosseville, "Characterization and simulation of optical sensors," *Accident Analysis and Prevention*, vol. 60, pp. 344–352, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0001457513001693
[10] D. Gruyer, M. Grapinet, and P. De Souza, "Modeling and validation of a new generic virtual optical sensor for adas prototyping," in *2012 IEEE Intelligent Vehicles Symposium*. IEEE, 2012, pp. 969–974.
[11] P. J. Durst, D. McInnis, J. Davis, and C. T. Goodin, "A novel framework for verification and validation of simulations of autonomous robots," *Simulation Modelling Practice and Theory*, vol. 117, p. 102515, 2022.
[12] A. Elmquist, R. Serban, and D. Negrut, "Evaluating a GAN for enhancing camera simulation for robotics," *arXiv preprint arXiv:2209.06710*, 2022.
[13] Z. Liu, T. Lian, J. Farrell, and B. A. Wandell, "Neural network generalization: The impact of camera parameters," *IEEE Access*, vol. 8, pp. 10 443–10 454, 2020.
[14] J. E. Farrell, F. Xiao, P. B. Catrysse, and B. A. Wandell, "A simulation tool for evaluating digital camera image quality," in *Image Quality and System Performance*, vol. 5294. International Society for Optics and Photonics, 2003, pp. 124–132.

[15] Z. Liu, T. Lian, J. Farrell, and B. Wandell, "Soft prototyping camera designs for car detection based on a convolutional neural network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.
[16] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2016, pp. 1–6.
[17] Y. Zhou, S. Song, and N.-M. Cheung, "On classification of distorted images with deep convolutional neural networks," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 1213–1217.
[18] Project Chrono, "Chrono: An open source framework for the physics-based simulation of dynamic systems," http://projectchrono.org, 2020, accessed: 2020-03-03.
[19] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. Fleischmann, M. Taylor, H. Sugiyama, and D. Negrut, "Chrono: An open source multi-physics dynamics engine," in *High Performance Computing in Science and Engineering – Lecture Notes in Computer Science*, T. Kozubek, Ed. Springer International Publishing, 2016, pp. 19–49.
[20] A. Elmquist, R. Serban, and D. Negrut, "A sensor simulation framework for training and testing robots and autonomous vehicles," *Journal of Autonomous Vehicles and Systems*, vol. 1, no. 2, p. 021001, 2021.
[21] S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M. Stich, "OptiX: A general purpose ray tracing engine," *ACM Transactions on Graphics*, August 2010.
[22] Z. Tang, R. G. von Gioi, P. Monasse, and J.-M. Morel, "A precision analysis of camera distortion models," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2694–2704, 2017.
[23] B. Burley and W. D. A. Studios, "Physically-based shading at disney," in *ACM SIGGRAPH*, vol. 2012. vol. 2012, 2012, pp. 1–7.
[24] MathWorks, "Using the Single Camera Calibrator App," https://www.mathworks.com/help/vision/ug/using-the-single-camera-calibrator-app.html, 2022, accessed: 2022-11-14.
[25] J. E. Farrell and B. A. Wandell, "Image systems simulation," *Handbook of Digital Imaging*, pp. 1–28, 2015.
[26] A. Prakash, S. Debnath, J.-F. Lafleche, E. Cameracci, S. Birchfield, M. T. Law *et al.*, "Self-supervised real-to-sim scene generation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 044–16 054.
[27] P. Sturm, S. Ramalingam, J.-P. Tardif, S. Gasparini, J. Barreto *et al.*, "Camera models and fundamental concepts used in geometric computer vision," *Foundations and Trends ® in Computer Graphics and Vision*, vol. 6, no. 1–2, pp. 1–183, 2011.
[28] S. W. Hasinoff, F. Durand, and W. T. Freeman, "Noise-optimal capture for high dynamic range photography," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 553–560.
[29] EMVA Standard, "1288, standard for characterization of image sensors and cameras," *European Machine Vision Association*, vol. 3, 2010.
[30] 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 Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 299–314, 2008.
[31] H. S. Malvar, L.-W. He, and R. Cutler, "High-quality linear interpolation for demosaicing of bayer-patterned color images," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3. IEEE, 2004, pp. iii–485.
[32] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
[33] S.-H. Choi, J. Cho, Y.-M. Tai, and S.-W. Lee, "Implementation of an image signal processor for reconfigurable processors," in *2014 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2014, pp. 141–142.
[34] R. Maini and H. Aggarwal, "A comprehensive review of image enhancement techniques," *arXiv preprint arXiv:1003.4053*, 2010.
[35] A. K. Vishwakarma and A. Mishra, "Color image enhancement techniques: a critical review," *Indian J. Comput. Sci. Eng*, vol. 3, no. 1, pp. 39–45, 2012.
[36] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.
[37] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European conference on computer vision*. Springer, 2016, pp. 102–118.
[38] S. R. Richter, H. A. AlHaija, and V. Koltun, "Enhancing photorealism enhancement," *arXiv preprint arXiv:2105.04619*, 2021.
[39] G. Jocher, A. Stoken, J. Borovec, NanoCode012, ChristopherSTAN, L. Changyu, Laughing, tkianai, A. Hogan, lorenzomammana, yxNONG, AlexWang1900, L. Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, F. Ingham, Frederik, Guilhen, Hatovix, J. Poznanski, J. Fang, L. Yu, changyu98, M. Wang, N. Gupta, O. Akhtar, PetrDvoracek, and P. Rai, "ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements," Oct. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4154370
[40] A. Tao, K. Sapra, and B. Catanzaro, "Hierarchical multi-scale attention for semantic segmentation," 2020. [Online]. Available: https://arxiv.org/abs/2005.10821
[41] NVIDIA, "semantic-segmentation," https://github.com/NVIDIA/semantic-segmentation, accessed: 2022-09-14.
[42] A. Elmquist, R. Serban, and D. Negrut, "Evaluating a GAN for enhancing camera simulation for robotics: supplemental material," https://amelmquist.github.io/GANCameraSimulation/, 2022.

Asher Elmquist is a Doctoral candidate in Mechanical Engineering at the University of Wisconsin-Madison. He received a B.S. and M.S. in Mechanical Engineering from the University of Wisconsin-Madison in 2017 and 2019. While an undergraduate, he received the Faustin Prinz Undergraduate Research Fellowship and graduated with Honors in Research. His research focuses on simulating autonomous vehicles and robots, specifically relating to realistic sensor simulation for developing, testing, and evaluating autonomous behavior.

Radu Serban is a Senior Scientist in the Department of Mechanical Engineering at the University of Wisconsin-Madison. Radu received his MS in Aerospace Engineering from the Polytechnic Institute of Bucharest in 1992 and his PhD in Mechanical Engineering from the University of Iowa in 1998. He worked at the University of California - Santa Barbara, in the Center for Applied Scientific Computing at the Lawrence Livermore National Laboratory, and in a Silicon Valley start-up, before joining the University of Wisconsin - Madison in 2013. His research interests are in computational science, numerical analysis, and mathematical software. At LLNL, he was the main architect of the Sundials suite of solvers and one of its lead researchers. Currently, Radu is a main architect of the Project Chrono software and the developer of the Chrono::Vehicle package.

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